1

Informatik Forschung und Entwicklung manuscript No.
(will be inserted by the editor)

Lernen mit Kernen

Support-Vektor-Methoden zur Analyse hochdimensionaler Daten

Bernhard Schölkopf, Klaus-Robert Müller, Alexander J. Smola

GMD FIRST, Rudower Chaussee 5, 12489 Berlin

Received: date / Revised version: date

Zusammenfassung Dieser Beitrag erläutert neue Ansätze und Ergebnisse der statistischen Lerntheorie. Nach einer Einleitung wird zunächst das *Lernen aus Beispielen* vorgestellt und erklärt, dass neben dem *Erklären* der Trainingdaten die *Komplexität von Lernmaschinen* wesentlich für den Lernerfolg ist. Weiterhin werden *Kern-Algorithmen in Merkmalsräumen* eingeführt, die eine elegante und effiziente Methode darstellen, verschiedene Lernmaschinen mit kontrollierbarer Komplexität durch *Kernfunktionen* zu realisieren. Beispiele für solche Algorithmen sind *Support-Vektor-Maschinen (SVM)*, die Kernfunktionen zur Schätzung von Funktionen verwenden, oder *Kern-PCA (principal component analysis)*, die Kernfunktionen zur Extraktion von nichtlinearen Merkmalen aus Datensätzen verwendet. Viel wichtiger als jedes einzelne Beispiel ist jedoch die Einsicht, dass *jeder* Algorithmus, der sich anhand von Skalarprodukten formulieren lässt, durch Verwendung von Kernfunktionen nichtlinear verallgemeinert werden kann.

Die Signifikanz der Kernalgorithmen soll durch einen kurzen Abriss einiger industrieller und akademischer Anwendungen unterstrichen werden. Hier konnten wir Rekordergebnisse auf wichtigen praktisch relevanten Benchmarks erzielen.

Schlüsselwörter Maschinelles Lernen, Klassifikation, Mustererkennung, Regression, Merkmalsextraktion, Data Mining, Neuronale Netze, Kernmethoden, Ziffernerkennung, Zeitreihenprognose

Abstract We describe recent developments and results of statistical learning theory. In the framework of learning from examples, two factors control generalization ability: *explaining* the training data by a learning machine of a suitable *complexity*. We describe *kernel algorithms in feature spaces* as elegant and efficient methods of realizing such machines. Examples thereof are *Support Vector Machines (SVM)* and *Kernel PCA (Principal Component Analysis)*. More important than any individual example of a kernel algorithm, however, is the insight that *any* algorithm that can be cast in terms of dot products can be generalized to a nonlinear setting using kernels.

Finally, we illustrate the significance of kernel algorithms by briefly describing industrial and academic applications, including ones where we obtained benchmark record results.

Keywords Machine learning, classification, pattern recognition, regression, feature extraction, data mining, neural networks, kernel methods, digit recognition, time series prediction

CR Subject Classification I.2.6 learning, I.5 pattern recognition, G.3 probability and statistics, I.4.7 feature measurement, I.7.5 document capture, I.4.8 scene analysis, F.1.1 models of computation, H.2.8 database applications

1 Einleitung

Bei der Klassifikation von Daten versucht man üblicherweise einen Satz von diskriminierenden Merkmalen zu extrahieren, die jedes Datum in einer für die nachfolgenden Verarbeitungsschritte geeigneten Weise repräsentieren. Oft ist dies ein Satz von wenigen relevanten *starken* Merkmalen, die die Informationsflut reduzieren sollen.

Das zweite Grundproblem der Diskriminanzanalyse ist, festzustellen, welche Daten relevant für die Klassifikationsentscheidung sind. Mit anderen Worten: gibt es einen Satz von Datenbeispielen, die besonders charakteristisch für ein gegebenes Klassifikationsproblem sind und die die Entscheidung maßgeblicher bestimmen als andere?

In dieser Arbeit illustrieren wir diese beiden Grundprobleme der Klassifikation anhand der Beispiele der nichtlinearen Merkmalsextraktion (Kern-PCA) [27] und

* email: {bs,klaus,smola}@first.gmd.de, Tel. 030-6392-1875, Fax 030-6392-1805, www: http://svm.first.gmd.de

der Support-Vektor-Lernmaschine [35,6,21,36]. Beide Algorithmen setzen sich von der oben beschriebenen Philosophie der starken Merkmale ab. Durch die Verwendung von Kernfunktionen zur effizienten Repräsentation der Daten in einem hochdimensionalen Raum wird implizit eine große Anzahl von Merkmalen erzeugt, die nachfolgenden Verarbeitungsschritten zur Verfügung stehen. Somit spiegelt sich in den beiden genannten Kern-Algorithmen ein Paradigmenwechsel in der Klassifikation und Merkmalsextraktion wider, der bereits 1992 mit der Entwicklung der Support-Vektor-Maschine begonnen hat [3, 35,6,20].

Anstatt mit möglichst einfachen niedrigdimensionalen Modellen realistische Datensätze zu modellieren, besteht nun aufgrund des Einsatzes von Kernfunktionen die Möglichkeit, komplexe nichtlineare Probleme effizient mit hoher Genauigkeit zu lösen. Hierzu gehören Probleme der Klassifikation, Regression, Lösung von Operatorgleichungen (Schätzung von Funktionen, von denen nur transformierte Beobachtungen vorliegen, wie z.B. in medizinischen Tomographie-Untersuchungen), sowie der nichtlinearen Merkmalsextraktion.

