# Support-Vektor-Lernen

**Bernhard Schölkopf**

GMD FIRST
Rudower Chaussee 5
D-12489 Berlin
email: bs@first.gmd.de
www: http://www.first.gmd.de/∼bs

Dieser Beitrag erläutert die Grundgedanken der statistischen Lerntheorie und fasst die wesentlichen Ergebnisse der Arbeit [4] zusammen. Dies geschieht in folgenden Abschnitten:

1. Ein mathematisches Beispiel für das *Lernen aus Beispielen* wird vorgestellt.
2. Es wird illustriert, dass neben dem Erklären der Trainingdaten die *Komplexität von Lernmaschinen* wesentlich für den Lernerfolg ist.
3. *Algorithmen in Merkmalsräumen* stellen eine elegante und effiziente Methode dar, verschiedene Lernmaschinen mit kontrollierbarer Komplexität durch Kernfunktionen zu realisieren.
4. *Support-Vektor-Maschinen* verwenden Kernfunktionen zur Schätzung von Funktionen.
5. *Kern-PCA* verwendet Kernfunktionen zur Extraktion von nichtlinearen Merkmalen aus Datensätzen.
6. Jeder Algorithmus, der sich anhand von Skalarprodukten formulieren lässt, kann durch Verwendung von Kernfunktionen nichtlinear gemacht werden.

# 1 Lernen aus Beispielen

Die statistische Lerntheorie [13] beschäftigt sich mit dem Lernen aus empirischen Daten anhand mathematisch fundierter Algorithmen und Methoden. Sie vertritt einen spezifischen Zugang zum Grundproblem der Erkennung von Regularitäten in einer komplexen Umwelt, den man folgendermaßen vereinfacht modellieren kann: auf Basis von *Beobachtungen* (oder *Trainingsbeispielen*) zweier Größen $\mathbf{x}$ und $y$,

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell) \in \mathbb{R}^N \times Y, \tag{1}$$

versucht man den Zusammenhang der Größen zu modellieren, um bei der Beobachtung eines neuen *Musters* $\mathbf{x}^*$ eine gute Prognose $y^*$ abgeben zu können. Bevor wir dieses Modell weiter präzisieren, sollen ein paar Beispiele gegeben werden, die sich in das Paradigma einfügen:

- *Mustererkennung:* angenommen, wir wollen zwei Klassen von Bilder, die in einer Auflösung $16 \times 16$ vorliegen, unterscheiden. Im einfachsten Fall repräsentieren wir die Bilder als Vektoren, d.h. wir wählen $N = 256$ und $Y = \{\pm 1\}$.

- *Eindimensionale Regression:* zur Schätzung eines Zusammenhanges zweier kontinuierlicher Größen $\mathbf{x}, y \in \mathbb{R}$ wählen wir $N = 1$ und $Y = \mathbb{R}$.

- *Zeitreihenprognose:* oft ist bei geeigneter Wahl von $N \in \mathbb{N}, \tau > 0$ ist der zeitliche Verlauf einer Größe $z$ zum Zeitpunkt $t$ durch die Werte zu den Zeitpunkten $t-\tau, t-2\tau, \ldots, t-N\tau$ bestimmt. In diesem Fall verwenden wir $\mathbf{x} = (z(t-\tau), \ldots, z(t-N\tau)) \in \mathbb{R}^N$ und $y = z(t) \in \mathbb{R}$.

Ganz allgemein lässt sich sagen, dass dieser *statistische* Zugang zum Schätzen von Regularitäten sich dadurch auszeichnet, dass er insbesondere auf Probleme angewandt werden kann, wo die explizite Modellierung eines Zusammenhangs schwierig oder unmöglich ist, dafür aber empirische Beobachtungen vorliegen (vgl. *data mining*). Ein Beispiel hierfür ist die Erkennung handgeschriebener Ziffern. Wie eine Ziffer im Detail geschrieben wird, hängt von einer Reihe schwer modellierbarer Faktoren ab, und so bietet es sich an, die Klassifikation aus Beispielen zu *lernen.*
Zur Verschärfung (bzw. Vereinfachung) der mathematische Modellbildung machen wir folgende Annahmen:

- Die Trainingsdaten sind gemäß einer unbekannten Wahrscheinlichkeitsverteilung $P(\mathbf{x}, y)$ erzeugt worden. Dies schließt den Fall eines deterministischen Zusammenhanges zwischen $\mathbf{x}$ und $y$ ein.

- Der Zusammenhang zwischen $\mathbf{x}$ und $y$ soll in Form einer Funktion $f : \mathbb{R}^N \to Y$ geschätzt werden. Dies bedeutet, dass wir nicht nur für *ein* Testmuster $\mathbf{x}^*$, sondern simultan für alle möglichen Eingaben $\mathbf{x} \in \mathbb{R}^N$ eine Prognose $y$ abgeben.

- Die beste Schätzung ist die, die das *Risiko*

$$R[f] = \int |f(\mathbf{x}) - y|^c \, dP(\mathbf{x}, y) \tag{2}$$

(z.B. für $c \in \{1, 2\}$), also den erwarteten Fehler, minimiert.

