Support Vector Machines
Teil 1: Ein theoretischer Überblick

Michael Heinert

Zusammenfassung
Eine in der Geodäsie bisher kaum beachtete Gruppe von Algorithmen zur Interpolation, Regression oder Mustererkennung sind die sogenannten *Support Vector Machines.* Sie sind vielseitig einsetzbar und es existiert im Internet bereits eine Vielzahl von frei erhältlichen Programmen. Daher soll hier ein kurzer Überblick über die theoretischen Grundgedanken und über die mathematischen Zusammenhänge, aber im zweiten Teil dieses Artikels in einem folgenden Heft auch – anhand von praktischen Beispielen – über die Anwendungsmöglichkeiten gegeben werden.

Summary
The support vector machines form a group of algorithms for interpolation, regression or pattern recognition. They are nearly unknown within the scope of geodetic works. These algorithms are suitable for many purposes. In the world wide web, one can find a lot of freeware solutions. A short overview will be presented about the theoretical as well as about the mathematical basics and finally over the potential of these methods. Within one of the next issues, the latter will be explained by presenting several practical examples in the second part of this paper.

1 Einleitung

Weitgehend unerkannt finden sich *Support Vector Machines* (SVM) in Programmen, die bereits von Geodäten verwendet werden, wie beispielsweise in Spezialanwendungen als Klassifizierungsoperatoren in Geoinformationssystemen oder Fernerkundungsauswertungen. Im Gegensatz dazu sind Fuzzy-Regelsysteme oder Künstliche Neuronale Netze schon vielen Geodäten bekannt und werden oft schon ganz selbstverständlich zur Bewältigung geodätischer Aufgaben eingesetzt (Heine 1999; Miima 2002; Akyilmaz und Kutterer 2004; Reiterer 2005, 2006; Heinert und Niemeier 2007; Heine 2008; Heinert 2008a,b, u.v.a.). Dennoch ist die SVM – der weitaus mächtigste lernende Algorithmus – bisher nahezu übersehen worden.

Die *Support Vector Machine* ist ein lernender Algorithmus, der die Beziehung von einer Sammlung von Eingangsdaten zu einer Ausgangsgröße lernt, imitiert, aber in den meisten Fällen identifiziert. Sein entscheidender Vorteil gegenüber allen lernenden Algorithmen liegt in den Strategien zur Vermeidung des sogenannten Übertrainierens, im Englischen als (*overfitting*) bezeichnet (Vapnik 1998; Burges 1998; Haykin 1999; Schölkopf und Smola 2001).

Dieser Algorithmus leitet sich aus der recht komplexen *Statistischen Lerntheorie* her (Vapnik und Chervonenkis 1974; Vapnik 1998) und es gibt bereits eine Reihe von frei verfügbaren Programmen im Internet (Rüping 1999, 2000; Chang und Lin 2000; Schölkopf et al. 2007), die sich recht robust bedienen lassen.

2 Theoretisches Grundprinzip

Was zeichnet eine unbekannte nichtparametrische Modellfunktion $\Phi_{\mathbf{w}}$ aus, die geeignet ist, viele verschiedene Systeme und ihre Prozesse abzubilden? Eine unbedarfte Vermutung würde einer solchen Funktion Komplexität und Nichtlinearität unterstellen. Schließlich muss diese Funktion lineare, periodische oder exponentielle Prozesse gleichermaßen zu beschreiben in der Lage sein. Dabei wäre eine lineare oder linearisierte Funktion doch wünschenswerter, da diese leichter zu handhaben ist.

Eine Lösung ergibt sich theoretisch, wenn der Zustandsraum $\mathbb{R}^n$ der Muster, also der erfassten Datenpaare von Systemein- und -ausgängen, durch eine Abbildung zugunsten eines höherdimensionalen Merkmalsraumes $\mathcal{H}$ verlassen wird. Das ist einerseits möglich, indem zusätzliche bisher nicht quantifizierte Charakteristika der Datenpaare mit berücksichtigt werden. Für ein Bewegungsfeld, das mit geodätischen Methoden punktweise erfasst wurde, kann ein solches Charakteristikum die Bodenbeschaffenheit, die unterliegende geologische Struktur oder schlicht auch die bereits existierende Topographie einschließlich ihrer Ableitungen wie Hangneigung oder Geländekrümmung sein.

Die andere theoretische Möglichkeit besteht in der Erweiterung der Muster um direkt empirisch abgeleitete Größen, wie z. B. einer Kreuzkorrelation zwischen den jeweiligen Elementen.

Bei vielen Modellgleichungen von Kalman-Filtern wird diese Technik in ähnlicher Weise eingesetzt. Hier wird der Zustandsvektor in den meisten Fällen um die jeweiligen Ableitungen erster und zweiter Ordnung erweitert (Minkler und Minkler 1993, S. 108).

Der Vorteil eines höherdimensionalen Merkmalsraumes erschließt sich aus einem umgekehrten Beispiel: In einem dreidimensionalen Raum $\mathcal{H}^3$ befinde sich eine schief liegende Ebene, die von n Punkten repräsentiert wird. Bildet man diese Punkte in die Ebene zweier Koordinatenachsen ab, wird sich eine funktionale Regression dieser als stochastisch erscheinenden Punktwolke als nahezu unmöglich erweisen. Schließlich ist eine wesentliche Information

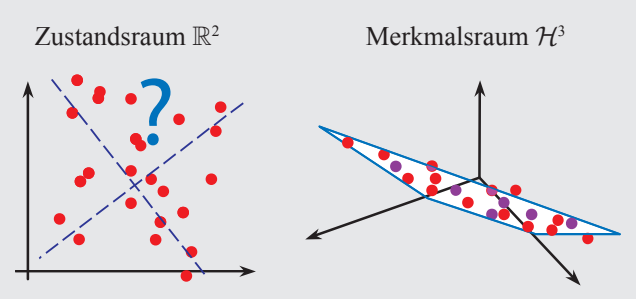

Abb. 1: Im Zustandsraum $\mathbb{R}^2$ versagt die Regression, weil sie eine Projektion einer Hyperebene aus einem Merkmalsraum höherer Ordnung $\mathcal{H}^3$ darstellt.

über diese Punkte verloren gegangen (Abb. 1). Die typische Ausgangssituation ist aber häufig gerade ein solcher abgebildeter Zustandsraum in der eine Modellbildung gestartet werden muss.

Das umgekehrte klassische Beispiel, das eine Problemlösung in einem höherdimensionalen Merkmalsraum beschreibt, ist das sogenannte XOR-Problem. Bei der Booleschen Funktion des »ausschließenden ODERs« – kurz XOR – wird den Wertepaaren $(0,0)$ und $(1,1)$ der Wert 0 und den Paaren $(0,1)$ und $(1,0)$ der Wert 1 zugewiesen. Das XOR-Problem besteht in der Aufgabe, die zweidimensional kartesisch aufgetragenen Wertepaare in Bezug auf ihren Ausgabewert in der Ebene durch eine beliebige Linie zu separieren (Haykin 1999, S. 175 ff. u. S. 259 ff.). In der Ebene ist diese Aufgabe linear unlösbar (Abb. 2a). Spannen aber die Funktionswerte eine weitere Dimension auf, so wird eine Hyperebene den so entstandenen Merkmalsraum geeignet trennen können, dass die Muster gemäß ihres Funktionswertes eindeutig geschieden sind (Abb. 2b).

Beide Beispiele zeigen, dass komplex erscheinende Beziehungen zwischen Systemein- und -ausgängen im Zustandsraum in einem höherdimensionalen Merkmalsraum linear sind oder sich geeignet linear nähern lassen.

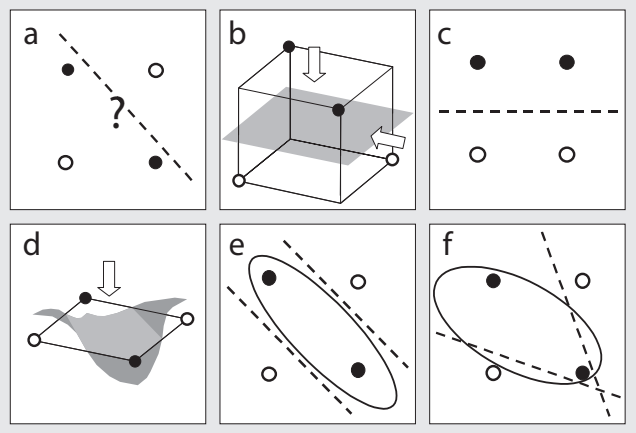

Abb. 2: Die XOR-Dichotomie a) im zweidimensionalen Datenraum, b) im dreidimensionalen Merkmalsraum mit den Normalenvektoren der Abbildungen, c) in einen orthogonalen Datenraum, d) die Abbildung der klassifizierenden Ebene in den Datenraum, die Klassifizierung durch drei Neuronen (gestrichelt) oder zwei RBF-Neuronen (durchgezogen) als e) theoretische und f) praktische Lösung.