SV-Algorithmen sind nicht nur mathematisch besser fundiert als viele der konkurrierenden Methoden, sondern halten inzwischen Rekorde auf folgenden wichtigen Benchmarks: NIST Handschrifterkennung [21], MPI Objekterkennung [2,21], CERN charm quark tagging [33], Santa Fe Zeitreihen Benchmark D [14], Gesichtsdetektion (vom MIT gehalten [9]), sowie Reuter Textkategorisierung (Microsoft Research [9]).

Zunächst wollen wir, in Anlehnung an [22], die Grundlagen der Kern-Algorithmen beschreiben.

2 Lernen aus Beispielen

Die statistische Lerntheorie [35,36] beschäftigt sich mit dem Lernen aus empirischen Daten mittels 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 Bildern, die in einer Auflösung 16×16 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 $N \in \mathbb{N}$ und $\tau > 0$ 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 sich dieser *statistische* Zugang zum Schätzen von Regularitäten dadurch auszeichnet, dass er auf Probleme angewandt werden kann, bei dem die explizite Modellierung eines Zusammenhangs schwierig oder unmöglich ist, dafür aber empirische Beobachtungen vorliegen. 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 wurden gemäß einer unbekannten Wahrscheinlichkeitsverteilung $P(\mathbf{x}, y)$ erzeugt. 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 die Eigenschaften hin 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.

3 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*

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

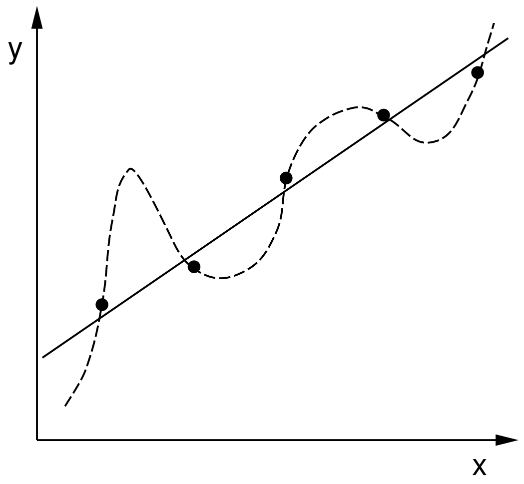

Abbildung 1 Aus einer gegebenen Menge von Beispielen (im Bild: schwarze Punkte) soll ein unbekannter funktionaler Zusammenhang geschätzt werden. Die statistische Lerntheorie formalisiert, in welchem Sinne die Gerade als Schätzung vorzuziehen ist (siehe Text).

also einen niedrigen *Trainingsfehler*, erreicht — ebenso 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 der Funktion 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 der Trainingsdaten allein kann also 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 verdeutlicht 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 *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-(Vapnik-Chervonenkis)Dimension* [35,36], bildet den Kern der statistischen Lerntheorie. Anhand der VC-Dimension oder verwandter Kapazitätsmaße 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. [35,36]). 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 gut funktionieren [8,16], sich theoretisch jedoch nur schwer analysieren lassen.

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 Lernpro-

[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 niedriger VC-Dimension die Daten erklären kann, dann hat man mit hoher Wahrscheinlichkeit richtig gelernt.
Darüber hinaus sollte betont werden, dass die Schranke oft pessimistisch ist, da sie für alle denkbaren zugrundeliegenden Wahrscheinlichkeitsverteilungen P gilt. Nichtsdestotrotz beschreibt sie das Skalierungsverhalten des tatsächlichen Testfehlers oft sehr gut [20,25].

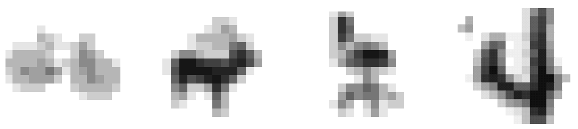

Abbildung 2 Beispielbilder aus verschiedenen Mustererkennungsdatenbanken [21]

bleme anwendbar zu sein. Zum anderen ist ihre Komplexität beschränkt, da sie mathematisch äquivalent zu einer sehr einfachen Klasse von Funktionen in Merkmalsräumen sind, wie im folgenden beschrieben wird.

4 Kern-Algorithmen in Merkmalsräumen

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

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

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

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

behandelt. Im Rahmen der klassischen Statistik erscheint dies zumindest fragwürdig: aufgrund des sogenannten *curse of dimensionality* wächst die Schwierigkeit eines Schätzproblems mit der Dimensionalität des Raumes, da experimentell mehr Daten benötigt werden, um den Raum gut abzudecken. Die statistische Lerntheorie zeigt jedoch, dass dies auf Kern-Algorithmen so nicht zutrifft. Im Gegenteil: in manchen Fällen ist das Lernen in F einfacher, da dort mit einer sehr einfachen Klasse von Algorithmen bzw. Lernmaschinen gearbeitet werden kann — die notwendige Reichhaltigkeit der Funktionenklasse steckt in der Abbildung Φ.