In der statistischen Lerntheorie werden Lernmaschinen auf Eigenschaften untersucht, die dafür verantwortlich sind, dass von endlichen Trainingsmengen auf potentiell unendlich viele verschiedene Testmuster *generalisiert* werden kann *(Induktion)*, d.h. ein geringes Risiko erreicht wird.

# 2 Komplexität von Lernmaschinen

Nun stellt sich die Frage, wie man aus den Trainingsbeispielen eine Funktion konstruieren kann, die ein geringes Risiko aufweist. Die wesentliche Einsicht, die im folgenden vermittelt werden soll, lautet: es genügt nicht, eine Funktion zu finden, die ein geringes *empirisches Risiko*

$$R_{emp}[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} |f(\mathbf{x}_i) - y_i|, \tag{3}$$

also einen niedrigen *Trainingsfehler*, erreicht — genauso wichtig für die Generalisierung auf neue Daten ist die *Komplexität* der *Lernmaschine*, also eine Eigenschaft der *Klasse von Funktionen*, aus denen der Lernprozess seine Schätzung auswählt.
Dies lässt sich schon anhand eines einfachen Beispiels aus der Mustererkennung verdeutlichen. Wir betrachten eine beliebige Klassifikationsfunktion $f : \mathbb{R}^N \to \{\pm 1\}$ und eine Testmenge

$$(\bar{\mathbf{x}}_1, \bar{y}_1), \ldots, (\bar{\mathbf{x}}_m, \bar{y}_m) \in \mathbb{R}^N \times \{\pm 1\} \tag{4}$$

mit der Eigenschaft $\{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m\} \cap \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\} = \{\}$ (d.h. alle Testmuster sind tatsächlich neu, vgl. (1)). Da die Funktionswerte auf den beiden disjunkten Mengen von Mustern völlig unabhängig sind, können wir offensichtlich immer eine zweite Funktion $f^*$ angeben mit der Eigenschaft dass $f^*(\mathbf{x}_i) = f(\mathbf{x}_i)$ für $i = 1, \ldots, \ell$, und nichtsdestotrotz $f^*(\bar{\mathbf{x}}_i) \neq f(\bar{\mathbf{x}}_i)$ für $i = 1, \ldots, m$. Auf Basis

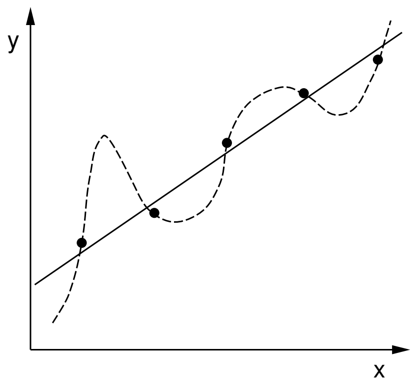


Abbildung 1: Aus einer gegebenen Menge von Beispielen (im Bild: schwarze Punkte) soll ein unbekannter funktionaler Zusammenhang geschätzt werden. Die statistische Lerntheorie formalisiert, in welchem Sinne die Gerade als Schätzung vorzuziehen ist (siehe Text).

der Trainingsdaten allein kann nicht entschieden werden, welche der beiden Funktionen besser ist, und auf den Testdaten machen sie völlig unterschiedliche Prognosen. In diesem Szenario ist Lernen unmöglich. Woran liegt dies? Verantwortlich ist die implizite Annahme, dass der Lernprozess aus der Menge *aller* Funktionen auswählen darf. Zum erfolgreichen Lernen muss die Menge der Funktionen, die eine Lernmaschine realisieren kann, eingeschränkt werden. Auf welche Art?

Abbildung 1 zeigt zwei Funktionen als Kandidaten für Schätzungen aus einem Datensatz. Zum einen zeigt dieses Beispiel nochmals den soeben vorgetragenen Punkt: es genügt nicht, die Trainingsdaten gut zu "erklären." Zum anderen deutet es an, welche Eigenschaft der Lernmaschine hinzukommen muss. Auf Basis der Trainingdaten würde die statistische Lerntheorie dem linearen Zusammenhang zwischen $\mathbf{x}$ und $y$ den Vorzug geben, da er *einfacher* als die komplizierte Funktion ist. In anderen Worten: schafft man es, mit einer Funktionenklasse von *geringer* Komplexität auf einer Menge von Trainingdaten ein geringes empirisches Risiko zu erreichen, so liegt dies mit hoher Wahrscheinlichkeit daran, dass die tatsächlich zugrundeliegende Regularität gefunden worden ist. Bei einer Funktionenklasse *hoher* Komplexität ist das Erreichen eines geringen empirischen Risikos hingegen nicht erstaunlich: lässt man Polynome vierter Ordnung zu, so kann man in dem abgebildeten Beispiel *immer* das empirische Risiko zu null machen, egal wie die $y$-Werte der Trainingspunkte gewählt werden.

Die Formalisierung dieses intuitiven Komplexitätsbegriffs, beispielsweise durch den Begriff der *VC-Dimension*, bildet den Kern der statistischen Lerntheorie.