Die vorgestellten Beispiele beziehen sich bis hierher noch auf die Mustererkennung, also auf die Trennung von Mustern in Klassen. Die Schwierigkeit, zwei Gruppen von Wertepaaren zu separieren, ist das inverse Problem zu der von uns angestrebten Regression. Die Minimierungsaufgabe für die Muster wird einfach umformuliert: Sollten die Abstände einer separierenden Hyperebene zu den Wertepaaren der beiden Klassen möglichst groß sein, so soll bei der Regression der Abstand zu allen Mustern möglichst klein werden.

3 Mustererkennung

Die SVM wurde ursprünglich zur Musterkennung entwickelt (Vapnik und Chervonenkis 1974; Vapnik 1998). Aus diesem Grund ist es zum Verständnis dieser Algorithmen notwendig, einen Umweg durch die Grundlagen der Mustererkennung zu unternehmen. Erst hiernach werden wir uns der interessanteren Regressionsaufgabe zuwenden können (Haykin 1999; Schölkopf und Smola 2001).

Eine Mustererkennung ist im einfachsten Fall die Separation von gekennzeichneten Mustern in zwei Klassen. Eine SVM als *lineare Maschine* führt diese Aufgabe aus, indem die Muster eben linear separiert werden (Haykin 1999, S. 318). Wie aber schon das Beispiel des ausschließenden Oders (XOR) gezeigt hat, kann eine lineare Trennung im eigentlichen zweidimensionalen Zustandsraum $\mathbb{X} \subset \mathbb{R}^N$, $N \in \mathbb{N}$ möglicherweise gar nicht vorgenommen werden. Die Muster sind nun gerade so im zweidimensionalen Raum angeordnet, dass keine geometrische Lösung zur linearen Trennung existiert. Um die Muster dennoch linear trennen zu können, muss der ursprüngliche Zustandsraum verlassen werden. Es muss zu diesem Zweck vielmehr gelingen, die Muster in einem mindestens dreidimensionalen Merkmalsraum $\mathcal{H}^{n\geq 3}$ neu anzuordnen. Also wird jedes Muster durch die Transformation

$$\mathbf{\Phi} : \mathcal{X} \to \mathcal{H} \quad \text{so dass} \quad x \mapsto \mathbf{x} = \mathbf{\Phi}(x) \tag{1}$$

auf einen Vektor $\mathbf{x}$ im Kern-Hilbert-Raum $\mathcal{H}$ abgebildet (Abb. 2b). Durch eine Hyperebene kann nun die Separation in zwei Klassen erfolgen (Abb. 2c). Mit einem weiteren Kunstgriff – dem sogenannten *kernel-trick* – der Nutzung einer Kernfunktion $\mathcal{K}(x, x')$ gelingt gemäß des Mercer-Theorems die Rücktransformation des höherdimensionalen Raumes in den gegebenen Datenraum (Abb. 2d-f). Der Satz von Mercer besagt vereinfacht, dass eine Hyperebene im Raum höherer Ordnung einen Repräsentanten im ursprünglichen Datenraum besitzt: Nämlich eine stetige, symmetrische, positiv definite Kernfunktion $\mathcal{K}(x, x')$ (Mercer 1909). Eine solche Kernfunktion ist bereits das hier verwendete Skalarprodukt $\langle \mathbf{x}, \mathbf{x}' \rangle = \langle \mathbf{\Phi}(x), \mathbf{\Phi}(x') \rangle$ (siehe Abschn. 4).

3.1 Voronoi-Klassifikator für linear trennbare Muster

Bevor wir in die Welt der n-dimensionalen Hyperräume mit ihren Hyperebenen eindringen, ist es zweckmäßig, die Arbeitsweise eines linearen Klassifikators im zweidimensionalen Datenraum, in diesem Falle anhand eines singulären Voronoi-Klassifikators, zu verstehen. Eine verhältnismäßig einfache, aber zugleich effiziente Methode zur Trennung der m_+ Muster $\mathbf{x}_i$, gekennzeichnet mit $y_i = +1$, und der m_- Muster $\mathbf{x}_j$, ihrerseits gekennzeichnet mit $y_j = -1$, stellt eine »1D-Hyperebene« in der Mitte zwischen den beiden Zentren c_+ und c_- der Stichproben in den Klassen dar (Abb. 3a). Der Vektor, der diese beiden Zentren verbindet, ist parallel zum Normalenvektor der 1D-Hyperebene (Schölkopf und Smola 2001, S. 4 ff.). Hiermit ist es nun möglich, die Kennzeichnung

$$y = \operatorname{sgn}\left(b + \frac{1}{m_+}\sum_{\{i|y_i=+1\}} \mathcal{K}(\mathbf{x},\mathbf{x}_i) - \frac{1}{m_-}\sum_{\{i|y_i=-1\}} \mathcal{K}(\mathbf{x},\mathbf{x}_i)\right) \tag{2}$$

eines beliebigen neu hinzugefügten Musters x zu errechnen. Dabei muss allerdings ein *bias*

$$b = \frac{1}{2m_-^2}\sum_{\{(i,j)|y_i=y_j=-1\}} \mathcal{K}(\mathbf{x}_i,\mathbf{x}_j) - \frac{1}{2m_+^2}\sum_{\{(i,j)|y_i=y_j=+1\}} \mathcal{K}(\mathbf{x}_i,\mathbf{x}_j) \tag{3}$$

berücksichtigt werden, der die Entfernung der Hyperebene zum Koordinatenursprung angibt.

Ein wesentlicher Nachteil des gezeigten Algorithmus ist aber, dass jedes weitere hinzukommende Muster $\mathbf{x}$ das Zentrum der Stichprobe innerhalb seiner Klasse verschieben kann (Abb. 3b). Damit wird auch die trennende Hyperebene gleich mit verschoben. Hierbei muss betont werden, dass der empirische Schwerpunkt der zufällig vorliegenden Stichprobe von Mustern nicht gleich dem theoretischen Zentrum der Klasse ist. Dennoch erzielt dieser Algorithmus gerade mit vielen Mustern passable Ergebnisse, die sich durch die Nutzung einer anderen Kernfunktion, wie hier gezeigt, mit dem Gauß'schen Kern

$$\mathcal{K}_G(\mathbf{x},\mathbf{x}') = e^{\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)}, \tag{4}$$

auf einfache Weise verbessern lassen (Abb. 3c).

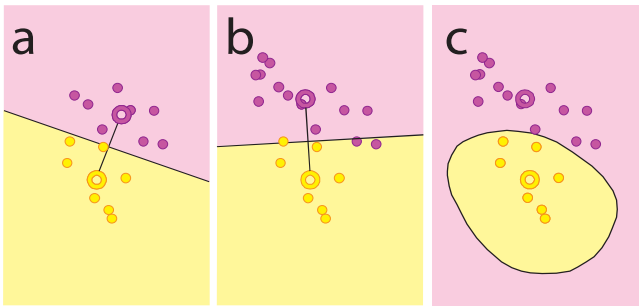

Abb. 3: Der singuläre Voronoi-2D-Klassifikator: Punkte markieren die einzelnen Muster, Ringe deren Schwerpunkte c_+ und c_-. Die Klassifikation erfolgt mittels des Skalarproduktes (a,b) über unterschiedliche Muster aus linear trennbaren Klassen, alternative Klassifikation mittels der Gaußschen Kernfunktion (c).

3.2 SVM für linear trennbare Muster

Eine SVM trennt die beiden Klassen durch eine *optimale Hyperebene* (Haykin 1999, S. 320). Ihre Lösung verwendet anstelle der – wie wir gerade gesehen haben – instabilen Zentren der Klassen vielmehr die Muster, die sich direkt an der Klassengrenze befinden. Nur diese wenigen Muster – die Stützvektoren – entscheiden über die Lage der Hyperebene. Damit verlieren zusätzliche Muster, die irgendwo in der Ferne von der Grenzfläche hinzugefügt werden, ihren Einfluss auf die Lösung. Zur mathematisch-geometrischen Festlegung einer solchen optimalen Hyperebene ist um diese ein symmetrischer Trennbereich – im Englischen als *margin* bezeichnet – zu bilden, der frei von Mustern ist. Dieser Trennbereich soll eine maximale Breite besitzen, damit noch bisher unbekannte Muster zukünftig auch richtig zugeordnet werden können.

3.2.1 Geometrischer Ansatz

Für den einfachsten Fall der linear trennbaren Muster soll exemplarisch die Herleitung der Lösungsfunktion der SVM ausführlich dargestellt werden.

