Klassifikation mit Support-Vektoren

von Sami Celik

Inhaltsverzeichnis

Zusammenfassung

In der vorliegenden Arbeit wird die Support-Vektor-Maschine (SVM) (nach V. Vapnik [1]) zur Lösung von Klassifikationsaufgaben vorgestellt. Ziel der SVM (die nur für 2-Klassenprobleme definiert ist, aber leicht auf K-Klassenprobleme übertragbar werden kann) ist es, die Klassen durch eine lineare Entscheidungsfläche zu trennen, wobei diese Fläche einen maximalen Abstand von den Daten haben soll. Dadurch wird eine hohe Generalisierungsfähigkeit der SVM gewährleistet. Neben der Konstruktion der optimalen Entscheidungsfläche (über die sogenannten Support-Vektoren) wird im folgenden auch die effiziente Berechnung von Skalarprodukten in hochdimensionalen Räumen behandelt.

Keywords: Mustererkennung, Support-Vektor-Maschine, Kernfunktionen

1 Einleitung

Informationsverarbeitung als Oberbegriff für alle Methoden der Gewinnung, Kontrolle, Verarbeitung und Interpretation von Informationen gehört seit jeher zum Wesen der Medizin. Die Systematisierung des Wissens um Krankheiten, die Vermehrung der zu berücksichtigenden Daten durch neue Untersuchungs- und Behandlungstechniken und automatisierte Verfahren (z.B. Labordatenverarbeitung) haben zur Entwicklung der Medizinischen Informatik geführt. Diese nutzt die Methoden der Informatik für die Krankenversorgung, die medizinische Forschung und die Betriebsablaufsteuerung. Ein Tätigkeitsschwerpunkt der technisch medizinischen Informatik ist die Bildverarbeitung und damit auch die Mustererkennung. Die Mustererkennung beschäftigt sich mit den mathematischen Aspekten der automatischen Verarbeitung und Interpretation von Mustern. Ein eminent wichtiges Teilgebiet der Mustererkennung ist die Klassifikation, womit sich diese Seminararbeit im folgenden beschäftigt.

Klassifikation bedeutet im Sinne der Mustererkennung, die Zuordnung eines Testmusters in eine von mehreren möglichen Zielklassen. Allgemein sind zunächst folgenden Schritte notwendig,

- Stichprobe von Mustern wird eingeteilt in eine Trainings- und Testmenge
- Merkmalsextraktion: Geeignete Merkmale aus Muster M_i bilden Merkmalsvektor x_i

Die Muster werden in 2-Tupel Form dargestellt,

$$(x_i, y_i) \in \Re^d \times N \qquad N \in \{1, ...K\}$$

wobei y_i die richtige Klasse des Musters M_i ist.
Die in dieser Arbeit vorgestellten Verfahren arbeiten nach dem Prinzip des „Überwachten Lernens“, d. h. die Zielklassen y_i sind bekannt. Im allgemeinen wird ein Klassifikator wie folgt konstruiert:

- Generierung einer spezifischen Diskriminantenfunktion $g : \Re^d \mapsto \{1, .., K\}$ mittels Trainingsdaten
- Evaluation mittels Testdaten, d. h. es wird die Fehlerrate des Klassifikationsverfahrens bestimmt

Der Klassifikator g ordnet einen beliebigen Merkmalsvektor x_i in folgender Weise zu einer Klasse $\hat{k}_i$:

$$g(x_i) = \hat{k}_i \qquad \hat{k}_i \in \{1, ..., K\}$$

Wir nehmen an, daß die richtige Klasse k_i zu einem Muster M_i bekannt sei. Dann nennt man den Fall $\hat{k}_i \neq k_i$ einen Fehler. Die *Fehlerrate* eines Klassifikators ist definiert als der Quotient zwischen der Summe der Gesamtfehler und der Anzahl der Testmuster.

Beispiele für solche Klassifikatoren sind Abstandsklassifikatoren (etwa k-Nearest-Neighbour), künstliche neuronale Netze und Mischverteilungen im Kontext der Bayesschen Entscheiungsregel.