Anhand der VC-Dimension, oder verwandter Begriffe, lassen sich probabilistische Schranken für das Risiko (2) angeben. Für den Fall der Mustererkennung gilt beispielsweise für alle $\eta > 0$ mit einer Wahrscheinlichkeit von mindestens $1 - \eta$ für $\ell > h$ die Ungleichung

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h\left(\log\frac{2\ell}{h} + 1\right) - \log(\eta/4)}{\ell}}. \tag{5}$$

Hierbei ist $h$ die VC-Dimension der zugrundeliegenden Funktionenklasse.[1]
Leider kann die statistische Lerntheorie hier nicht im Detail beschrieben werden (siehe z.B. [13]). Wir beschränken uns diesbezüglich auf die Aussage, dass sich in der Regel die Komplexität nur bei relativ einfachen Lernmaschinen präzise auswerten lässt. Einfache Lernmaschinen sind jedoch komplexen Datensätzen oft nicht gewachsen, und so ist in den letzten zehn Jahren, vor allem im Bereich der *Neuronalen Netze*, eine Vielzahl von Algorithmen entwickelt worden, die sehr gut funktionieren, sich aber theoretisch nur schwer analysieren lassen und daher nur begrenzt Einsichten in das Grundproblem des Lernens aus Beispielen vermitteln.
*Kern-Algorithmen*, die in diesem Beitrag näher beschrieben werden sollen, haben den Vorzug, beides zu verbinden: zum einen verwenden sie hinreichend reichhaltige Funktionenklassen, um auf realistische Lernprobleme anwendbar zu sein. Zum anderen ist ihre Komplexität beschränkt, da sie mathematisch äquivalent zu einer sehr einfachen Klasse von Funktionen in Merkmalsräumen sind.

# 3 Algorithmen in Merkmalsräumen

Die Grundidee von Algorithmen in Merkmalsräumen ist folgende: über eine nichtlineare Abbildung

$$\begin{aligned} \Phi : \mathbb{R}^N &\rightarrow F \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) \end{aligned} \tag{6}$$

werden die Trainingsdaten (1) in einen hochdimensionalen Raum $F$ transformiert. Das Lernproblem wird sodann für

$$(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_\ell), y_\ell) \in F \times Y \tag{7}$$

[1]Die Existenz von Schranken wie (5) impliziert nicht, dass jede beliebige, z.B. in der Natur vorkommende Regularität *gelernt* werden kann. Es kann durchaus vorkommen, dass die VC-Dimension, die man benötigt, um $R_{emp}$ klein zu machen, so groß ist, dass der zweite Term auf der rechten Seite zu groß wird. Die Schranke macht lediglich eine methodologische Aussage: wenn man mit geringer VC-Dimension die Daten erklären kann, dann hat man mit hoher Wahrscheinlichkeit richtig gelernt.

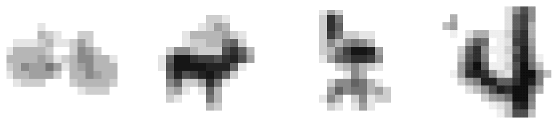

Abbildung 2: Beispielbilder aus verschiedenen Mustererkennungsdatenbanken

behandelt. Im Rahmen der klassischen Statistik erscheint dies unsinnig: nach dem sogenannten *curse of dimensionality* wächst die Schwierigkeit eines Schätzproblems in der Regel mit der Dimensionalität des Raumes. Die statistische Lerntheorie zeigt jedoch, dass dies auf Kern-Algorithmen so nicht zutrifft. Im Gegenteil: in gewissem Sinne ist das Lernen in $F$ einfacher, da dort mit einer sehr einfachen Klasse von Algorithmen gearbeitet werden kann — die notwendige Reichhaltigkeit steckt in der Abbildung $\Phi$.
Eine in der Mustererkennung besonders verbreitete Klasse von Merkmalen [10] ist die der Monome. Für das eingangs besprochene Beispiel der $16 \times 16$ Bilder (Abbildung 2) könnte man also eine Abbildung $\Phi$ definieren, die jedes Bild in einen Vektor abbildet, der alle Produkte von 5 der 256 Pixel als Komponenten enthält. Dies führt jedoch zu einem $10^{10}$-dimensionalen Raum. Selbst wenn man die statistische Komplexität des nachfolgenden Algorithmus kontrollieren kann, so bleibt immer noch das Problem, in diesem hochdimensionalen Raum überhaupt einen Algorithmus auszuführen.
An dieser Stelle kommt eine sehr wirkungsvolle Methode zum Berechnen von Skalarprodukten mittels Kernfunktionen ins Spiel. Beginnen wir mit einem einfachen Beispiel [13]. Sei

$$\begin{aligned} \Phi : \mathbb{R}^2 &\rightarrow F \\ (x_1, x_2) &\mapsto (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2), \end{aligned} \tag{8}$$

dann findet man

$$\begin{aligned} (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) &= (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)(y_1^2, \sqrt{2}\, y_1 y_2, y_2^2)^\top \\ &= ((x_1, x_2)(y_1, y_2)^\top)^2 \\ &= (\mathbf{x} \cdot \mathbf{y})^2 \\ &=: k(\mathbf{x}, \mathbf{y}). \end{aligned} \tag{9}$$

Dieses einfache Beispiel lässt sich verallgemeinern:

- für $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$, und $N, d \in \mathbb{N}$, berechnet

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d \tag{10}$$

ein Skalarprodukt im Raum aller Produkte von $d$ Einträgen der Vektoren $\mathbf{x}, \mathbf{y}$ [13, 8].