Die gesuchte Hyperebene wird eindeutig und unabhängig von der Dimension des Merkmalsraumes durch die Punkt-Normalenform

$$\mathbf{w}^{\mathsf{T}}(\mathbf{x} - \mathbf{x}_0) = 0 \tag{5}$$

oder vereinfacht durch

$$\mathbf{w}^{\mathsf{T}}\mathbf{x} + b = 0 \quad \text{mit} \quad b = -\mathbf{w}^{\mathsf{T}}\mathbf{x}_0 \tag{6}$$

beschrieben (Haykin 1999, S. 319). Die Trennung der Muster $[\mathbf{x}_i, y_i]$ ergibt sich durch das Einsetzen von $\mathbf{x}_i$ in die Indikatorfunktion, gegeben durch die Punkt-Normalengleichung der Hyperebene mit dem Resultat:

$$\mathbf{w}^{\mathsf{T}}\mathbf{x}_i + b \geq 0 \quad \forall \quad y_i = +1 \tag{7}$$

$$\mathbf{w}^{\mathsf{T}}\mathbf{x}_i + b < 0 \quad \forall \quad y_i = -1. \tag{8}$$

Ein positives oder negatives y_i ordnet das Muster aufgrund der aktuellen Lage der Hyperebene der jeweiligen Klasse zu. Das bedeutet, Muster mit positiven y_i befinden sich auf der dem Ursprung abgewandten Seite der Hyperebene, solche mit negativem y_i auf der zugewandten Seite. In diesem Moment sind die Parameter $\mathbf{w}$ und b aber

noch nicht optimal, daher wird es noch zu fehlerhaften Zuordnungen kommen.

Es sei hier angenommen, dass die Hyperebene bereits näherungsweise so günstig liegt, dass es zu keiner fehlerhaften Zuordnung kommt. Um nun aber einen Trennbereich zwischen den Mustern $[\mathbf{x}_i, y_i]$ zu erschaffen, wird die Forderung aufgestellt:

$$\mathbf{w}^\mathsf{T}\mathbf{x}_i + b \geq +1 \quad \forall \quad y_i = +1 \tag{9}$$

$$\mathbf{w}^\mathsf{T}\mathbf{x}_i + b \leq -1 \quad \forall \quad y_i = -1. \tag{10}$$

Das bedeutet, Muster $[\mathbf{x}_i, y_i]$, für die jetzt noch gilt

$$-1 < \mathbf{w}^\mathsf{T}\mathbf{x}_i + b < +1, \tag{11}$$

befinden sich innerhalb des aktuellen Trennbereichs. Dessen Breite muss also auch angepasst werden.

Multipliziert man hierzu die beiden Ungleichungen (9) und (10) jeweils mit dem Zuordnungsresultat y_i, erhält man den geschlossenen Ausdruck

$$y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b\right) \geq 1. \tag{12}$$

Der ausformulierte Ausdruck

$$y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i - \mathbf{w}^\mathsf{T}\mathbf{x}_0\right) \geq 1 \tag{13}$$

zeigt, dass der einzige variable Anteil im Normalenvektor $\mathbf{w}$ besteht, denn die Trainingsmuster $[\mathbf{x}_i, y_i]$ sind vorgegeben. Auch der – wenngleich noch variable – Vektor $\mathbf{x}_0$, der die Lage des Hauptpunktes der Hyperebene beschreibt, hat keinen Einfluss auf die Breite des Trennbereiches. Daher wird zunächst der m-dimensionale Normalenvektor mit der euklidischen Norm

$$||\mathbf{w}|| = \left(\mathbf{w}^\mathsf{T}\mathbf{w}\right)^{\frac{1}{2}} = \sqrt{w_1^2 + w_2^2 + \ldots + w_m^2} \tag{14}$$

auf den Einheitsvektor

$$\mathbf{w}_0 = \frac{\mathbf{w}}{||\mathbf{w}||} \tag{15}$$

normiert. Dazu ist Gleichung (13) auf beiden Seiten durch $||\mathbf{w}||$ zu teilen. Damit stehen in dem Ausdruck

$$y_i\left(\frac{\mathbf{w}^\mathsf{T}}{||\mathbf{w}||}\mathbf{x}_i - \frac{b}{||\mathbf{w}||}\right) \geq \frac{1}{||\mathbf{w}||} \tag{16}$$

ablesbare euklidische Entfernungen. So ist $\mathbf{w}_0^\mathsf{T}\mathbf{x}_i$ die Projektion eines Mustervektors $\mathbf{x}_i$ auf den Einheitsnormalenvektor $\mathbf{w}_0$. Das entspricht der Entfernung von $\mathbf{x}_i$ zu der Ursprungsebene, die parallel zur Hyperebene liegt (Abb. 4). Der Quotient $-b \cdot ||\mathbf{w}||^{-1}$ ist die Projektion des Hauptpunktes $\mathbf{x}_0$ auf den Einheitsnormalenvektor $\mathbf{w}_0$ und steht für den Abstand der Hyperebene zum Ursprung. Die Differenz aus diesen beiden Ausdrücken bildet also den orthogonalen Abstand $||\mathbf{w}||^{-1}$ des Punktes $\mathbf{x}_i$ zur Hyperebene. In Gleichung (16) erkennt man demzufolge die Hesse-Normalenform der Hyperebene. Hierin kann man

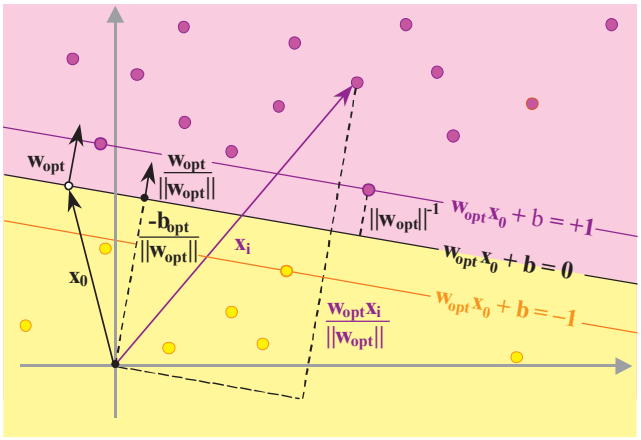

Abb. 4: SVM für linear trennbare Muster: x_0 und w_{opt} legen die Lage und Richtung der trennenden Hyperebene fest. Der Trennbereich wird durch die Lage der Stützvektoren zu beiden Seiten der Hyperebene festgelegt.

den Abstand der Punkte am Rand des Trennbereiches der Hyperebene angeben als

$$\left|\mathbf{w}_0^\mathsf{T}\mathbf{x}_i - b||\mathbf{w}||^{-1}\right| = ||\mathbf{w}||^{-1}. \tag{17}$$

In diesem Ausdruck wird klar, dass die Breite des Trennbereiches als doppelter Abstand der Hyperebene zu den Rändern des Trennbereiches $2 \cdot ||\mathbf{w}||^{-1}$ antiproportional zur Länge des Normalenvektors sein muss.

Wenn also ein möglichst kurzer Normalenvektor $\mathbf{w}$ auf einen breiten Trennbereich führt, so ist zu dessen Maximierung $||\mathbf{w}||$ zu minimieren. Aus der Ausgleichung sind solche Forderungen bekannt (Niemeier 2008, S. 132, 139, 176, 184).

Man minimiere also die quadratische Verlustfunktion

$$\Phi(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w}. \tag{18}$$

Geometrisch bedeutet dieser Minimierungsauftrag folgendes: Die Hyperebene soll sich immer weiter von den Mustern $[\mathbf{x}_i, y_i]$ entfernen. Leider tut sie dieses nicht, indem sie die Muster voneinander trennt, sondern sie entfernt sich von allen Mustern gleichermaßen, indem sie den Zwischenraum zwischen den Klassen verlässt (Abb. 5a). Wird der Normalenvektor – wie gefordert –

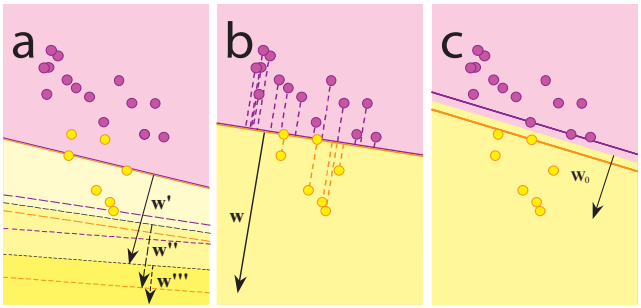

Abb. 5: a) Die Mimimierung von $\mathrm{w}^\mathrm{T}\mathrm{w}$ treibt die Hyperebene von allen Mustern weg, b) bei der Maximierung aller $y_i(\mathrm{w}^\mathrm{T}\mathrm{x}_i - b) - 1$ liegt die Hyperebene an den Punkten der kleineren Stichprobe ohne jeden Trennbereich. c) Beide Bedingungen wirken geeignet zusammen.

infinitesimal klein, degeneriert er im Grenzübergang zu einem Punkt. Das bedeutet, dass die Hyperebene in unendlicher Entfernung um den Schwerpunkt aller Muster rotiert.