Eine in der Mustererkennung besonders verbreitete Klasse von Merkmalen ist die der Monome [29]. Für das eingangs besprochene Beispiel der 16×16-Bilder (Abbildung 2) könnte man also eine Abbildung Φ definieren, die jedes Bild in einen Vektor abbildet, der alle Produkte von jeweils 5 der 256 Pixel als Komponenten enthält. Dies führt jedoch zu einem 10^{10}-dimensionalen Raum. Selbst wenn man die lerntheoretische Komplexität des nachfolgenden Algorithmus kontrollieren kann, so bleibt immer noch das praktische Problem, in diesem hochdimensionalen Raum überhaupt einen Algorithmus auszuführen.

An dieser Stelle kommt eine sehr effektive Methode zum Berechnen von Skalarprodukten mittels Kernfunktionen ins Spiel. Beginnen wir mit einem einfachen Beispiel [35]. Sei

$$\Phi : \mathbb{R}^2 \to F$$

$$(x_1, x_2) \mapsto (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2), \tag{8}$$

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 $\mathbf{x}, \mathbf{y}$ [35, 27].
- ist $k : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ stetiger Kern eines positiven Integraloperators auf $L_2(\mathcal{C})$ (für eine kompakte Menge $\mathcal{C} \subset \mathbb{R}^N$, die die Muster enthält), d.h.

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

  so existieren ein Merkmalsraum F und eine Abbildung $\Phi : \mathbb{R}^N \to F$ sodass $k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$ [3]. 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 ψ_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 Φ.

Man kann also das Skalarprodukt in F berechnen, *ohne* explizit die Abbildung nach F zu berechnen [1]. Die direkte Folgerung, explizit erstmals in [27] formuliert, ist: *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* (linear) im Merkmalsraum, erzeugen aber trotzdem reichhaltige (nichtlineare) Funktionenklassen im Eingaberaum, und beide lassen sich so formulieren, dass sie nur von Skalarprodukten abhängen.

[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 [21].

5 Support-Vektor-Maschinen

5.1 Klassifikation

Support-Vektor(SV)-Maschinen zur Mustererkennung separieren zwei gegebene Klassen von Trainingsdaten in einem hochdimensionalen Merkmalsraum durch eine Hyperebene $\mathbf{w}$ minimaler VC-Dimension [35].

$$f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b). \tag{14}$$

Durch die zuvor besprochene Kernmethode kann das Skalarprodukt $\mathbf{w} \cdot \Phi(\mathbf{x})$ (im Merkmalsraum F), welches potentiell algorithmisch beliebig aufwendig sein kann, im Eingaberaum ausgewertet werden. Dies führt zu der Entscheidungsfunktion

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{\ell} y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b\right). \tag{15}$$

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:

$$\begin{array}{ll} \text{maximiere} & W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{unter den Nebenbedingungen} & \sum_{i=1}^{\ell} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell. \end{array} \tag{16}$$

Die Konstante $C > 0$ spielt die Rolle eines Regularisierungsparameters, der die relative Gewichtung der Minimierung von R_{emp} und der Minimierung des Komplexitätsterms bestimmt (vgl. (5)). Die SV-Entscheidungsfunktion ist durch (16) eindeutig bestimmt.

Die Ableitung des Optimierungsproblemes verwendet Standardmethoden der Theorie konvexer Optimierung. Man stellt, zunächst für den linearen Fall ($\Phi = \mathrm{id}$), eine Zielfunktion $\tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell} \xi_i$ auf, die es unter den Nebenbedingungen $n_i(\mathbf{w}, \xi_i) := y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1 + \xi_i \geq 0$ und $\xi_i \geq 0$ ($i = 1, \ldots, \ell$) zu minimieren gilt. Man kann zeigen, dass die Minimierung von $\|\mathbf{w}\|$ der Maximierung des Klassenabstands senkrecht zur Entscheidungsebene $(\mathbf{w} \cdot \mathbf{x}) + b = 0$ und der Minimierung der VC-Dimension entspricht.[3] Der zweite Term in τ sorgt dafür, dass möglichst viele der Trainingsbeispiele die Nebenbedingung n_i mit $\xi_i = 0$ erfüllen und daher auf der richtigen Seite der Entscheidungsebene liegen.

Als nächster Schritt wird mittels positiver Lagrange-Multiplikatoren $\boldsymbol{\alpha}, \boldsymbol{\beta}$ eine Lagrange-Funktion $\tau(\mathbf{w}, \boldsymbol{\xi}) - \sum_i \alpha_i n_i(\mathbf{w}, \xi_i) - \sum_i \beta_i \xi_i$ konstruiert. Ihre Minimierung über $\mathbf{w}$ und $\boldsymbol{\xi}$ liefert einen Satz von Bedingungen, der die Eliminierung der ursprünglichen Variablen erlaubt und zu einem dualen Problem führt, das nurmehr von $\boldsymbol{\alpha}$ abhängt. Da jenes Problem die Trainingsmuster nur in Form von Skalarprodukten verwendet, lassen sich zuletzt Kernfunktionen $k(\mathbf{x}, \mathbf{y})$ für alle Skalarprodukte substituieren, was zu (16) führt.