Es stellt sich die Frage wozu man Klassifikationsverfahren in der Medizin braucht. Der Motivationsaspekt liegt darin, daß die Einsatzmöglichkeiten im medizinischen Bereich sehr groß sind. Es wäre möglich z. B. Gewebeuntersuchungen mittels Klassifikatoren durchzuführen. Der Klassifikator entscheidet dann, ob das untersuchte Gewebe gesund oder pathologisch ist. Generell kann man Diagnose mit Klassifikation gleichsetzen, denn jede Art von Untersuchung führt auf eine Klassifikation zurück. Z. B. ordnet ein Arzt einer Person anhand bestimmter Symptome eine Krankheit zu. Klassifikationsverfahren eignen sich also im medizinischen Bereich insbesondere zur Diagnoseunterstützung. Generell werden hohe Anforderungen an ein Klassifikationsverfahren gestellt, insbesondere im medizinischen Bereich. Ein Klassifikator soll minimale Fehlerraten erzielen und universell Einsetzbar sein. Im folgenden soll ein spezielles Klassifikationsverfahren vorgestellt werden, nämlich die Support-Vektor-Maschine.

2 Support-Vektoren

Die Support-Vektor-Maschine (SVM) ist ein neues Verfahren zur Lösung von Klassifikationsaufgaben. Vapnik [1][2] entwickelte diese Methode für die Lösung eines 2-Klassen Problems. Soll ein K-Klassen Problem gelöst werden, so erreicht man dies durch eine Umformulierung auf K 2-Klassen Probleme. Das bedeutet, daß eine Klassifikationsfunktion die folgende Form bekommt,

$$g: \Re^d \mapsto \{1,2\},$$

Ein Beispiel, um den wesentlichen Unterschied zwischen der Support-Vektor-Maschine und einem K-Klassen Klassifikator zu sehen, ist z.B die Ziffernerkennung, auf die später noch genauer eingegangen wird. Möchte man handschrifliche Zahlen (Digits) mit einem Rechner automatisch klassifizieren und hat dazu einen K-Klassen Klassifikator, so erhält man zu einem betrachteten Testvektor x_i, als Ergebnis unmittelbar eine Klasse $\hat{k}_i \in \{0,...,9\}$.
Der Support-Vektor-Algorithmus muß dieses Mehrklassenproblem zunächst auf zehn 2-Klassen Probleme abbilden. Die Lösung zu diesem Problem liegt nun darin, daß jeder dieser Klassifikatoren die Entscheidung 'k' oder 'nicht k' trift, wobei diese z.B mit sogenannten Scores gewichtet werden können. Man entscheidet sich dann für die Klasse 'k' deren Score maximal ist.
Als Beispiel würde die Abbildung, ob eine Ziffer als 0 oder als nicht 0 klassifiziert wird, wie folgt aussehen,

$$g: \Re^d \longmapsto \{\underbrace{0}_{k}, \underbrace{\{1,..,9\}}_{\neg k}\}$$

Es erscheint zwar im ersten Moment, als wäre die SVM aufwendiger, allerdings liegt die Priorität bei Klassifikatoren nicht im Aufwand, sondern in der Qualität. Das Ziel besteht also darin, die Fehlerrate zu minimieren, die benötigte Rechenzeit ist dabei i.d.R. nebensächlich. Nachdem nun die Grundproblematik der Mustererkennung und der Klassifikation beschrieben worden ist, wird im folgenden die Funktionsweise der Support-Vektor-Maschine erklärt.

2.1 Die Grundidee einer Support-Vektor-Maschine

Die Support-Vektor-Maschine ist ein Klassifikationsverfahren, welches man zu einem gegebenen Klassifikationsproblem (über die gegebenen Trainingsdaten der zu klassifizierenden Objekte) konstruiert. Die Grundidee bei der Konstruktion eines SV-basierten Klassifikators liegt darin, die Trainingsdaten durch eine lineare Entscheidungsfläche zu trennen. Eine lineare Entscheidungsfläche ist im eindimensionalem Raum ein Punkt (also ein Schwellwert), im zweidimensionalem eine Gerade und allgemein im hochdimensionalem Raum eine Hyperebene. Die Hyperebene ist also eine Entscheidungsfläche die den hochdimensionalen Raum in zwei Hälften teilt. In der Trainingsphase wird die Entscheidungsfunktion (1) mittels der Trainingsdaten (2) trainiert. Diese klassifiziert dann ein Testdatum t_i ($t_i \in \Re^d$ ist nicht in der Menge der Trainingsdaten enthalten) in eine der beiden möglichen Klassen, d. h. es wird geprüft, auf welcher Seite der Hyperebene sich das Testdatum befindet.