Es sind also weitere Randbedingungen nötig, um die Hyperebene in den Bereich zwischen die Klassen zu zwingen. Hierzu kann man fordern, dass alle orthogonalen Abstände auf den positiven, wie negativen Rand des Trennbereichs

$$\sum_i d_i = \sum_{i=0}^{N} y_i \left(\frac{\mathbf{w}^\mathsf{T}}{||\mathbf{w}||} \mathbf{x}_i - \frac{b}{||\mathbf{w}||} \right) - \frac{1}{||\mathbf{w}||} \tag{19}$$

gemeinsam zu maximieren sind. Jede Bedingung für sich ist nur exakt dann erfüllt, wenn a) die Klassenzuordnung y_i stimmt, denn andernfalls wäre das Resultat $d_i \leq -1$, und wenn b) der Trennbereich verlassen worden ist, denn andernfalls wäre das Resultat $-1 < d_i < 0$. Die einzelne Randbedingung

$$d_i = y_i \left(\mathbf{w}^\mathsf{T} \mathbf{x}_i - b \right) - 1 \geq 0 \tag{20}$$

kann schadlos ohne die Normierung des Normalenvektors formuliert werden. Zusammen zwingen sie die Hyperebene zwischen die beiden Klassen $[\mathbf{x}_i, y_i^+]$ und $[\mathbf{x}_i, y_i^-]$. Dabei darf sich die Hyperebene aber einzelnen Mustern unmittelbar nähern, denn mit einer wachsenden euklidischen Norm $||\mathbf{w}||$ wird der Trennbereich infinitesimal schmal und liegt nicht mehr mittig zwischen den beiden Klassen (Abb. 5b). Jede einzelne Randbedingung

$$\alpha_i \left(y_i \left(\mathbf{w}^\mathsf{T} \mathbf{x}_i - b \right) - 1 \right) = 0 \tag{21}$$

wird für sich mit einem Vorfaktor, nämlich einem Lagrange-Multiplikator

$$\alpha_i \geq 0 \tag{22}$$

gewichtet, um zu verhindern, dass alle Muster gleichermaßen – wie zuvor beim Voronoi-Klassifikator – an der Lösung teilhaben. Die Forderung, dass $\alpha_i \geq 0$ sein muss, verhindert, dass sich einzelne Muster bei dieser Optimierung implizit der falschen Klasse zuschlagen lassen (Schölkopf et al. 2007, S. 12 ff.).

Die Gleichungen (20), (21) und (22) heißen Karush-Kuhn-Tucker-Bedingungen (Burges 1998, S. 131). Bei ihnen handelt es sich um allgemein gültige und notwendige Bedingungen der bedingten nichtlinearen Optimierung (Kuhn und Tucker 1951; Hillier und Liebermann 2002).

Damit lässt sich sich eine Lagrange-Funktion in der Form

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} ||\mathbf{w}||^2 - \sum_{i=0}^{N} \alpha_i \left(y_i \left(\mathbf{w}^\mathsf{T} \mathbf{x}_i - b \right) - 1 \right) \tag{23}$$

aufstellen. Diese Funktion muss in Hinblick auf den Normalenvektor $\mathbf{w}$ und den *bias* b minimiert werden.

Es gilt also

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{0} \quad \text{und} \quad \frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0. \tag{24}$$

Die Ableitung nach dem Normalenvektor führt auf

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \frac{1}{2} \cdot 2\mathbf{w} - \sum_{i=0}^{N} \alpha_i y_i \mathbf{x}_i \stackrel{!}{=} \mathbf{0}, \tag{25}$$

was gleichbedeutend ist mit

$$\mathbf{w} = \sum_{i=0}^{N} \alpha_i y_i \mathbf{x}_i. \tag{26}$$

Damit ergibt sich die Berechnung von $\mathbf{w}$ als die Summe über die gewichteten Produkte aus dem Eingang und Ausgang eines jeden Musters. Jedes Produkt entspricht einer unnormierten Kovarianz für ein $\mathbf{x}_i$ und ein y_i.

Die Anwendung der zweiten Ableitung

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = -\sum_{i=0}^{N} \alpha_i y_i \stackrel{!}{=} 0, \tag{27}$$

was gleichbedeutend ist mit

$$\sum_{i=0}^{N} \alpha_i y_i = 0, \tag{28}$$

hat eine interessante geometrische Bedeutung. Aus (28) kann man schließen, dass

$$\underbrace{\sum_i \alpha_i^+ y_i^+}_{\forall i = 0 \ldots n^+} = \underbrace{-\sum_\iota \alpha_\iota^- y_\iota^-}_{\forall \iota = 0 \ldots n^-} \tag{29}$$

gelten muss. Die Anzahl n^+ und n^- der Muster auf der jeweiligen Seite der Hyperebene ist demnach irrelevant. Hierzu erinnere man sich: Auch beim Voronoi-Klassifikator sind die Klassen, vertreten durch ihren Mittelwert, für die Berechnung der Hyperebene gleichbedeutend. Die Gewichtung innerhalb einer Klasse ist bis hierhin für eine SVM allerdings noch offen.

Zur weiteren Lösung wird die Lagrange-Funktion

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w}^\mathsf{T} \mathbf{w} - \sum_{i=0}^{N} \alpha_i y_i \mathbf{w}^\mathsf{T} \mathbf{x}_i - b \sum_{i=0}^{N} \alpha_i y_i + \sum_{i=0}^{N} \alpha_i \tag{30}$$

ausmultipliziert. Aus der Ableitung der Lagrange-Funktion nach ∂b (28) ergibt sich für den dritten Term

$$b \sum_{i=0}^{N} \alpha_i y_i = 0. \tag{31}$$

Setzt man die Gleichung für $\mathbf{w}$ (26) aus der Ableitung nach $\partial\mathbf{w}$ in die ersten beiden Terme der Lagrange-Funktion ein, so gilt

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=0}^{N}\alpha_i y_i \mathbf{x}_i \cdot \sum_{j=0}^{N}\alpha_j y_j \mathbf{x}_j - \sum_{i=0}^{N}\alpha_i y_i \mathbf{x}_i \cdot \sum_{j=0}^{N}\alpha_j y_j \mathbf{x}_j + \sum_{i=0}^{N}\alpha_i. \tag{32}$$

3.2.2 Lösung für linear trennbare Muster

Die Gleichung (32) enthält nun die Bedingungen, die sich aus der Minimierung hinsichtlich $\mathbf{w}$ und b ergeben. Gesucht ist aber der Sattelpunkt auch in Bezug auf die einzelnen Lagrange-Multiplikatoren α_i. Hierbei galt es, alle Bedingungen (21) zu maximieren, so dass sich die Gesamtlösungsfunktion als die Maximierung von

$$Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{(i,j)=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j \tag{33}$$

unter den Bedingungen

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{34}$$

und

$$\alpha_i \geq 0 \;\; \forall \;\; i = 1 \ldots N \tag{35}$$

formulieren lässt (Haykin 1999, S. 322 f.). Für eine hier nun notwendige bedingte nichtlineare Optimierung stehen verschiedene Algorithmen zur Verfügung (Domschke und Drexl 2002; Grundmann 2002; Hillier und Liebermann 2002; Rardin 1998). Ihre Beschreibung ist nur im Rahmen eines weiteren umfangreichen Fachartikels möglich. Die Verfahren sind vom Ergebnis weitgehend gleichwertig, sie unterscheiden sich vielmehr im Rechenaufwand. Daher wird aktuell die Lösung über *sequentielle Minimumsoptimierung* (SMO) empfohlen (Platt 1998, 1999).

Ist die Lösungsfunktion $Q(\alpha)$ unter der Berücksichtigung von (34) und (35) maximiert, ergibt sich der optimale Gewichtsvektor

$$\mathbf{w}_{opt} = \sum_{i=1}^{N}\alpha_{opt,i} y_i \mathbf{x}_i. \tag{36}$$

An dieser Summation wirkt nur eine verhältnismäßig kleine Anzahl von Mustern mit, denn für die meisten Muster wird $\alpha_{opt,i}$ den Wert null angenommen haben. Damit reduziert sich die Summe auf

$$\mathbf{w}_{opt} = \sum_{\iota=1}^{N^{(s)}}\alpha_{opt,\iota} y_\iota^{(s)} \mathbf{x}_\iota^{(s)} \;\forall \alpha_{opt,\iota} > 0. \tag{37}$$

Diese $N^{(s)}$ Muster $[\mathbf{x}_\iota^{(s)}, y_\iota^{(s)}]$ nennt man die Stützvektoren oder *support vectors*. Sie bilden die Untermenge von Mustern, die jeweils am Rand ihrer Klasse die Lage und Ausrichtung der Hyperebene und damit auch die Lage und Ausrichtung des zugehörigen Trennbereiches bestimmen. Der zugehörige bestangepasste *bias* der Hyperebene errechnet sich aus einem beliebigen Stützvektor zu

$$b_{opt} = y_\iota^{(s)} - \mathbf{w}_{opt}^{\mathsf{T}}\mathbf{x}_\iota^{(s)} \tag{38}$$

analog mit

$$[\mathbf{x}_\iota^{(s)}, y_\iota^{(s)}] = [\mathbf{x}_i, y_i] \;\forall \alpha_i > 0. \tag{39}$$

3.3 SVM für linear nicht trennbare Muster

Es kann sich als notwendig erweisen, einige wenige falsche Klassifizierungen zuzulassen, nämlich dann, wenn die gegebenen Muster *a priori* linear untrennbar sind. Die Klassifikation entlang einer scharfen Hyperebene kann dahingehend erweitert werden, dass die Muster ausnahmsweise auch auf die falsche Seite der Hyperebene fallen dürfen. Zu diesen Zweck werden positive Schlupfvariablen – im Englischen *slack variables* genannt – als $\xi_i \geq 0$ eingeführt. Diese haben die Funktion von Fehlertermen. Damit erweitert sich der Klassifikator wie folgt:

$$y_i(\mathbf{w}^{\mathsf{T}}\mathbf{x}_i + b) \geq 1 - \xi_i \;\; \forall \;\; i = 1 \ldots N. \tag{40}$$

Die neu definierten Variablen ξ_i beschreiben das empirische Risiko einer Fehlklassifikation. Drei Fälle von Fehlklassifikationen sind möglich:

- $0 < \xi_i \leq 1$: Das Muster fällt in den Trennbereich auf Seiten der richtigen Klasse,
- $1 < \xi_i \leq 2$: Das Muster fällt in den Trennbereich auf Seiten der falschen Klasse,
- $\xi_i > 2$: Das Muster fällt in die falsche Klasse.