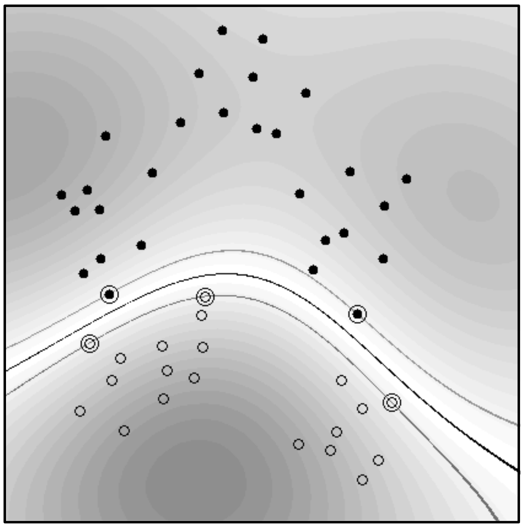

Abbildung 3 Beispiel eines SV-Klassifikators mit RBF-Kern (18). Kreise und Punkte stellen zwei Klassen von Trainingsbeispielen dar. Die mittlere Linie ist die Entscheidungsgrenze, die äußeren Linien kennzeichnen diejenigen Beispiele, die in F am dichtesten an der separierenden Hyperebene liegen. Die Support-Vektoren, die der Algorithmus findet (eingekreist), sind nicht Clusterzentren, sondern Beispiele, die für die Lösug der gegebenen Klassifikationsaufgabe kritisch sind.

Empirisch findet man, dass oft wenige Beispiele des gesamten Datensatzes, die *Support-Vektoren*, Hyperebene und Entscheidungsfunktion definieren — für alle anderen Daten $\mathbf{x}_i$ gilt $\alpha_i = 0$: sie sind für das Mustererkennungsproblem irrelevant [20] (cf. Abb.3). Durch die Wahl verschiedener Kernfunktionen, und dadurch verschiedener Merkmalsräume, ergeben sich als Spezialfälle der SVM auch Neuronale Netze,

$$k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \Theta), \tag{17}$$

Gaußsche Radiale-Basis-Funktionen(RBF)-Netze (vgl. Abbildung 3)

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\,\sigma^2)\right), \tag{18}$$

und Polynomklassifikatoren (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 [20,21].

[3] Die VC-Dimension charakterisiert die Reichhaltigkeit einer Funktionenklasse. Eine Einschränkung auf Funktionen, die einen großen Klassenabstand aufweisen, entspricht daher einer Verringerung der VC-Dimension. Details sind z.B. in Einleitung und Kapitel 4 von [23] ausgeführt.

5.2 Regression

In der Verallgemeinerung des SV-Algorithmus auf Regression [35] schätzt man lineare Funktionen im Merkmalsraum,

$$f(\mathbf{x}) = (\mathbf{w} \cdot \Phi(\mathbf{x})) + b. \tag{19}$$

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

$$|y - f(\mathbf{x})|_\varepsilon := \max\{0, |y - f(\mathbf{x})| - \varepsilon\}. \tag{20}$$

Die Funktion f findet man wiederum durch Lösung eines quadratisches Optimierungsproblem, das die Eigenschaft hat, nur von Skalarprodukten abzuhängen. Für die Klasse allgemeiner konvexer Kostenfunktionen kann ebenfalls effizient eine Lösung gefunden werden [33,31]. Somit können wie im Fall der Mustererkennung Kernfunktionen verwendet werden.

Alternativ zur Interpretation der resultierenden nichtlinearen Schätzung

$$f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{21}$$

als lineare Funktion (19) 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*

$$R_{reg}[f] = R_{emp}[f] + \frac{\lambda}{2}\|Pf\|^2. \tag{22}$$

Dies ist äquivalent zum SV-Algorithmus, sofern k eine Green's-Funktion von P^*P ist [32]. So wird ersichtlich, dass der Kern eine Doppelrolle spielt: zum einen legt er die funktionale Form der zugelassenen Lösungen (21) 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 [31].

Die optimale Wahl des Parameters ε, 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 ε automatisch bestimmt wird, und man stattdessen den Anteil ν der Datenpunkte vorgibt, die außerhalb der Genauigkeit ε liegen dürfen [28].

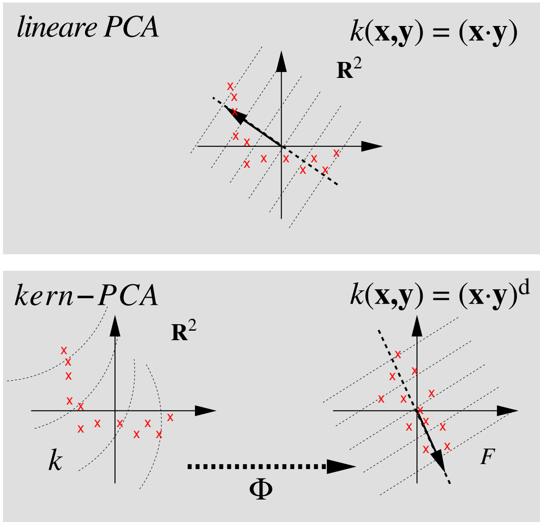

Abbildung 4 Durch Verwendung einer Kernfunktion führt Kern-PCA implizit eine lineare PCA in einem hochdimensionalen Raum durch, der nichtlinear mit dem Ursprungsraum zusammenhängt [27].

6 Kern-PCA

Die oben beschriebene Kern-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):

Für einen N-dimensionalen Datensatz wird ein Satz orthogonaler Richtungen berechnet, sodass sich die Daten aus den ersten n Richtungen ($n = 1, \ldots, N$) mit minimalem quadratischen Fehler rekonstruieren lassen. Dies entspricht der Konstruktion eindimensionaler 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 und 5 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. Im folgenden soll der Fall von bereits im Merkmalsraum zentrierten Daten ausgeführt werden; darüber hinausgehende Details sind in [27] ausgeführt.