$$g: \Re^d \mapsto \{-1,1\} \tag{1}$$

$$(x_1,y_1),...,(x_l,y_l) \qquad x_i \in \Re^d \quad y_i \in \{-1,1\} \tag{2}$$

Man wird sehen, daß für die Konstruktion einer linearen Entscheidungsfiche eine Teilmenge der Trainingsdaten ausreichen wird, die man mit „Support-Vektoren“ bezeichnet.
Es ergibt sich jedoch ein Problem, welches darin besteht, daß Trainingsdaten nicht immer linear separabel sind. Eine Idee zur Lösung dieses Problems liegt darin, daß man die Trainingsdaten aus dem Eingaberaum Raum in den hochdimensionalen Merkmalsraum transformiert, denn im hochdimensionalem Raum ist die Wahrscheinlichkeit linearer Separabilität wesentlich größer. Abbildung 1 zeigt ein triviales Beispiel, in dem eine Transformation vom $\Re^1$ in den $\Re^2$ schon ausreicht, um die Trainingsdaten linear zu separieren. Der Merkmalsraum ist im Normalfall sehr viel größer als der Eingaberaum.

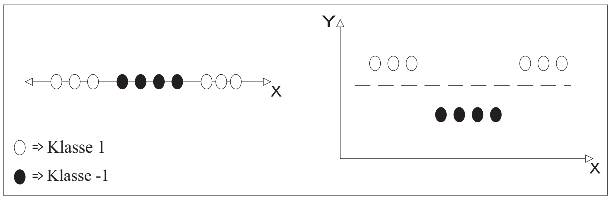

Abbildung 1: Zur Klassentrennung geeignete Projektion $\phi : \Re^1 \mapsto \Re^2$

Allgemein werden die Trainingsdaten x_i, durch eine nichtlineare Abbildung ϕ, in den hochdimensonalen Merkmalsraum transformiert.

$$x \in \Re^d \quad \longmapsto \quad \phi(x) \in \Re^D, \qquad \text{mit} \;\; D >> d \tag{3}$$

Der Ablauf für die Konstruktion einer Hyperbene im hochdimensionalem Merkmalsraum ist folgender,

- a-priori Auswahl einer geeigneten Abbildung ϕ
- Bestimmung der optimalen Hyperebene

Im hochdimensionalem Merkmalsraum gibt es nicht nur eine mögliche Entscheidungsfläche, sondern es gibt eine Menge von Hyperebenen, die die Trainingsdaten in zwei Klassen trennen. Ausserdem sind Berechnungen im hochdimensionalen Raum wesentlich aufwendiger als die im Eingaberaum.

Es gibt deshalb zwei Grundlegende Probleme bei der Konstruktion einer Entscheidungsfläche, nämlich ein Konzeptionelles und ein technisches Problem:

- Wie bestimmt man eine lineare Entscheidungsfläche im hochdimensionalem Merkmalsraum und welche von diesen ist die optimale Hyperbene ?
- Curse-of-Dimensionality ?
  (Die Komplexität der Berechnung steigt mit der Dimension des Merkmalsraumes)

Der Algorithmus der Support-Vektor-Maschine ist in der Lage diese Probleme zu lösen. Das konzeptionelle Problem ist 1982 gelöst worden und es wird im nächsten Abschnitt gezeigt wie man eine zunächst eine lineare Entscheidungsfläche für den Fall der fehlerfrei separierbaren Trainingsdaten konstruiert. In Abschnitt 2.3 wird die SVM auf den eher praxisnahen Fall der nicht fehlerfrei separierbaren Trainingsdaten erweitert (Vapnik,1995). Der nächste Abschnitt widmet sich zunächst dem konzeptionellem Problem, um die beiden Probleme getrennt zu behandeln. Die Konstruktion einer lineare Entscheidungsfläche wird also erstmal im Eingaberaum durchgeführt, um dann in Abschnitt 2.4 auf das technische Problem einzugehen.

2.2 Fehlerfrei separierbare Trainingsdaten

In diesem Abschnitt wird der ideale Fall behandelt, d. h. es wird gezeigt wie man eine lineare Entscheidungsfläche für fehlerfrei separierbare Trainingsdaten im Eingaberaum konstruiert.