Die hierfür zu bildende Indikatorfunktion ist nichtkonvex, daher wird die zugehörige Verlustfunktion

$$\boldsymbol{\Phi}(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w} + C\sum_{i=1}^{N}\xi_i \tag{41}$$

um einen zweiten Term ergänzt, der eine Vereinfachung der nichtkonvexen Indikatorfunktion darstellt (Haykin 1999, S. 327).

Die Lösungsfunktion dieses Klassifikators (33) bleibt dieselbe. Jedoch führen die Ableitungen nach $\partial\mathbf{w}$, ∂b und $\partial\boldsymbol{\xi}$ zusammen mit den Karush-Kuhn-Tucker-Bedingungen (Burges 1998, S. 136 f.) anstelle der zweiten Lösungsbedingung (35) auf

$$0 \leq \alpha_i \leq C \;\; \forall \;\; i = 1 \ldots N. \tag{42}$$

Überraschenderweise tauchen die ξ_i in dieser Optimierung nicht mehr explizit auf. Vielmehr werden sie durch die Wahl von C definiert. Der Parameter C wirkt demnach als Stellgröße für den Klassifikator: Mit dem C wächst auch die Komplexität des Klassifikators. Dieser Parameter muss durch den Nutzer heuristisch angepasst werden. Indikatorgrößen zur Steuerung dieses Parameters sind im wesentlichen die Anzahl der als Stützvektoren erkannten Muster und die VC-Dimension des Modells (Kap. 6).

Die Berechnung von $\mathbf{w}_{opt}$ erfolgt analog zur Lösung für linear trennbare Muster (37). Der *bias* b_{opt} wird hingegen aus dem Mittelwert aus allen Stützvektoren (38) gebildet. Anschließend können alle positiven Schlupfvariablen

$$\xi_i = 1 - y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \quad \forall \xi_i > 0 \tag{43}$$

berechnet werden.

Ein abstrakter Wert wie C ist in der Nutzung nicht immer hilfreich. Daher kann ebenso ein positiver Maximalwert für ξ_i als Randbedingung eingeführt werden, wobei C dann als Parameter fungiert, für den ein optimaler Wert zu bestimmen ist.

4 Kernfunktionen

Es ist sehr aufwendig, große Mengen von Datenpaaren in einen höherdimensionalen Raum zu transformieren, um hier die Suche nach einer optimalen Hyperebene vorzunehmen. Daher wird, anstatt die Daten in den Merkmalsraum zu transformieren, jeweils eine Funktion gesucht, die einer Rückabbildung der Hyperebene aus dem Merkmalsraum in den Zustandsraum entspricht. Dieses Vorgehen wird – wie bereits erwähnt und beim Voronoi-Klassifikator bereits eingesetzt – als der *kernel trick*, also der Kunstgriff über die Kernfunktionen bezeichnet.

Eine solche stetige symmetrische Kernfunktion

$$\mathcal{K}(x_i, x_j) = \sum_{\iota=1}^{\infty} \lambda_\iota \varphi_\iota(\mathbf{x}_i)\varphi_\iota(\mathbf{x}_j) \tag{44}$$

mit den Eigenfunktionen $\varphi(\mathbf{x})$ und den positiven Eigenwerten λ sei definiert auf dem geschlossenen Intervall $\mathbf{a} \leq \mathbf{x}_i, \mathbf{x}_j \leq \mathbf{b}$. Soll eine Kernfunktion für die SVMs tauglich sein, also in der Lösungsfunktion $Q(\,\cdot\,)$ gleichmäßig und vollständig konvergieren, dann muss jene positiv definit sein. Positiv definit ist die Kernfunktion $\mathcal{K}(x_i, x_j)$ genau dann, wenn die Mercer-Bedingung[1]

$$\int_{\mathbf{a}}^{\mathbf{b}} \int_{\mathbf{a}}^{\mathbf{b}} \mathcal{K}(x_i, x_j)\psi(\mathbf{x}_i)\psi(\mathbf{x}_j)\partial\mathbf{x}_i\partial\mathbf{x}_j \geq 0 \tag{45}$$

erfüllt ist (Mercer 1909, S. 442). Die Funktionen $\psi(\mathbf{x})$ müssen dabei quadratintegrierbar sein:

$$\int_{\mathbf{a}}^{\mathbf{b}} \psi^2(\mathbf{x})\partial\mathbf{x} < \infty. \tag{46}$$

Mercers Bedingung beschreibt nicht etwa, wie Kernfunktionen gebildet werden können, sondern nur ob eine Funktion eine geeignete Kernfunktion sein kann (Haykin 1999, S. 332). Unter diesen Funktionen findet sich die Kernfunktion der Kreuzkorrelationsfunktion ebenso, wie auch die gewichtete Summe eines Neurons eines künstlichen neuronalen Netzes (KNN) oder die radiale Basisfunktion (RBF) über der euklidischen Norm.

Wir erinnern uns nochmals an die Lösungsfunktion (33) mit den Langrange-Multiplikatoren:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{(i,j)=1}^{N} \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle. \tag{47}$$

Eine Überführung der Muster in den Hyperraum führte gemäß der Transformationsanweisung (1) auf

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{(i,j)=1}^{N} \alpha_i\alpha_j y_i y_j \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j), \tag{48}$$

also die aufwendige Berechnungsvariante. Mit der Vorgabe, dass nun eine geeignete Kernfunktion

$$\mathcal{K}(x_i, x_j) = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j) \tag{49}$$

bereits im Zustandsraum berechnet werden kann, erfolgt die eigentliche Transformation lediglich implizit. Die Kernfunktion ist dann geeignet, wenn sie das Skalarprodukt $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ der Mustereingänge $\mathbf{x}_i$ enthält. Diese ist im Zustandsraum eben weit weniger rechenintensiv. Bereits das Skalarprodukt für sich allein fungiert $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ als eine Kernfunktion, so dass wir (33) schreiben dürfen als

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{(i,j)=1}^{N} \alpha_i\alpha_j y_i y_j \mathcal{K}(x_i, x_j), \tag{50}$$

jedoch mit der jetzt verallgemeinerten Kernfunktion $\mathcal{K}(x_i, x_j)$.

Der jetzt mehrfach erwähnte Skalarprodukt-Kern

$$\mathcal{K}(x_i, x_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i \cdot \mathbf{x}_j \tag{51}$$

eignet sich in erster Linie für linear trennbare Muster. Unter der Vorgabe, dass einige wenige – stochastisch bedingte – Fehlklassifikationen zugelassen werden dürfen, kann dieser Kern auch verwendet werden. Eine erste wesentliche Erweiterung stellt der Polynom-Kern

$$\mathcal{K}(x_i, x_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle)^d \tag{52}$$

bzw. der inhomogene Polynom-Kern

$$\mathcal{K}(x_i, x_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^d \tag{53}$$

1 Bei Haykin (1999, S. 331) sowie bei Haykin (2009, S. 311) finden sich vertauschte Integrationsgrenzen, vergleiche hierzu Mercer (1909, S. 416) und Schölkopf (2001, S. 36).

dar (Schölkopf und Smola 2001, S. 45 f.), der es beispielsweise in seiner quadratischen Form – also mit $d = 2$ – ermöglicht, das eingangs erläuterte XOR-Problem zu lösen (Haykin 1999, S. 355 ff.). Ein weitgehend gebräuchlicher Kern ist der Gauss- oder in erweiterter Form der RBF-Kern

$$\mathcal{K}(x_i, x_j) = e^{(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|)}. \tag{54}$$

Grundsätzlich eignen sich radiale Basisfunktionen in der Interpolation, in den Fuzzy Clustering Methoden oder als neuronale RBF-Netze für Modellierungsaufgaben. Dementsprechend sind sie auch als Kernfunktionen in den SVMs von größtem Interesse. Gleichfalls aus dem Bereich der neuronalen Netze kommt der Neuronale Kern

$$\mathcal{K}(x_i, x_j) = \tanh(a\langle \mathbf{x}_i, \mathbf{x}_j\rangle + b), \tag{55}$$

der anstelle eines Tangens Hyperbolicus auch andere sigmoidale Aktivierungsfunktionen verwenden kann. Der mit Abstand flexibelste Algorithmus ist der ANOVA-Kern

$$\mathcal{K}_D(x_i, x_j) = \sum_{1 \le i_1 < \ldots < i_D \le N} \left(\prod_{d=1}^{D} \mathcal{K}^{i_d}(\mathbf{x}_{i_d}, \mathbf{x}_{j_d}) \right). \tag{56}$$

Dieser Kern ermöglicht die mehrdimensionale Regression insbesondere unter der Voraussetzung, dass die κ verschiedenen Elemente des Eingangsvektors $(\mathbf{x}_i)_\kappa$ sehr unterschiedliche Verteilungen besitzen.