- ist $k : \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}$ stetiger Kern eines positiven Integraloperators auf $L_2(\mathcal{C})$ (für eine kompakte Menge $\mathcal{C} \subset \mathbb{R}^N$), d.h.

$$\forall f \in L_2(\mathcal{C}) : \quad \int_{\mathcal{C}} k(\mathbf{x}, \mathbf{y}) f(\mathbf{x}) f(\mathbf{y})\, d\mathbf{x}\, d\mathbf{y} \geq 0, \tag{11}$$

so existiert ein Merkmalsraum $F$ und eine Abbildung $\Phi : \mathbb{R}^N \rightarrow F$ sodass $k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$ [13]. Dies sieht man direkt, wenn man den Integralkern als

$$k(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{N_F} \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{y}) \tag{12}$$

in seinen Eigenfunktionen $\psi_j$ entwickelt ($\lambda_j > 0$, $N_F \leq \infty$). In diesem Fall ist nämlich

$$\Phi(\mathbf{x}) = (\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \ldots) \tag{13}$$

eine mögliche Realisierung von $\Phi$.

Man kann also das Skalarprodukt in $F$ berechnen, ohne explizit die Abbildung nach $F$ zu berechnen. *Jeder Algorithmus, der nur Skalarprodukte verwendet, kann so implizit in $F$ ausgeführt werden.*[2]
In den folgenden Kapiteln besprechen wir zwei Beispiele solcher Algorithmen. Beide sind *einfach* im Merkmalsraum, erzeugen aber trotzdem reichhaltige Funktionenklassen, und beide lassen sich so formulieren, dass sie nur von Skalarprodukten abhängen.

# 4 Support-Vektor-Maschinen

## 4.1 Klassifikation

Support-Vektor(SV)-Maschinen zur Mustererkennung separieren zwei gegebene Klassen von Trainingsdaten in einem Merkmalsraum beliebiger Dimensio-

[2] Auch Algorithmen, die auf *Ähnlichkeitsmaßen* $k$ operieren, die positive Matrizen $k(\mathbf{x}_i, \mathbf{x}_i)_{ij}$ erzeugen, können als lineare Algorithmen in $F$ interpretiert werden [4]; vgl. T. Hofmanns Beitrag.

nalität durch eine Hyperebene minimaler VC-Dimension [13]. Im Eingaberaum haben SV-Entscheidungsfunktionen die Form

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{\ell} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b\right). \tag{14}$$

Um $b \in \mathbb{R}$ und $\alpha_1, \ldots, \alpha_\ell \geq 0$ zu bestimmen, und somit die Hyperebene in $F$ zu konstruieren, muss folgendes quadratische Optimierungsproblem mit Nebenbedingungen gelöst werden:

$$\begin{array}{ll} \text{maximiere} & W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{unter den} & \\ \text{Nebenbedingungen} & \sum_{i=1}^{\ell} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell. \end{array} \tag{15}$$

Die Konstante $C > 0$ spielt die Rolle eines Regularisierungsparameters, der den trade-off zwischen der Minimierung von $R_{emp}$ und der Minimierung des Komplexitätsterms bestimmt (vgl. (5)). Die SV-Entscheidungsfunktion ist durch (15) eindeutig bestimmt.

Empirisch findet man, dass wenige Beispiele des gesamten Datensatzes, die *Support-Vektoren*, die Hyperebene definieren — für alle anderen Daten $\mathbf{x}_i$ gilt $\alpha_i = 0$, sie sind für das Mustererkennungsproblem irrelevant [3]. Durch die Wahl verschiedener Kernfunktionen, und dadurch verschiedener Merkmalsräume, ergeben sich als Spezialfälle Neuronale Netze,

$$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \Theta), \tag{16}$$

Gaußsche Radialbasisfunktionen(RBF)-Netze (vgl. Abbildung 3)

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\,\sigma^2)\right), \tag{17}$$

und Polynomialklassifikatoren (10).

Bemerkenswerterweise erhält man auf einer Reihe von Klassifikationsproblemen bei Benutzung dieser unterschiedlichen Kerne, bzw. Klassifikatorarchitekturen, sehr ähnliche Mengen von Support-Vektoren. In diesem Sinne charakterisieren jene das gegebene Klassifikationsproblem [3, 4].

In Anwendungen der Mustererkennung, Regression und Zeitreihenprognose wurde mit SV-Maschinen eine ganze Reihe von Rekordergebnissen erzielt [2, 4, 9], insbesondere durch Verwendung von Methoden zur Einbeziehung von *a-priori-Wissen*. Dies bezeichnet Information über die zu lernende Funktion, die zusätzlich zur Trainingsmenge vorhanden ist. In der Ziffern- und Objekterkennung gibt es beispielsweise Invarianztransformationen, wie die räumliche Verschiebung von Mustern, die Klassenzugehörigkeiten nicht verändern. Dies kann