Seien die Trainingsdaten, die mit ihren Zielklassen gelabelt seien, gegeben durch,

$$(x_1, y_1), ..., (x_l, y_l) \qquad\qquad x_i \in \Re^d, \quad y_i \in \{-1, 1\} \tag{4}$$

Definition:

Eine Menge von Trainingsdaten ist linear separabel genau dann, wenn ein Vektor $w \in \Re^d$ und ein Skalar $b \in \Re$ existiert, so daß für alle Trainingsdaten $x_i \in \Re^d$ gilt

$$w \cdot x_i + b \;\; \geq \;\; 1 \qquad\qquad \text{wenn } y_i = 1 \tag{5}$$

$$w \cdot x_i + b \;\; \leq \;\; -1 \qquad\qquad \text{wenn } y_i = -1 \tag{6}$$

Die Ungleichungen (5) und (6) können in einer Formel ausgedrückt werden als

$$y_i(w \cdot x_i + b) \;\; \geq \;\; 1 \qquad\qquad \text{für } \; i = 1, ..., l \tag{7}$$

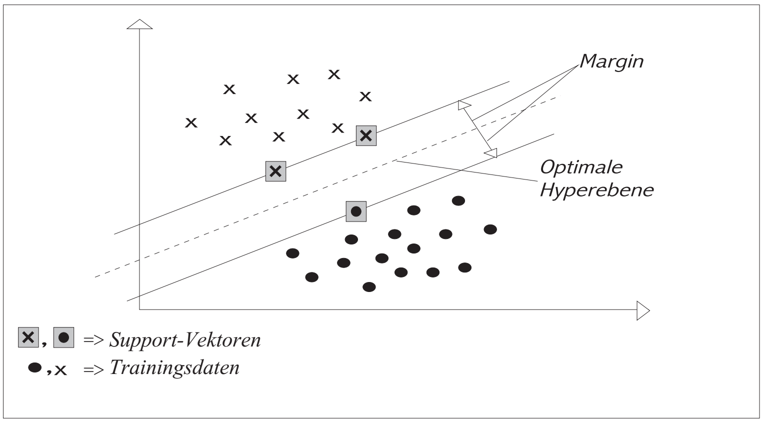

Abbildung 2: Optimale Hyperbene & Support-Vektoren

Definition: *Optimale Hyperebene*

> *Die optimale Hyperebene ist eine lineare Entscheidungsfläche mit maximalem Abstand der beiden Klassen $k = 1$ und $k = -1$.*

Die optimale Hyperebene ist in Normalenform gegeben durch,

$$w_0 \cdot x + b_0 = 0 \tag{8}$$

mit:

- w_0 : Normalenvektor im $\Re^d$
- x : Trainingsdatum im $\Re^d$
- b_0 : „Bias“

Es zeigt sich, daß eine Teilmenge aus den Trainingsdaten zur Konstruktion der Hyperebene ausreicht. Diese Teilmenge ist die Menge der sogenannten Support-Vektoren (SV). Support-Vektoren sind genau die Trainingsdaten, die der zu bestimmenden Hyperebene am nächsten liegen (siehe Abbildung 2).

$$I_{SV} = \{x_{i_1}, ..., x_{i_j}\} \subseteq \{x_1, ..., x_l\} \qquad \text{mit} \quad x_i \in \Re^d \tag{9}$$

Eine Ebene wird durch den Normalenvektor eindeutig bis auf den Bias b beschrieben, d. h. um die gesuchte Hyperbene zu finden, ist der zugehörige Normalenvektor zu bestimmen. Es zeigt sich, daß der Normalenvektor durch die Support-Vektoren bestimmt wird, d. h. es gilt;

$$w_0 = \sum_{i \in I_{SV}} \lambda_i x_i \tag{10}$$

mit:

- $\lambda_i \in \Re$: skalare Gewichte
- I_{SV} : Vektoren x_i mit $\lambda_i > 0$ und das sind genau die Support-Vektoren

Der Beweis dieser Behauptung wird im folgenden geführt werden. Ist der Normalenvektor w_0 bestimmt, dann kann der Bias b auch berechnet werden, womit dann die Klassifikationsfunktion (11) gegeben ist.