Allen dargestellten Kernfunktionen ist sichtbar zu eigen, dass sie um das Skalarprodukt $\langle \mathbf{x}_i, \mathbf{x}_j\rangle$ herum arrangiert sind. Eine Ausnahme bilden die RBF-Kerne. Sie bauen mit der euklidischen Norm auf dem Skalarprodukt der Musterdifferenz $\langle \mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_i - \mathbf{x}_j\rangle$ auf (Schölkopf und Smola 2001, S. 46).

5 Regression mit SVM

Im letzten Schritt kann das gewonnene Wissen aus der Mustererkennung auf die Regression

$$y_i = \mathrm{f}(\mathbf{x}_i) + \nu_i = \mathbf{w}^\mathsf{T}\mathbf{x}_i + b + \nu_i \tag{57}$$

übertragen werden. Dabei ist y_i eine Zufallsvariable aus einem deterministischen Anteil $f(\mathbf{x_i})$ und einem stochastischen Rauschen ν_i. Für die Regression wird die Bedeutung der Trennfläche umdefiniert: Alle Muster, die in die Trennfläche zwischen $-\varepsilon \ldots \varepsilon$ beiderseits der Hyperebene fallen, nehmen an der Optimierung nicht mehr teil (Abb. 6). Vielmehr erhalten nun die Muster außerhalb des Trennbereiches nichtnegative Schlupfvariablen. Daraus ergibt sich eine ε-unempfindliche Einflussfunktion Ψ_ε (Haykin 1999, Kap. 6.7). Dazu wird nochmals die Verlustfunktion (41) in der Form

$$\mathbf{\Phi}(\mathbf{w}, \xi, \xi^*) = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w} + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{58}$$

definiert, hier allerdings mit zwei Mengen nichtnegativer Schlupfvariablen ξ_i, ξ_i^*.

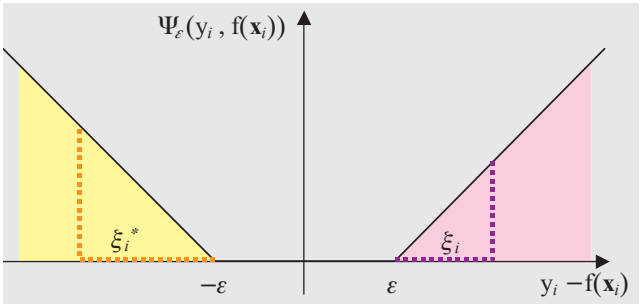

Abb. 6: Einflussfunktion der SV-Regression: Im 2ε breiten Trennbereich sind die Residuen gleich Null, außerhalb werden den Mustern positive und negative Schlupfvariablen zugewiesen, damit jene an der Optimierung teilnehmen.

Die Minimierungsaufgabe der so aus (58) definierten Verlustfunktion mit Hinblick auf

$$\mathbf{w}^\mathsf{T}\mathbf{x} + b \left\{ \begin{array}{l} \ge y_i - \varepsilon - \xi_i^* \\ \le y_i + \varepsilon + \xi_i \end{array} \right\} \quad \forall \;\; i = 1 \ldots N \tag{59}$$

unter der Voraussetzung

$$\xi_i, \xi_i^* \ge 0 \;\; \forall \;\; i = 1 \ldots N$$

kann gleichermaßen durch die Maximierung der Lösungsfunktion

$$\begin{aligned} Q(\alpha_i, \alpha_i^*) = &-\varepsilon \sum_{i=1}^{N}(\alpha_i + \alpha_i^*) \\ &+ \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \\ &- \frac{1}{2}\sum_{(i,j)=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\mathcal{K}(x_i, x_j) \end{aligned} \tag{60}$$

unter den Bedingungen

$$\sum_{i=1}^{N}\alpha_i = \sum_{i=1}^{N}\alpha_i^*, \tag{61}$$

und

$$0 \le \alpha_i, \alpha_i^* \le C \;\; \forall \;\; i = 1 \ldots N \tag{62}$$

gelöst werden. Während der Modellbildung müssen die Parameter ε, bzw. $+\xi_i$ und $-\xi_i^*$ im Einklang mit C derart angepasst werden, dass die Anzahl der Stützvektoren und die Komplexität des Modells, also die VC-Dimension (Kap. 6), möglichst klein gehalten werden.

Die Regressionslösung für beliebige Eingänge $\mathbf{x}$ ergibt sich als

$$\mathrm{f}(\mathbf{x}) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)\mathcal{K}(\mathbf{x}, \mathbf{x}_i) + b. \tag{63}$$

Die praktische Berechnung des *bias* b ist bei der SV-Regression ein eigenes Problem (Smola und Schölkopf 2004, Kap. 1.4) und kann nicht geschlossen gelöst werden.

6 Die Modellkapazität

Jedes Modell – so auch eine Regression – weist eine bestimmte Komplexität auf. Diese Komplexität muss in einem geeigneten Verhältnis zu der Dimension und der Anzahl der Muster stehen. Ein Missverhältnis als Resultat eines zu komplexen Modells führt zum gefürchteten Übertrainieren oder *overfitting* (Heine 1999; Miima 2002; Heinert und Niemeier 2007; Heinert 2008a,b). Dieses Missverhältnis setzt bereits lange vor einer zu niedrigen Redundanz im Sinne einer Ausgleichungsaufgabe ein (Niemeier 2008).

6.1 Die VC-Dimension

Da dieses Problem so schwerwiegend ist, muss zunächst ein eindeutiges numerisches Maß für die Komplexität eines Modells gefunden werden. Ein solches Maß ist beispielsweise die sogenannte Vapnik-Chervonenkis-Dimension oder abgekürzt VC-Dimension (Haykin 1999, S. 95).

> **Definition:** *Die VC-Dimension eines Ensembles von Dichotomien* $\mathcal{F} = \{\Phi_w(x) : w \in \mathbb{W}, \Phi : \mathbb{R}^m \mathbb{W} \rightarrow 0,1\}$ *ist die Kardinalität* $h = |\mathcal{L}|$ *der größten Untermenge* $\mathcal{L}$ *die von* $\mathcal{F}$ *zerschmettert wird.*

Diese Definition hat Vapnik selbst etwas erklärender abgefasst (Vapnik 1998, S. 147):

> *Die VC-Dimension einer Schar von Indikatorfunktionen* $\Phi_w(x), w \in \mathbb{W}$ *ist die größte Anzahl* h *von Vektoren, die in jeder der* 2^h *möglichen Verteilung unter Nutzung der Funktionsschar auf zwei unterschiedliche Klassen verteilt werden können.*

Die VC-Dimension ist also die maximale Anzahl $\mathcal{L}$ von Mustern in einem n-dimensionalen Ursprungsraum $\mathbb{R}^n$, die irrtumsfrei in zwei Klassen getrennt werden können (Haykin 1999, S. 94 f.). Da allerdings auch unscharfe Mengenzugehörigkeiten von Mustern oder unscharfe Zuweisungen durch die Funktionsschar möglich sind, erweitert sich $h = |\mathcal{L}| = \mathrm{VC}(\mathcal{N})$ zur *Kardinalität* der irrtumsfrei trennbaren Muster. Die *Mächtigkeit* oder *Kardinalität* einer endlichen Menge ist die Summe aller charakteristischen Funktionswerte ihrer Elemente (Bothe 1995, S. 35). Dabei beschreibt die charakteristische Funktion eines Elementes dessen unscharfe Zugehörigkeit zu einer Menge mit Werten von 0 bis 1. Diese Definition bezieht sich auf die Anwendung einer geeigneten Schar von w Indikatorfunktionen $\Phi_w(x)$, die dieses »Zerschmettern« der Gesamtmenge in zwei Klassen vornehmen. Die maximale Anzahl h muss dabei in allen 2^h möglichen Arten irrtumsfrei erfolgen können (siehe Abb. 7). Damit wird jeder Funktionsschar und weiterhin ihren verschiedenen Kombinationen eine *Kapazität* in Form der VC-Dimension zur Trennung zugeschrieben.