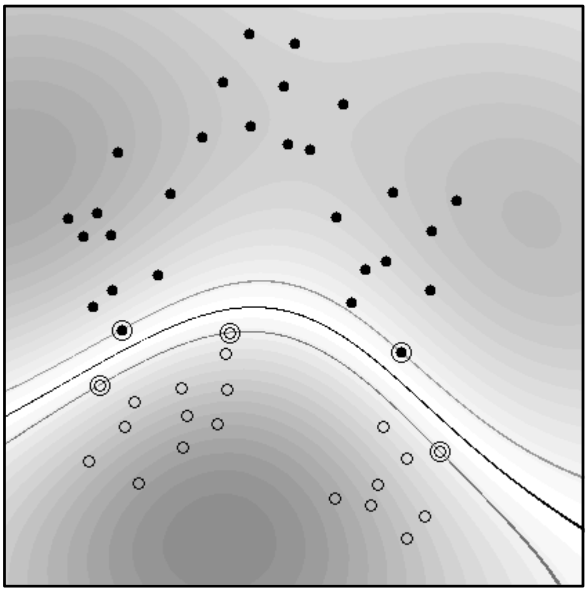

Abbildung 3: Beispiel eines SV-Klassifikators mit RBF-Kern (17). Kreise und Punkte stellen zwei Klassen von Trainingsbeispielen dar. Die mittlere Linie ist die Entscheidungsgrenze, die äußeren Linien kennzeichnen diejenigen Beispiele, die in $F$ am dichtesten an der separierenden Hyperebene liegen. Die Support-Vektoren, die der Algorithmus findet (eingekreist), sind nicht Clusterzentren, sondern Beispiele, die für die gegebene Klassifikationsaufgabe kritisch sind.

ausgenutzt werden, indem *virtuelle* Trainingsbeispiele aus Support-Vektoren erzeugt werden, oder indem die Kernfunktion so gewählt wird, dass die Hyperebene lokal invariant wird. Durch Modifikation der Kernfunktion kann auch die Information genutzt werden, dass Produkte räumlich benachbarter Pixel robustere Merkmale sind als Produkte beliebiger Pixel [7].

## 4.2 Regression

In der Verallgemeinerung des SV-Algorithmus auf Regression [13] schätzt man lineare Funktionen im Merkmalsraum,

$$f(\mathbf{x}) = (\mathbf{w} \cdot \Phi(\mathbf{x})) + b. \tag{18}$$

Um die Eigenschaft der Entwicklung der Lösung in wenigen Support-Vektoren beizubehalten, verwendet man eine Kostenfunktion, die Abweichungen der Trainingsdaten von der Schätzung unterhalb von $\varepsilon > 0$ nicht bestraft,

$$|y - f(\mathbf{x})|_\varepsilon := \max\{0, |y - f(\mathbf{x})| - \varepsilon\}. \tag{19}$$

Die Funktion $f$ findet man wiederum durch Lösung eines quadratisches Optimierungsproblem, das die Eigenschaft hat, nur von Skalarprodukten abzuhängen. Somit können wie im Fall der Mustererkennung Kernfunktionen verwendet werden.
Alternativ zur Interpretation der resultierenden nichtlinearen Schätzung

$$f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{20}$$

als lineare Funktion (18) im Merkmalsraum kann man $f$ auch als Lösung des folgenden regularisierten Problems interpretieren: gegeben sei ein Regularisierungsoperator $P$ (beispielsweise ein Differentialoperator, der die Glattheit von $f$ misst). Minimiere das *regularisierte Risiko*

$$R_{reg}[f] = R_{emp}[f] + \frac{\lambda}{2}\|Pf\|^2. \tag{21}$$

Dies ist äquivalent zum SV-Algorithmus, sofern $k$ eine Green's-Funktion von $P^*P$ ist [12]. So wird ersichtlich, dass der Kern eine Doppelrolle spielt: zum einen legt er die funktionale Form der zugelassenen Lösungen (18) fest; zum anderen bestimmt er die Art der Regularisierung. Wenn man die beiden Aspekte trennt, gelangt man zu allgemeineren Lernverfahren, die bei geeigneter Wahl der Kostenfunktion immer noch als quadratische Optimierungsprobleme lösbar sind [11].
Die optimale Wahl des Parameters $\varepsilon$, der die geforderte Genauigkeit der Schätzung charakterisiert, hängt vom Grad der Verrauschung der Zielwerte $y_i$ ab. Da diese in der Regel nicht a priori bekannt ist, kann es sinnvoll sein, das Problem so umzuschreiben, dass $\varepsilon$ automatisch bestimmt wird, und man stattdessen den Anteil $\nu$ der Datenpunkte vorgibt, die außerhalb der Genauigkeit $\varepsilon$ liegen dürfen [5].

# 5 Kern-PCA

Der Schlüssel zu den exzellenten Ergebnissen von SV-Maschinen liegt in der Verwendung der Kernmethode zur Berechnung von Skalarprodukten. Diese Methode ist jedoch nicht auf SV-Maschinen beschränkt — sie setzt nur voraus, dass ein Algorithmus sich auf Basis von Skalarprodukten ausführen lässt. Daher lag es nahe, dieselbe Methode zur Verallgemeinerung eines anderen vormals linearen Algorithmus zu verwenden. Die Wahl fiel auf den wahrscheinlich am weitesten verbreiteten Algorithmus aus der statistischen Datenanalyse: die *Hauptkomponentenanalyse* (*PCA*, für *principal component analysis*). Die Grundidee der PCA kann wie folgt skizziert werden (vgl. Abbildung 4):