Durch die nichlineare Abbildung (6) werden die Daten $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \mathbb{R}^N$ in den Merkmalsraum F überführt, in dem die aus den Daten geschätzte Kovarianzmatrix die Form

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

annimmt. Zielwerte $y_1, \ldots, y_\ell$ 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

$$\lambda \mathbf{V} = C\mathbf{V} = \frac{1}{\ell} \sum_{j=1}^{\ell} (\Phi(\mathbf{x}_j) \cdot \mathbf{V}) \Phi(\mathbf{x}_j). \tag{24}$$

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

$$\mathbf{V} = \sum_{i=1}^{\ell} \alpha_i \Phi(\mathbf{x}_i) \tag{25}$$

ansetzen, und das zu (24) äquivalente System von Gleichungen

$$\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot C\mathbf{V}) \text{ für alle } k = 1, \ldots, \ell \tag{26}$$

betrachten. Durch Einsetzen von (23) und (25) sowie Definition einer $\ell \times \ell$-Matrix

$$K_{ij} := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j) \tag{27}$$

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

$$\lambda \boldsymbol{\alpha} = K \boldsymbol{\alpha} \qquad (\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\ell)^\top) \tag{28}$$

Die Normierung des k-ten Eigenvektors $\mathbf{V}^k$ auf die Länge 1 in F läuft auf eine Normierung seiner Entwicklungskoeffizienten $\boldsymbol{\alpha}^k$) gemäß $\lambda_k(\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k) = 1$ hinaus.

Zur Merkmalsextraktion projiziert man das Bild eines Musters $\mathbf{x}$ auf $\mathbf{V}^k$ durch

$$(\mathbf{V}^k \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{\ell} \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{\ell} \alpha_i^k k(\mathbf{x}_i, \mathbf{x}). \tag{29}$$

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 (27), und zum anderen bei der Auswertung der Entwicklung (29). Wesentlich ist, dass die Berechnung der Haupkomponenten nach wie vor ein lineares Eigenwertproblem ist. Die Größe der Matrix K ist hierbei durch die Anzahl der Trainingsbeispiele gegeben. Bei großen Trainingsmengen kann es daher sinnvoll sein, approximative

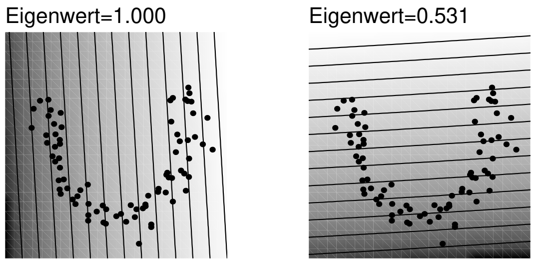

Abbildung 5 Lineare PCA, oder, äquivalent, Kern-PCA mit $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})$. Darstellung der zwei linearen Merkmalsextraktoren auf einem künstlichen Datensatz (entsprechend den beiden Eigenvektoren, nach Größe der Eigenwerte sortiert). Spezifische Grauwerte und Höhenlinien kennzeichnen Gebiete, wo (29) 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.

Methoden zu verwenden. So kann die PCA auf einer Teilmenge der gesamten Daten berechnet werden, und nachfolgende Verarbeitungsschritte können wieder die gesamte Trainingsmenge miteinbeziehen. Die im nächsten Abschnitt besprochenen Experimente verwendeten beispielsweise für die PCA nur 3000 Trainingsbeispiele.[4]

Die Abbildungen 5 – 7 zeigen Beispiele für Merkmalsextraktoren, die durch lineare PCA und Kern-PCA für künstliche Datensätze berechnet wurden.

7 Anwendungen

Die ersten Arbeiten über SV-Maschinen bei den AT&T Bell Labs verwendeten allesamt Ziffernerkennungs-Benchmarks zur Evaluierung der neuen Algorithmen. Schnell wurden Genauigkeiten erreicht [6,20,21,26], die sich mit den besten Verfahren messen konnten. Auf dem USPS und NIST OCR Benchmarks waren dies konvolutive Mehrschicht-Perzeptrons, auch bei AT&T entwickelt [11,4, 30].

Man erkennt in der Tabelle das gute Abschneiden der SVM.[5] Regularisierte RBF-Netze liegen zwar gleichauf mit Standard-SVMs, jedoch kann das SVM Ergebnis

[4] Eine alternative Methode wird in [24] beschrieben. Wählt man $\{\mathbf{z}_1, \ldots, \mathbf{z}_m\} = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, so führt lineare PCA auf den durch $\Phi_m(\mathbf{x}) := (k(\mathbf{z}_1, \mathbf{x}), \ldots, k(\mathbf{z}_m, \mathbf{x}))$ abgebildeten Daten zu einem Eigenwertproblem für K^2 mit denselben Eigenvektoren wie Kern-PCA. Durch Wahl einer kleineren Menge $\{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$ lässt sich diese 'empirische Kernabbildung' Φ_m als Abbildung in einen niedrigdimensionaleren Raum approximieren, in welchem sich das Eigenwertproblem schneller lösen lässt.