$$g(t) = \text{sign}\,(w_0 \cdot t + b) \qquad w_0, t \in \Re^d, \;\; b_0 \in \Re \tag{11}$$

Es ist $w_0 \cdot t$ das Skalarprodukt im $\Re^d$, und g entscheidet für das Testdatum t, ob es sich in der Klasse $k = 1$ oder $k = -1$ befindet. Im folgenden soll die formale Herleitung gezeigt werden.
Der Normalenvektor w wird über den Abstand der Trainingsdaten bestimmt. Die Trainingsdaten werden auf den Normalenvektor projiziert (Abbildung 3), womit sich der maximale Abstand der beiden Klassen

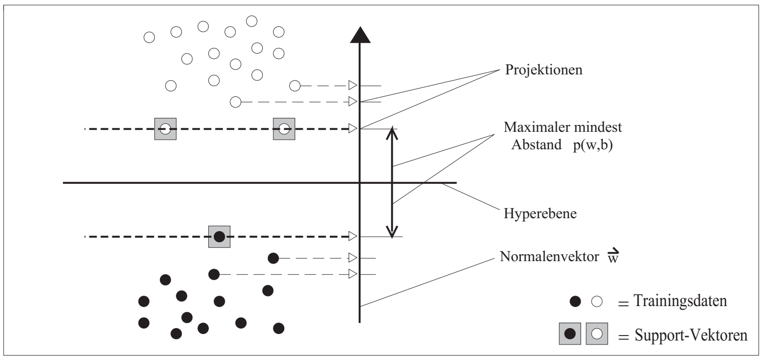

Abbildung 3: Projektion auf Normalenvektor

ergibt. Durch diese Projektion wird die Richtung des Normalenvektors w (und damit die Hyperebene bis auf den Bias b) festgelegt. Dieser Ansatz wird durch die folgende Abstandsfunktion (12) formalisiert.

$$\rho(w,b) = \min_{\{x:y=1\}} \frac{(w \cdot x + b)}{||w||} - \max_{\{x:y=-1\}} \frac{(w \cdot x + b)}{||w||} \qquad x, w \in \Re^d, \quad b \in \Re \tag{12}$$

Gesucht ist jetzt das Minimum der Klasse $y = 1$ und das Maximum der Klasse $y = -1$. Es wird sich zeigen, daß die gesuchten Extrema genau die Support-Vektoren der jeweiligen Klasse sind. Seien die Extrema gegeben durch (folgt aus der Definition (5,6) !):

$$1 + \alpha \;=\; \{ \; \min(w \cdot x_i + b) \;|\; i = 1, ..., l \; \} \tag{13}$$

$$-1 - \beta \;=\; \{ \; \max(w \cdot x_i + b) \;|\; i = 1, ..., l \; \} \tag{14}$$

Setzt man die Extremwerte (13) und (14) in die Abstandsfunktion (12) ein, so ergibt sich

$$\begin{aligned} \rho(w,b) &= \frac{(1+\alpha)}{||w||} - \frac{-(1+\beta)}{||w||} && (15) \\ &= \frac{1+\alpha+1+\beta}{||w||} \\ &= \frac{2+\alpha+\beta}{||w||} \\ &= \frac{2+\alpha+\beta}{\sqrt{w \cdot w}} && (16) \end{aligned}$$

Es wird sich zeigen, daß für Support-Vektoren der beiden Klassen $\alpha = \beta = 0$ gilt. Um den größtmöglichen Abstand zwischen den beiden Klassen zu bestimmen, ist die Abstandsfunktion $\rho(w, b)$ zu maximieren. Dabei gilt folgende Äquivalenz:

$$\text{Max !} \;\; \rho(w,b) \iff \text{Min !} \;\; (w \cdot w) \tag{17}$$

Um den gesuchten Normalenvektor w zu bestimmen, ist das folgende Optimierungsproblem zu lösen.
Minimiere

$$\Phi(w) \;=\; \frac{1}{2} \, (w \cdot w) \tag{18}$$

unter den Nebenbedingungen

$$y_i(w \cdot x_i + b) - 1 \geq 0 \tag{19}$$

In der Gleichung (18) ist zu bemerken, das der Faktor 1/2 keinen Einfluß auf die Optimierungsaufgabe (bezüglich w_0) hat. Es hat jedoch rechnerische Vorteile, wie sich im folgenden zeigen wird. Die in den Gleichungen (18) und (19) geforderte Optimierungaufgabe entspricht einem nichtlinearen Optimierungsproblem:

Minimiere nicht lineare Funktion $\Phi(w)$ *unter den Nebenbedingungen* $g_i(w) \geq 0$

Das Verfahren, womit ein nichtlineares Optimierungsproblem gelöst wird, ist die Lagrange'sche Multiplikatorenmethode [4]. Die Lagrangefunktion wird im allgemeinen wie folgt konstruiert,

$$L(w,\Lambda) = \Phi(w) - \sum_{i=1}^{l} \lambda_i g_i(w) \qquad w \in \Re^d,\ \ \Lambda \in \Re^l_+ \tag{20}$$

mit:

- $\Phi(w)$: nichtlineare Optimierungsfunktion
- $g_i(w)$: Nebenbedingungen
- $\Lambda = (\lambda_1, ..., \lambda_l)$: Lagrange Multiplikatoren

Für die Support-Vektor-Maschine ergibt sich nach der Optimierungsfunktion (18) und den Nebenbedingungen (19) folgende Lagrangefunktion:

$$L(w,b,\Lambda) \ = \ \frac{1}{2} w \cdot w \ - \ \sum_{i=1}^{l} \lambda_i \left[y_i(x_i \cdot w + b) - 1 \right] \tag{21}$$

mit:

- $w, x_i \in \Re^d, \quad \lambda_i \in \Re_+, \quad b \in \Re, \quad y_i \in \{1, -1\}$

Es ist bekannt, daß die Lösung des Optimierungsproblems auf eine Sattelpunktbestimmung hinausläuft. Um den Sattelpunkt der Lagrangefunktion (21) zu bestimmen, ist das Minimum bezüglich w und b und das Maximum bezüglich $\Lambda = (\lambda_1, ..., \lambda_l)$ zu bestimmen. Zunächst soll das Minimum berechnet werden und dazu werden für die Lagrangefunktion folgende Differentialquotienten nach w und b gebildet.

$$\frac{\partial L(w,b,\Lambda)}{\partial w} |_{w=w_0} \ = \ \left(w_0 - \sum_{i=1}^{l} \lambda_i y_i x_i \right) = 0 \tag{22}$$

$$\frac{\partial L(w,b,\Lambda)}{\partial b} |_{b=b_0} \ = \ \sum_{\lambda_i} y_i \lambda_i = 0 \tag{23}$$

Die Gleichungen (22) und (23) beschreiben Eigenschaften der gesuchten Hyperebene. Aus der Gleichung (22) folgt, daß der Normalenvektor w_0, der die optimale Hyperbene bestimmt, aus einer Linearkombination der Trainingsdaten bestimmt wird. Dabei tragen nur die Trainingsdaten x_i zur Summe bei, deren Faktor λ_i größer Null ist:

$$w_0 = \sum_{i=1}^{l} \lambda_i y_i x_i \tag{24}$$

Setzt man die Gleichungen (22) und (23) in die Lagrangefunktion (21) ein, so folgt,

$$\begin{aligned}
L(w_0, b_0, \Lambda) &= \frac{1}{2} w_0 \cdot w_0 - \sum_{i=1}^{l} \lambda_i [y_i(x_i \cdot w_0 + b_0) - 1] && (25) \\
&= \frac{1}{2} w_0 \cdot w_0 - \sum_{i=1}^{l} \lambda_i [y_i \cdot x_i \cdot w_0 + y_i \cdot b_0 - 1] \\
&= \frac{1}{2} w_0 \cdot w_0 - \left(\sum_{i=1}^{l} \lambda_i y_i x_i \cdot w_0 \ + \ \sum_{i=1}^{l} \lambda_i y_i b_0 \ - \ \sum_{i=1}^{l} \lambda_i \right) \\
&= \frac{1}{2} w_0 \cdot w_0 - \left(\underbrace{\sum_{i=1}^{l} \lambda_i y_i x_i}_{= w_0 \ (24)} \cdot w_0 \ + \ b_0 \cdot \underbrace{\sum_{i=1}^{l} \lambda_i y_i}_{= 0 \ (23)} \ - \ \sum_{i=1}^{l} \lambda_i \right) \\
&= \frac{1}{2} w_0 \cdot w_0 - w_0 \cdot w_0 + \sum_{i=1}^{l} \lambda_i \\
&= \sum_{i=1}^{l} \lambda_i - \frac{1}{2} w_0 \cdot w_0 && (26)
\end{aligned}$$