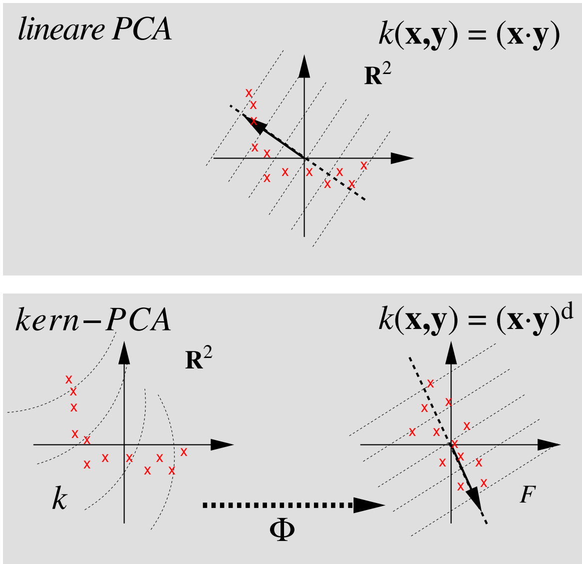


Abbildung 4: Durch Verwendung einer Kernfunktion führt Kern-PCA implizit eine lineare PCA in einem hochdimensionalen Raum durch, der nichtlinear mit dem Ursprungsraum zusammenhängt.

Für einen $N$-dimensionalen Datensatz wird ein Satz orthogonaler Richtungen berechnet, sodass sich die Daten aus den ersten $n$ Richtungen ($n = 1, \dots, N$) mit minimalem quadratischen Fehler rekonstruieren lassen. Dies entspricht der Konstruktion von eindimensionalen Projektionen, die maximale Varianz aufrechterhalten. In der Praxis ist mit der Anwendung der PCA meist die Hoffnung verbunden, durch Extraktion weniger Hauptkomponenten (d.h. durch Verwendung weniger Projektionen) die Dimensionalität der Daten zu reduzieren und gleichzeitig die wesentliche Struktur zu bewahren. Freilich ist der Algorithmus linear, und daher nicht in der Lage, wichtige nichtlineare Strukturen aus Mustern wie z.B. jenen in Abbildung 2 zu extrahieren.
Hier setzt *Kern-PCA* als nichtlineare Verallgemeinerung von PCA an. Zur Ableitung des Algorithmus formuliert man die lineare PCA im Merkmalsraum und reduziert sie auf Skalarprodukte, die sodann durch Kernfunktionen berechnet werden können:
Anhand der nichlinearen Abbildung (6) werden die Daten $\mathbf{x}_1, \dots, \mathbf{x}_\ell \in \mathbb{R}^N$ in den Merkmalsraum $F$ überführt, wo die aus den Daten geschätzte Kovarianzmatrix die Form

$$C = \frac{1}{\ell} \sum_{j=1}^{\ell} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^\top \tag{22}$$

annimmt. Zielwerte $y_1, \dots, y_\ell$ werden nicht benötigt — die PCA ist ein *unüberwachtes* Lernverfahren.
Zum Auffinden der Hauptkomponenten muss folgendes Eigenwertproblem gelöst werden: suche $\lambda > 0, \mathbf{V} \neq 0$ mit

$$\lambda \mathbf{V} = C\mathbf{V} = \frac{1}{\ell} \sum_{j=1}^{\ell} (\Phi(\mathbf{x}_j) \cdot \mathbf{V}) \Phi(\mathbf{x}_j). \tag{23}$$

Hieraus ist ersichtlich, dass für alle Lösungen gilt: $\mathbf{V} \in \operatorname{span}\{\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_\ell)\}$. Daher dürfen wir

$$\mathbf{V} = \sum_{i=1}^{\ell} \alpha_i \Phi(\mathbf{x}_i) \tag{24}$$

ansetzen, und das zu (23) äquivalente System von Gleichungen

$$\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot C\mathbf{V}) \text{ für alle } k = 1, \dots, \ell \tag{25}$$

betrachten. Durch Definition einer $\ell \times \ell$-Matrix

$$K_{ij} := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j) \tag{26}$$

gelangt man zu einem Eigenwertproblem für die Entwicklungskoeffizienten $\alpha_i$, das nurmehr von den Werten der Kernfunktion abhängt:

$$\lambda \boldsymbol{\alpha} = K \boldsymbol{\alpha} \qquad (\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_\ell)^\top) \tag{27}$$

Die Lösungen $(\lambda_k, \boldsymbol{\alpha}^k)$ werden durch die Forderung $\lambda_k(\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k) = 1$ in $F$ normiert. Details der Rechnung, und die Zentrierung der Daten in $F$, sind in [8] ausgeführt.
Zur Merkmalsextraktion projiziert man das Bild eines Musters $\mathbf{x}$ auf $\mathbf{V}^k$ durch

$$(\mathbf{V}^k \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{M} \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{M} \alpha_i^k k(\mathbf{x}_i, \mathbf{x}). \tag{28}$$

Dieser Algorithmus zur nichtlinearen PCA hat also die Eigenschaft, dass die Nichtlinearität über die Kernfunktion an zwei vergleichsweise harmlosen Stellen eingeht: zum einen bei der Berechnung der Matrixelemente von $K$ (26), und zum anderen bei der Auswertung der Entwicklung (28). Wesentlich ist, dass die Berechnung der Haupkomponenten nach wie vor ein lineares Eigenwertproblem ist.
Die Abbildungen 5 – 7 zeigen Beispiele für Merkmalsextraktoren, die durch lineare PCA und Kern-PCA für künstliche Datensätze berechnet wurden.