[5] Ergänzend sei gesagt, dass das Training der SVM auf einem Pentium-Rechner rund eine Stunde dauert; die Klassifikation der Testmenge nimmt größenordnungsmäßig eine Minute in Anspruch. Neuronale Netze sind im Training meist langsamer, in der Klassifikation aber schneller; bei prototy-

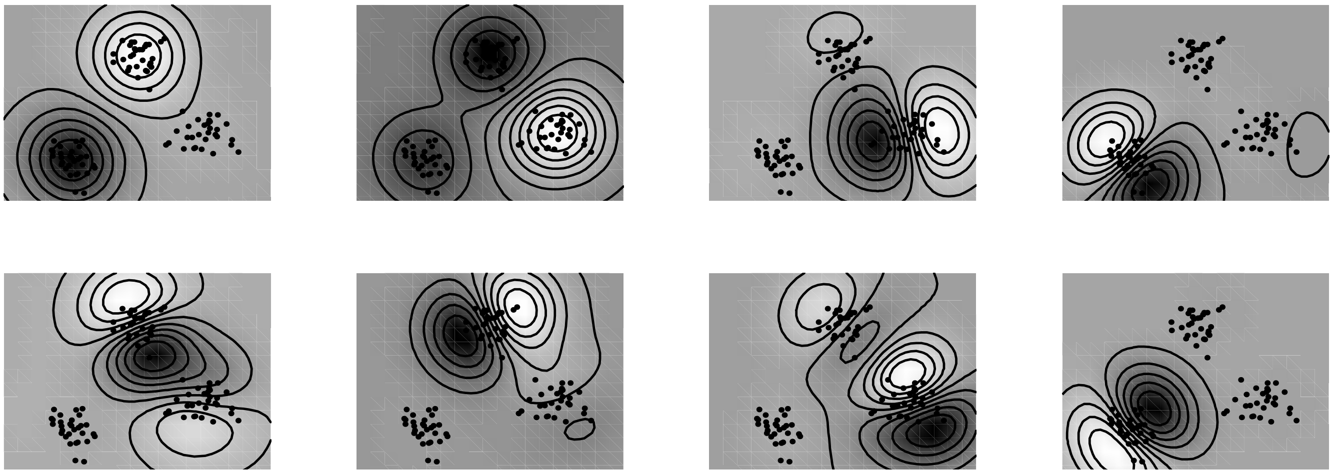

Abbildung 7 Darstellung der ersten 8 nichtlinearen Merkmalsextraktoren, die man durch Anwendung von Kern-PCA mit RBF-Kern (18) auf einen künstlichen Datensatz aus drei Gauß-Clustern erhält (aus [27]). Links oben: die ersten beiden Komponenten teilen die Daten in drei Cluster auf. (Dies geschieht, obwohl 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 orthogonal zu den vorherigen Halbierungen.

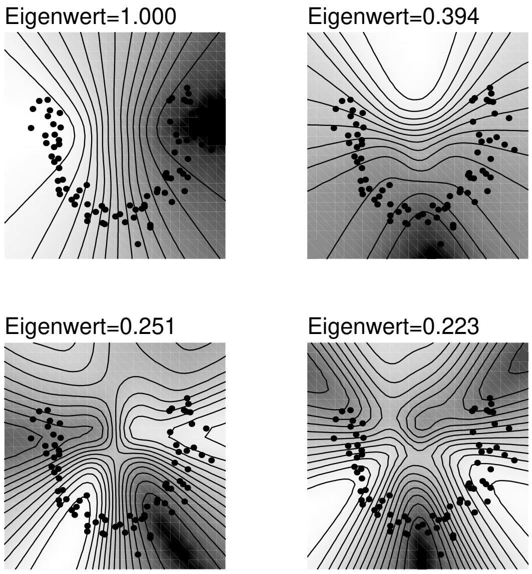

Abbildung 6 Darstellung der ersten 4 nichtlinearen Merkmalsextraktoren bei Kern-PCA mit Sigmoid-Kern (17) auf dem Datensatz aus Abbildung 5. Die Komponenten extrahieren nichtlineare Struktur aus dem Datensatz; so passt sich beispielsweise der erste Extraktor (links oben) besser der Krümmung der Daten an als die entsprechende Komponente in Abbildung 5.

lineare PCA & lineare SVM (Schölkopf et al. [27])	8.7%
k-Nearest Neighbor (Simard et al. [30])	5.9%
LeNet1 (LeCun et al. [11])	5.0%
Regularisierte RBF-Netze (Rätsch [19])	4.1%
Kern-PCA & lineare SVM (Schölkopf et al. [27])	4.0%
SVM (Schölkopf et al. [20])	4.0%
Invariante SVM (Schölkopf [26])	3.0%
Boosting (Drucker et al. [7])	2.6%
Tangent Distance (Simard et al. [30])	2.6%
Menschliche Fehlerrate	2.5%

Tabelle 1 Klassifikationsfehler in % für Ziffernerkennung auf dem US Postal service benchmark (USPS) mit 7291 Trainingsbeispielen. Hier schneiden invariante SVM nur wenig über dem Rekordergebnis ab. Bemerkenswert daran ist, dass die Algorithmen in [11, 7, 30] auf einer Datenmenge trainiert wurden, die um maschinengeschriebene Ziffern ergänzt waren, die nach Aussage von [7] den Testfehler verringern. Alle Ergebnisse beziehen sich auf dieselbe Testmenge von 2007 Ziffern; Zurückweisung (reject) von Mustern war nicht zugelassen. Sowohl SVM als auch Kern-PCA wurden mit polynomialen Kernen (10) trainiert.

noch weiter durch Einbeziehen von *a-priori-Wissen* verbessert werden. Letzteres 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 die Klassenzugehörigkeiten nicht verändern. Dies kann 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 räum-

penbasierten Ansätzen wie Tangent Distance verhält es sich umgekehrt.

lich benachbarter Pixel robustere Merkmale sind als Produkte beliebiger Pixel [26].