Ein theoretisches Gütekriterium für ein Modell muss die Aussage treffen können, ob die verwendeten Operatoren jeweils Funktionsscharen angehören, deren Anpassungsfähigkeit nicht zu groß gegenüber der Anzahl von Paaren von Eingangs- und Ausgangsvektoren $[\mathbf{x}_i, y_i]$ ist. Ist dieses Gütekriterium eingehalten und ist eine Anpassung des Modells an die Trainingsdaten gelungen, so ist es wahrscheinlich, dass das Modell die Struktur zwischen den Eingängen und Ausgängen eines Systems bestmöglich abbildet. Ist dieses Gütekriterium nicht erfüllt, besteht eine große Wahrscheinlichkeit, dass die Operatoren ihre Anpassungsfähigkeit zur ungeneralisierten Speicherung der Trainingsdaten verwendet haben.

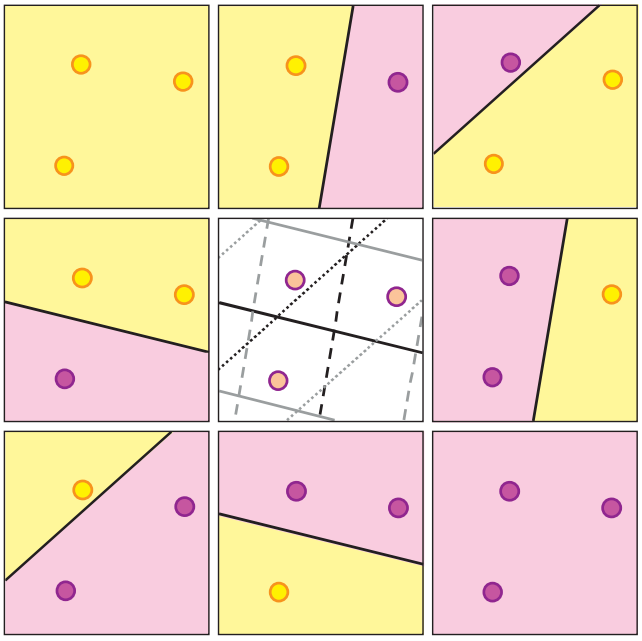

Abb. 7: Ein Beispiel für die VC dim$(\Phi(x)) = 3$. Die 2^3 Möglichkeiten der Verteilung der Muster kann durch $h = 3$ Linien irrtumsfrei in zwei Klassen geteilt werden.

Die Berechnung dieser Dimension kann bisher noch nicht verallgemeinert werden (Koiran und Sontag 1996; Elisseeff und Paugam-Moisy 1997; Sontag 1998; Schmitt 2001, 2005). Vielmehr ist es oft nur möglich, die oberen und unteren Grenzen in Form von Landau-Bachmann-Symbolen in Abhängigkeit von der Anzahl der Modellkerne anzugeben.

Im Regressionsfall ist die VC-Dimension die Kardinalität der Muster, die durch eine Funktionsschar irrtumsfrei approximiert werden.

6.2 Verteilungsunabhängige Grenzen der Risikofunktion

Im besonderen Fall der Regression ist die Ausgangssituation gegeben, dass eine Reihe von Zufallswerten mit einer Funktion beliebiger Komplexität beschrieben werden kann. Wo werden noch funktionale Anteile berücksichtigt, wo werden stochastische Anteile in die Regressionsfunktion aufgenommen? Man sucht demnach einen Gleichgewichtszustand zwischen einer komplexen Regressionsfunktion und dem empirischen Risiko. Um die-

ses Gleichgewicht finden zu können, benötigt man eine Beschreibung des vollständigen Risikos: Die Grenze der Risikofunktion oder Risikoschranke $R(h)$ (Abb. 8) berücksichtigt durch die Summe

$$R(h) = R_{\text{emp}} + \epsilon_1(N, h, \alpha, R_{\text{emp}}) \tag{64}$$

sowohl das empirische Risiko R_{emp} als auch das Konfidenzintervall

$$\epsilon_1(N, h, \alpha, R_{\text{emp}}) = 4\epsilon_0^2(N, h, \alpha) \tag{65}$$

in Abhängigkeit des empirischen Risikos und des Konfidenzintervalls der VC-Dimension (Haykin 1999, S. 99 ff.). Dieses Konfidenzintervall der VC-Dimension (Vapnik 1998, S. 219 ff.)

$$\epsilon_0(N, h, \alpha) = \sqrt{\frac{h}{N}\left(\log\left(\frac{2N}{h} + 1\right) - \frac{1}{N}\log\alpha\right)} \tag{66}$$

ist eine Komplexitätsschranke abhängig von der Anzahl der Muster N, der VC-Dimension h und der Wahrscheinlichkeit

$$\alpha = \left(\frac{2eN}{h}\right)^h e^{-\eta^2 N}. \tag{67}$$

Darin ist η die Genauigkeit, mit der die Approximation erfolgen soll.

Entscheidend für die Komplexität eines Modells ist der Bereich um das Minimum der Risikoschranke (Abb. 8). In diesem Bereich ist ein Modell mit den entsprechenden VC-Dimensionen richtig bestimmt. Ist die VC-Dimension kleiner, besteht Grund zur Annahme, dass das Modell *überbestimmt* ist, also zuviele Muster für ein simples Modell

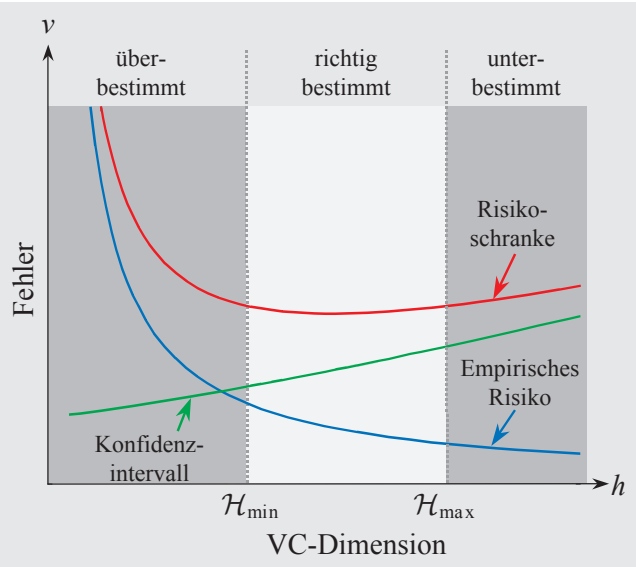

Abb. 8: Die Risikoschranke als obere Hüllende über dem empirischen Risiko und dem Konfidenzintervall als Maß der Modellkapazität hat ihr Minimum im Bereich der optimalen VC-Dimension. Überbestimmte Modelle generalisieren möglicherweise zu stark, während unterbestimmte Modelle zu stark memorisieren.

aufweist (Haykin 1999, S. 100 f.). Es ist damit aber keineswegs ausgeschlossen, dass das Modell die Muster so »intelligent« generalisiert hat, dass das Modell dennoch ein kleines empirisches Risiko hat und damit durchaus identifizierend ist.

Umgekehrt ist ein Modell mit hoher VC-Dimension *unterbestimmt*. Es gibt also möglicherweise zu wenige Muster um das Modell zu stützen. Bei einer nur schwachen Unterbestimmung kommt es zunächst zu dem Phänomen, dass sich die Abbildung von Subprozessen additiv auf mehr Modellkerne verteilt, als zur Abbildung notwendig wären. Im schlimmsten Fall wird das Modell seine Kapazität darauf verwenden, anstatt eine Generalisierung zu erreichen, jedes einzelne Muster zu memorisieren, was man als *Übertrainieren* bezeichnet. Es wird sich also im Extremfall an jedes einzelne Muster »erinnern«. Das Minimum der Risikoschranke liegt also im Optimum zwischen »Wissen« und »Intelligenz« des Modells.

7 Resümee

Wie genau realisiert eine SVM die Robustheit gegen das Problem des Übertrainierens? Obgleich gerade die Nutzung der mächtigen Kernfunktionen, wie beispielsweise der RBF- und ANOVA-Kern, theoretisch auf unendliche VC-Dimensionen führt, soll der Algorithmus robust sein?

Der entscheidende Schritt hierfür ist bereits getan: Wie immer auch die Hyperebene in den Mustern liegt, ob sie denn zur Mustererkennung oder zur Regression im höherdimensionalen Merkmalsraum zu liegen kommt, die Hyperebene verfügt bei richtiger Verwendung der SVM eine endliche und im günstigsten Fall eine signifikant kleinere Anzahl an Stützvektoren als die Anzahl der verfügbaren Muster. Denn man erinnere sich, dass die Hyperebene in jedem höherdimensionalen Raum und sowohl im linearen als auch im nichtlinearen Fall sich auf eine Untergruppe von Mustern stützt, nämlich auf möglichst wenige Stützvektoren. Alle anderen Muster haben keinen Einfluss auf die Lage der Hyperebene. Sie tragen also nicht zum Lernprozess bei, obgleich sie implizit mitgelernt sind. Damit ist ein Übertrainieren *a priori* ausgeschlossen.