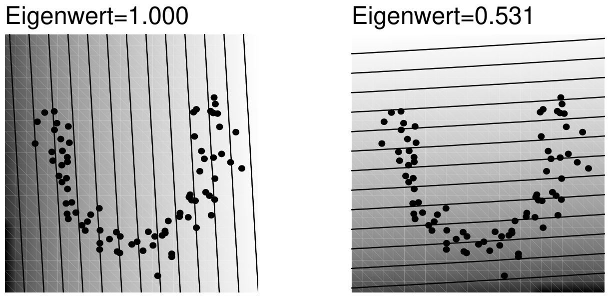


Abbildung 5: Lineare PCA, oder, äquivalent, Kern-PCA mit $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})$. Darstellung der zwei linearen Merkmalsextraktoren auf einem künstlichen Datensatz (entsprechend den beiden Eigenvektoren, nach Größe der Eigenwerte sortiert). Spezifische Grauwerte und Höhenlinien kennzeichnen Gebiete, wo (28) gleiche Werte annimmt. Der erste Extraktor (links) projiziert auf die Richtung maximaler Varianz in den Daten, und verdeutlicht, dass man allein durch Betrachtung der Varianz im Eingaberaum bei dem zugrundeliegenden Datensatz die nichtlineare Struktur nicht angemessen identifizieren kann.

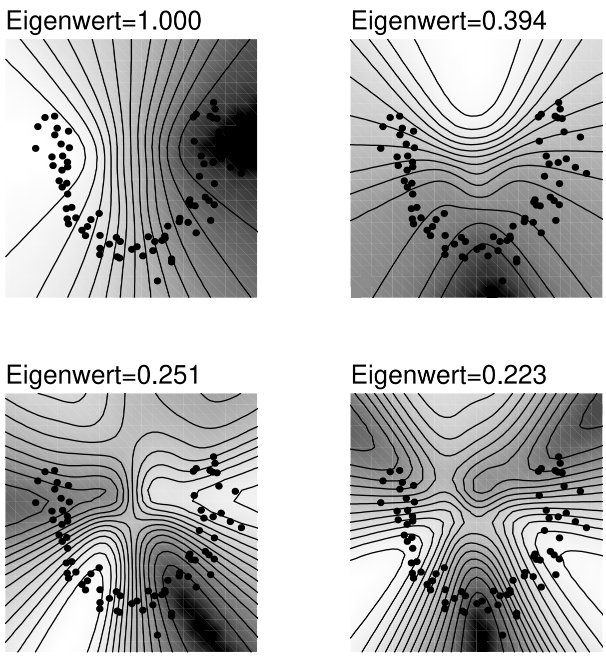


Abbildung 6: Darstellung der ersten 4 nichtlinearen Merkmalsextraktoren bei Kern-PCA mit Sigmoid-Kern (16) auf dem Datensatz aus Abbildung 5. Die Komponenten extrahieren nichtlineare Struktur aus dem Datensatz; so passt sich beispielsweise der erste Extraktor (links oben) besser der Krümmung der Daten an als die entsprechende Komponente in Abbildung 5.

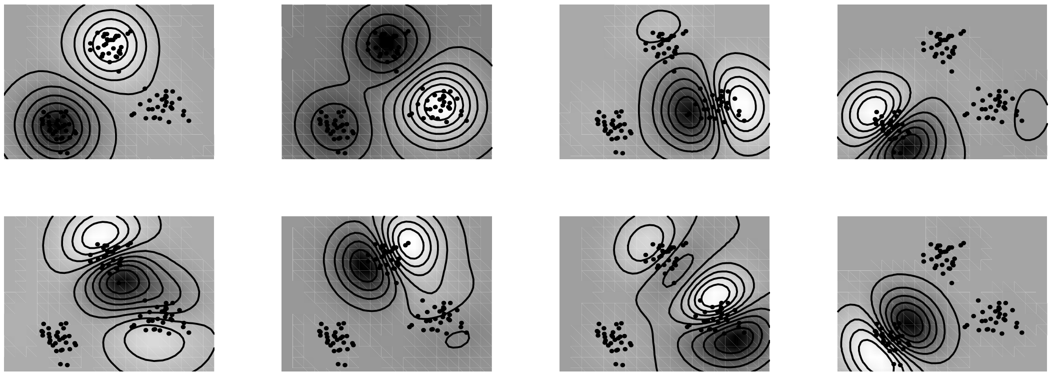

Abbildung 7: Darstellung der ersten 8 nichtlinearen Merkmalsextraktoren, die man durch Anwendung von Kern-PCA mit RBF-Kern (17) auf einen künstlichen Datensatz aus drei Gauß-Clustern erhält (aus [8]). Links oben: die ersten beiden Komponenten teilen die Daten in drei Cluster auf. (Dies geschieht, obwohl der Algorithmus nicht explizit die Clusterung zum Ziel hat.) Diese nichtlineare Struktur in den Daten trägt dementsprechend im Merkmalsraum maximale Varianz. Die höheren Komponenten untersuchen die verbleibende Varianz, die darin besteht, dass die Cluster ausgedehnt sind. Komponenten 3 – 5 halbieren jeweils einen Cluster, wie auch 6 – 8, letztere jedoch orthogonal zu den vorherigen Halbierungen. Ein Matlab-Beispielprogramm für Kern-PCA ist über http://svm.first.gmd.de erhältlich.