Auch Kern-PCA konnte für die Ziffernerkennungsdatenbank des US-Postal Service (USPS) sehr gute Ergebnisse erzielen, so erreicht man durch Kern-PCA-Merkmalsextraktion und nachfolgende lineare Klassifikation eine Fehlerrate von 4% — besser als klassische Neuronale Netze [11], und Größenordnungen besser als lineare PCA mit 8.7% [27].

Obgleich der USPS Datensatz lange als Standard-Benchmark bei AT&T Verwendung fand, wurde im Lauf der Zeit klar, dass größere Datenbanken zur verlässlicheren Evaluierung der verschiedenen Klassifikatoren vonnöten waren. Der vom NIST zusammengestellte Datensatz von 120000 handgeschriebenen Ziffern (davon die Hälfte als Trainingsmenge) füllte diese Lücke; gleichzeitig stellte er ein realistischeres Modell für praktische Anwendungen in der maschinellen Ziffernerkennung dar, wo in der Regel mit großen Datensätzen gearbeitet wurde. Auf diesem Datensatz schneiden invariante SVMs mit 0.8% Fehler [21] besser ab als der Rekordhalter auf der USPS Datenbank (Tangent Distance), und auch besser als alle anderen Einzelklassifikatoren, dicht gefolgt von LeNet 5 mit 0.9% Fehler. Ein *Ensemble* von LeNet 4 Netzwerken, trainiert auf einer Datenbank die Millionen künstlich generierter Beispiele enthielt, liegt hier jedoch noch ein wenig besser (0.7%) [4].

Durch die beschriebenen Ergebnisse wurde klar, dass die SV-Maschinen einen leistungsfähigen Ansatz darstellen, und andere Anwendungen ließen nicht lange auf sich warten. Zunächst waren dies weitere Mustererkennungsprobleme, wie 3D-Objekterkennung [2], Gesichtsdetektion [9], sowie Fußgängerdetektion [15]. Ein Problem, das den Stärken von SV-Maschinen in besonderem Maße entgegenzukommen scheint, ist die Textkategorisierung. Ein typischer Benchmark ist die Reuters-22173 Sammlung. Hier wurden 21450 Reuters News-Stories aus dem Jahre 1997 gesammelt und nach 135 verschiedenen thematischen Kategorien indiziert, um den Reuters-Kunden den Zugriff auf die Datenbank zu erleichtern. Die Reuters-Dokumente werden entsprechend ihrer Dokument- und Worthäufigkeit als ca. 10000-dimensionale Trainingsvektoren codiert. In diesem äußerst hochdimensionalen Problem erreichen SVMs exzellente Ergebnisse [9,10]. Ferner wurden SVMs zur Lösung inverser Probleme eingesetzt [36,37].

Nachdem der SV-Algorithmus auf Regression verallgemeinert worden war, konnten auch für Funktionsschätzprobleme wie das Boston-Housing-Problem [34] und verschiedene Zeitreihenvorhersageprobleme Rekordergebnisse [12,14] erreicht werden. Auf dem Benchmarkdatensatz D des Santa-Fe-Wettbewerbs zur Zeitreihenvorhersage konnte eine Verbesserung der Vorhersagequalität von 30% [14] bezüglich des bisherigen Rekordes erzielt werden. Wir führen unser gutes Ergebnis auf eine saubere Modellierung des nichtstationären Prozesses und auf die guten Regularisierungseigenschaften der eingesetzten SVMs zurück. Letztere machen sich besonders im Falle weniger und gleichzeitig hochdimensionaler Trainingsbeispiele bemerkbar.

Ein anfängliches Problem von SV-Maschinen war die Klassifikationsgeschwindigkeit. Zwar ist das Training, also die Lösung von (16), als quadratisches Optimierungsproblem effizient lösbar,[6] die Komplexität der Auswertung der gefundenen Entscheidungsfunktion (15) (nach dem Training) ist jedoch proportional zur Anzahl der Support-Vektoren, welche bei stark verrauschten Problemen einen wesentlichen Teil der Datenbank umfassen kann. Um dieses Problem zu beheben, wurden spezielle Approximationsverfahren entwickelt, die SV-Maschinen in denselben Geschwindigkeitsbereich wie Neuronale Netze brachten [5,23].

8 Diskussion

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 von Algorithmen der klassischen Statistik zu. Ein Beitrag unserer Arbeit besteht darin, dies aufgezeigt und anhand der Verallgemeinerung der PCA illustriert zu haben.