Gleichwohl ist während der Nutzung einer SVM fortwährend darauf zu achten, dass die Wahl der Parameter, insbesondere C und ε, zu Lösungen mit niedrigen VC-Dimensionen führen.

Im zweiten Teil dieses Artikels in einem Folgeheft wird auf verschiedene Berechnungsbeispiele und Anwendungen eingegangen werden. Hier wird sich zeigen, dass die aufwendigen theoretischen Herleitungen und die zunächst ungewohnten mathematischen Betrachtungen zu interessanten praktischen Ergebnissen führen.

Literatur

Akyilmaz, O. und Kutterer, H.: Prediction of Earth rotation parameters by fuzzy inference systems. J. Geodesy, 78 (1-2), S. 82–93, 2004.

Bothe, H.-H.: Fuzzy Logic – Einführung in Theorie und Anwendung. 2. erw. Aufl., Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong, Barcelona, Budapest, 1995.

Bothe, H.-H.: Neuro-Fuzzy-Methoden – Einführung in Theorie und Anwendung. Springer-Verlag, Berlin, Heidelberg, New York, Barcelona, Budapest, Hong Kong, London, Mailand, Paris, Santa Clara, Singapur, Tokyo, 1998.

Burges, C.H.C.: A Tutorial on Support Vector Machines for Pattern Recognition. In: Data Mining and Knowledge Discovery. vol. 2, S. 121–167, Kluwer Academic Publishers, 1998.

Chang, Ch.-Ch. und Lin, Ch.-J.: LIBSVM: a library for support vector machines, 2001. Software, URL: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Domschke, W. und Drexl, A.: Einführung in Operations Research. 5. überarb. u. erw. Aufl., Springer Berlin/Heidelberg, 2002.

Elisseeff, A. und Paugam-Moisy, H.: Size of multilayer networks for exact learning: analytic approach. NeuroCOLT Techn. Rep. Series, NC-TR-97-002, 1997.

Grundmann, W.: Operations Research Formeln und Methoden. Teubner, Stuttgart, Leipzig, Wiesbaden, 2002.

Haykin, S.: Neural Networks – A Comprehensive Foundation. 2nd ed., Prentice Hall, Upper Saddle River NJ, 1999.

Haykin, S.: Neural Networks and Learning Machines: A Comprehensive Foundation. 3nd ed., Prentice Hall, Upper Saddle River NJ, 2009.

Heine, K.: Beschreibung von Deformationsprozessen durch Volterra- und Fuzzy-Modelle sowie Neuronale Netze. Diss., DGK Reihe C, 516, München, 1999.

Heine, K.: Fuzzy Technology and ANN for Analysis of Deformation Processes. In: Reiterer, A. und Egly, U. (Hrsg.): Application of Artificial Intelligence in Engineering Geodesy (AIEG 2008), S. 9–25, Wien, 2008.

Heinert, M. und Niemeier, W.: From fully automated observations to a neural network model inference: The Bridge "Fallersleben Gate" in Brunswick, Germany, 1999–2006. J. Appl. Geodesy, 1, S. 71–80, 2007.

Heinert, M.: Systemanalyse der seismisch bedingten Kinematik Islands. Diss., Geod. Schriftenr. 22, Techn. Univ. Braunschweig, 2008.

Heinert, M.: Artificial neural networks – how to open the black boxes? In: Reiterer, A. und Egly, U. (Hrsg.): Application of Artificial Intelligence in Engineering Geodesy (AIEG 2008), S. 42–62, Wien, 2008.

Heinert, M.: Modellierung von gemessenen Zeitreihen für Monitoringaufgaben. In: Foppe, K., Knaak, L. u. Pawel, B. (Hrsg.): Zeitabhängige Messgrößen – Verborgene Schätze in unseren Daten. DVW-Schriftenr., 59, S. 133–154, Augsburg, 2009.

Hillier, F.S. und Liebermann, G.J.: Operations Research Einführung. 5. Aufl., unveränd. Nachdr. d. 4. Aufl., Oldenbourg Verlag, München, Wien, 2002.

Koiran, P. und Sontag, E.D.: Neural Networks with Quadratic VC-Dimension. Advances in Neural Information Systems, 8:197–203, MIT Press, Cambridge (MA), 1996.

Kuhn, H.W. und Tucker, A.W.: Nonlinear Programming. Proc. of 2nd Berkeley Symp., S. 481–492, Univ. of California Press, Berkeley, 1951.

Mercer, J.: Functions of Positive and Negative Type and their Connection with the Theory of Integral Equations. Phil. Trans. R. Soc. Lond. A, 209, S. 415–446, 1909.

Miima, J.B.: Artificial Neural Networks and Fuzzy Logic Techniques for the Reconstruction of Structural Deformations. Diss., Geod. Schriftenr., 18, Techn. Univ. Braunschweig, 2002.

Minkler, G. und Minkler, J.: Theory and Application of Kalman Filtering. Magellan Book Comp., Palm Bay, U.S.A, 1993.

Niemeier, W.: Ausgleichsrechnung – Eine Einführung für Studierende und Praktiker des Vermessungs- und Geoinformationswesens. 2. überarb. u. erw. Ausg., Walter de Gruyter, Berlin, New York, 2008.

Platt, J.C.: Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Microsoft Research, Technical Report MSR-TR-98-14.

Platt, J.C.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: Schölkopf, B., Burges, C.J.C. and Smola, A.J. (1999): Advances in kernel methods: support vector learning. MIT Press, Cambridge (MA), S. 185–208.

Rardin, R.L. (1998): Optimization in Operation Research. Prentice Hall, Upper Saddle River USA, 1998.

Reiterer A.: Ein wissensbasiertes Entscheidungssystem für ein online Videotheodolit-basiertes Messsystem. zfv, 130 (4), S. 218–225, 2005.

Reiterer A.: Künstliche Intelligenz in der Ingenieurgeodäsie. zfv, 131 (3), S. 141–147, 2006.

Riedel, B. und Heinert, M.: An adapted support vector machine for velocity field interpolation at the Baota landslide. In: Reiterer, A. und Egly, U. (Hrsg.): Application of Artificial Intelligence in Engineering Geodesy (AIEG 2008), S. 101–116, Wien, 2008.

Rüping, St.: Zeitreihenanalyse für Warenwirtschaftssysteme unter Berücksichtigung asymmetrischer Kostenfunktionen. URL: http://www.stefan-rueping.de/publications/rueping-99-a.pdf.

Rüping, St.: mySVM – Manual. URL: http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/mysvm-manual.pdf.

Schmitt, M.: Radial basis function neural networks have superlinear VC dimension. In: Helmbold, D. and Williamson, B. (Eds.): Proceedings of the 14th Annual Conference on Computational Learning Theory COLT 2001 and 5th European Conference on Computational Learning Theory EuroCOLT 2001, Lecture Notes in Artificial Intelligence, 2111, S. 14–30, Springer-Verlag, Berlin, 2001.

Schmitt, M.: On the capabilities of higher-order neurons: A radial basis function approach. Neural Computation, 17 (3), S. 715–729, 2005.

Schölkopf, B. und Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning). MIT Press, Cambridge (MA), 2001.

Schölkopf, B., Cristianini, N., Jordan, M., Shawe-Taylor, J., Smola, A.J., Vapnik, V.N., Wahba, G., Williams, Chr. und Williamson, B.: KernelMachines.Org, 2007. URL: http://www.kernel-machines.org.

Smola, A.J. und Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing 14, S. 199–222, 2004.

Sontag, E.D.: VC Dimension of Neural Networks. In: Bishop, C. (Ed.): Neural networks and machine learning. S. 69–95, Springer-Verlag, Berlin, 1998.

Vapnik, V.N.: Statistical Learning Theory. In: Haykin, S. (Ed.): Adaptive and Learning Systems for Signal Processing, Communications and Control. John Wiley & Sons, New York, Chichester, Weinheim, Brisbane, Singapore, Toronto, 1998.

Vapnik, V.N. und Chervonenkis, A.Ya.: Theory of Pattern Recognition. Nauka, Moskau, Deut. Übers.: Wapnik, W.N. und Tscherwonenkis, A.Ja.: Theorie der Zeichenerkennung, in deutscher Spr. v. Unger, S. und Fritzsch, K. In: Frühauf, H., Kämmerer, W., Thiele, H. und Völz, H. (Hrsg.): Elektronisches Rechnen und Regeln. Sonder-Bd. 28, Akademie-Verlag, Berlin, 1979.

Anschrift des Autors

Dr.-Ing. Michael Heinert
Institut für Geodäsie und Photogrammetrie
Technische Universität Braunschweig
Gaußstraße 22, 38106 Braunschweig
Tel.: 0531 391-7494
Fax: 0531 391-7499
m.heinert@tu-bs.de