Auch auf realistischen Daten sind sehr gute Ergebnisse erzielt worden. Für die Ziffernerkennungsdatenbank der US-Post erreicht man durch Kern-PCA-Merkmalsextraktion und nachfolgende lineare Klassifikation eine Fehlerrate von 4% — besser als klassische Neuronale Netze [1], und Größenordnungen besser als lineare PCA [8].

# 6 Schluss

Durch Verwendung von Kernfunktionen kann einer ganze Reihe von nichtlinearen Lern- und Datenanalyseverfahren eine solide mathematische Grundlage gegeben werden. Kernfunktionen stellen eine theoretisch und algorithmisch fruchtbare Korrespondenz zwischen komplexen Algorithmen und linearen Methoden her: der theoretische Vorteil liegt darin, dass für lineare Algorithmen Ergebnisse und Einsichten vorliegen, die nun für eine große Klasse von Methoden herangezogen werden können. Der algorithmische Vorteil besteht in der

Möglichkeit der *impliziten* Ausführung von komplexen Methoden in hochdimensionalen Räumen, indem man Skalarprodukte durch Kernfunktionen berechnet.
Die Methode der Kernfunktionen setzt lediglich voraus, dass sich ein Algorithmus anhand von Skalarprodukten formulieren lässt. Dies trifft auf eine ganze Reihe klassischer Algorithmen zu. Wichtigster Beitrag der Arbeit ist es, dies aufgezeigt und anhand der Verallgemeinerung der PCA illustriert zu haben.
Diejenigen Leser, die die weitere Entwicklung dieses faszinierenden Forschungsgebietes näher verfolgen möchten, seien auf den Sammelband [6] und die www-Seite http://svm.first.gmd.de verwiesen.

***Danksagung*** Den Betreuern der Arbeit, Heinrich Bülthoff, Stefan Jähnichen und Vladimir Vapnik, möchte ich herzlich danken.

# Literatur

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, und L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541 – 551, 1989.

[2] K.-R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, und V. Vapnik. Predicting time series with support vector machines. In W. Gerstner, A. Germond, M. Hasler, und J.-D. Nicoud, Hrsg., *Artificial Neural Networks — ICANN'97*, S. 999 – 1004, Berlin, 1997. Springer Lecture Notes in Computer Science, Vol. 1327.

[3] B. Schölkopf, C. Burges, und V. Vapnik. Extracting support data for a given task. In U. M. Fayyad und R. Uthurusamy, Hrsg., *Proceedings, First International Conference on Knowledge Discovery & Data Mining.* AAAI Press, Menlo Park, CA, 1995.

[4] B. Schölkopf. *Support Vector Learning.* R. Oldenbourg Verlag, München, 1997.

[5] B. Schölkopf, P. Bartlett, A. Smola, und R. Williamson. Support vector regression with automatic accuracy control. In L. Niklasson, M. Bodén, und T. Ziemke, Hrsg., *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, Berlin, 1998. Springer Verlag. Im Druck.

[6] B. Schölkopf, C.J.C. Burges, und A.J. Smola. *Advances in Kernel Methods — Support Vector Learning.* MIT Press, Cambridge, MA, 1998. Im Druck.

[7] B. Schölkopf, P. Simard, A. Smola, und V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, und S. Solla, Hrsg., *Advances in Neural Information Processing Systems 10*, Cambridge, MA, 1998. MIT Press.

[8] B. Schölkopf, A. Smola, und K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299 – 1319, 1998.

[9] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, und V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Trans. Sign. Processing*, 45:2758 – 2765, 1997.

[10] J. Schürmann. *Pattern Classification: a unified view of statistical and neural approaches.* Wiley, New York, 1996.

[11] A. Smola und B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 1998. Im Druck.

[12] A. Smola, B. Schölkopf, und K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 1998. Im Druck.

[13] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

**Bernhard Schölkopf**, geboren 1968, studierte von 1988 bis 1994 Mathematik, Physik und Philosophie in Tübingen und London. Er erhielt den Lionel Cooper Memorial Prize der Universität London für den besten M.Sc. in Mathematik (1992), und wurde von 1992 bis 1997 von der Studienstiftung des deutschen Volkes gefördert. Nach dem Diplom in Physik (1994) erstellte er am Max-Planck-Institut für biologische Kybernetik und bei AT&T Bell Laboratories seine Dissertation. Er hat Forschungsaufenthalte bei GMD FIRST, am MIT und an der Australian National University verbracht, und ist derzeit bei der GMD tätig, wo er seine wissenschaftlichen Interessen im Bereich des maschinellen Lernens und der Wahrnehmung weiter verfolgt. Sein Projekt *KALIF* (Kern-Algorithmen zum Lernen in Feature-Räumen, mit A. Smola, K. Müller, S. Mika, T. Onoda, und G. Rätsch) wurde 1998 als bestes wissenschaftlich-technisches Projekt der GMD ausgezeichnet.