Die praktische Einsetzbarkeit von SVMs konnten wir anhand verschiedener Rekordergebnisse illustrieren, beispielsweise in der Handschrifterkennung und der Zeitreihenvorhersage. Darüber hinaus kann Kern-PCA nicht nur zur Vorverarbeitung und Extraktion nichtlinearer Merkmale eingesetzt werden, vielmehr sind Anwendungen zur nichtlinearen Entrauschung, Kompression und Visualisierung ebenfalls denkbar und teilweise bereits realisiert [13]. Parallel zu diesen praktischen Arbeiten,

[6] Die Komplexität des Optimierungsproblems skaliert zunächst ungefähr mit der dritten Potenz der Anzahl der Trainingsbeispiele. Jedoch können Methoden zur Dekomposition des Problems gefunden werden, die in etwa quadratisch oder sogar noch besser skalieren [17,18,10]. Die Grundidee dieser Methoden liegt darin, dass in der Lösung von (16) typischerweise die meisten der Variablen an den Rändern von $[0, C]$ zu liegen kommen, sodass es im wesentlichen genügt, die Optimierung über die verbleibenden vorzunehmen.

und grundlegend für praktische Arbeiten in der Zukunft, ist in jüngster Zeit auch unser theoretisches Verständnis der zugrundeliegenden statistischen und funktionalanalytischen Prinzipien wesentlich erweitert worden [38, 25].

Leser, die die Entwicklung dieses faszinierenden Forschungsgebietes näher verfolgen möchten, seien auf den Sammelband [23] und die www-Seite http://svm.first.gmd.de verwiesen, wo auch Programme für Kern-PCA und Support-Vektor-Maschinen frei erhältlich sind.

Danksagung Wir danken für Förderung durch die DFG in den Projekten JA 379/51, JA 379/71 und JA 379/9-1, sowie durch die EU im Projekt 25387-STORM. Für Diskussionen mit Heinrich Bülthoff, Stefan Jähnichen und Vladimir Vapnik möchten wir herzlich danken.

Literatur

1. M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821 – 837, 1964.
2. V. Blanz, B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view-based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, pages 251 – 256, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.
3. B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.
4. L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Müller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th International Conference on Pattern Recognition and Neural Networks, Jerusalem*, pages 77 – 87. IEEE Computer Society Press, 1994.
5. C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings, 13th Intl. Conf. on Machine Learning*, pages 71–77, San Mateo, CA, 1996. Morgan Kaufmann.
6. C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273 – 297, 1995.
7. H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705 – 719, 1993.
8. S. Haykin. *Neural Networks : A Comprehensive Foundation*. Macmillan, New York, 1998. 2nd edition.
9. M.A. Hearst, B. Schölkopf, S. Dumais, E. Osuna, and J. Platt. Trends and controversies — support vector machines. *IEEE Intelligent Systems*, 13:18 – 28, 1998.
10. T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185 – 208. MIT Press, Cambridge, MA, 1999.
11. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541 – 551, 1989.
12. D. Mattera and S. Haykin. Support vector machines for dynamic reconstruction of a chaotic system. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 211 – 241. MIT Press, Cambridge, MA, 1999.
13. S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In *Advances in Neural Information Processing Systems 11*, 1999.
14. K.-R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 243 – 253. MIT Press, Cambridge, MA, 1999.
15. M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates. In *Proc. Computer Vision and Pattern Recognition*, pages 193–199, Puerto Rico, June 16–20 1997.
16. G. B. Orr and K.-R Müller. *Neural Networks: Tricks of the Trade*. LNCS 1524, Springer, Heidelberg, 1998.
17. E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276 – 285, New York, 1997. IEEE.
18. J. Platt. Fast training of SVMs using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185 – 208. MIT Press, Cambridge, MA, 1999.
19. G. Rätsch. Ensemble-Lernmethoden zur Klassifikation. Diplomarbeit, Universität Potsdam, 1998.
20. B. Schölkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings, First International Conference on Knowledge Discovery & Data Mining*. AAAI Press, Menlo Park, CA, 1995.
21. B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997. Doktorarbeit, TU Berlin.
22. B. Schölkopf. Support-Vektor-Lernen. In G. Hotz, H. Fiedler, P. Gorny, W. Grass, S. Hölldobler, I. O. Kerner, and R. Reischuk, editors, *Ausgezeichnete Informatikdissertationen 1997*, pages 135 – 150. Teubner, Stuttgart, 1998.
23. B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1999.
24. B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 1999. in press.
25. B. Schölkopf, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Kernel-dependent support vector error bounds. In *Proceedings ICANN*, 1999. to appear.
26. B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan,

M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640 – 646, Cambridge, MA, 1998. MIT Press.

27. B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299 – 1319, 1998.
28. B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *To appear in: Neural Computation*, 1999. Also: NeuroColt2-TR 1998-031.
29. J. Schürmann. *Pattern Classification: a unified view of statistical and neural approaches.* Wiley, New York, 1996.
30. P. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 50–58, San Mateo, CA, 1993. Morgan Kaufmann.
31. A. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22:211 – 231, 1998.
32. A. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.
33. A.J. Smola. *Learning with Kernels.* PhD thesis, Technische Universität Berlin, 1998. Also: GMD Research Series No. 25.
34. M. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with ANOVA decomposition kernels. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 285 – 291. MIT Press, Cambridge, MA, 1999.
35. V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.
36. V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.
37. J. Weston, A. Gammerman, M. Stitson, V. Vapnik, V. Vovk, and C. Watkins. Support vector density estimation. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 293 – 305. MIT Press, Cambridge, MA, 1999.
38. R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. Technical Report NC-TR-98-019, NeuroCOLT, http://www.neurocolt.com, 1998.