Nachdem das Minimierungsproblem durch Differenzieren gelöst worden ist, muß als nächstes noch das Maximum bestimmt werden, welches nur noch von den λ_i abhängt. Nach Gleichung (26) folgt,

$$W(\Lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} w_0 \cdot w_0 \tag{27}$$

Substituiert man w_0 wieder in die Form der Linearkombination, so ergibt sich:

$$W(\Lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \lambda_i \lambda_j y_i y_j x_i \cdot x_j \tag{28}$$

Formal kann Gleichung (28) in folgende Vektor-Notation umgeschrieben werden

$$W(\Lambda) = \Lambda^T 1 - \frac{1}{2} \Lambda^T D \Lambda \tag{29}$$

mit:

- $\Lambda = (\lambda_1, ..., \lambda_l)$: Skalare λ_i
- l : l-dimensionaler Einheitsvektor
- $D_{ij} = y_i y_j x_i \cdot x_j$: Symmetrische $l \times l$-Matrix

Um den gesuchten Sattelpunkt zu bestimmen, ist die Gleichung (29) zu maximieren, d. h. :

$$\text{Max !} \quad W(\Lambda) = \Lambda^T 1 - \frac{1}{2} \Lambda^T D \Lambda \tag{30}$$

unter den folgenden Nebenbedingungen

$$\Lambda \geq 0 \tag{31}$$

$$\Lambda^T Y = 0 \tag{32}$$

mit:

- $Y = (y_1, ..., y_l)$: $y_i \in \{1, -1\}$
- $1 = (1_1, ..., 1_l)$: l-dimensionaler Einheitsvektor

Es ist zu bemerken, daß Bedingung (32) aus der Forderung (23) folgt.
Es ist in (30) ein quadratisches Optimierungsproblem, welches unter ihren Nebenbedingungen gelöst werden muß. Ist das quadratische Optimierungsproblem gelöst, so ist der Vektor Λ_0 bestimmt, der die Gleichung (30) maximiert.

$$\Longrightarrow \qquad \Lambda_0 = (\lambda_1^0, ..., \lambda_l^0) \tag{33}$$

Der Normalenvektor w_0 ist damit bestimmt, womit auch die Konstante b_0 wie folgt berechnet werden kann.

$$b_0 = \frac{1}{2}[(w_0 \cdot x^*(1)) + (w_0 \cdot x^*(-1))] \tag{34}$$

mit:

- $x^*(1)$: beliebiger Support-Vektor der zur ersten Klasse gehört
- $x^*(-1)$: beliebiger Support-Vektor der zur zweiten Klasse gehört

Der Sattelpunkt (w_0, b_0, Λ_0) ist damit bestimmt und es gilt für alle Trainingsdaten:

$$\lambda_i [y_i (x_i \cdot w_0 + b_0) - 1] = 0, \qquad i = 1, ..., l. \tag{35}$$

Aus der Gleichung (35) folgt, daß für $\lambda_i > 0$ gilt,

$$y_i (x_i \cdot w_0 + b_0) - 1 = 0. \tag{36}$$

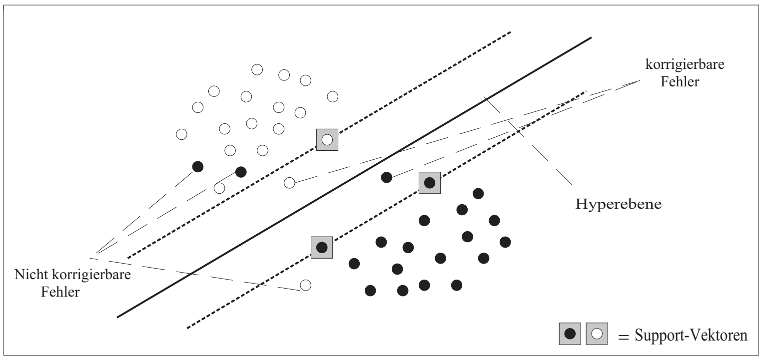

Abbildung 4: Fehler in den Trainingsdaten

Es ist damit gezeigt, daß nur die Trainingsdaten mit $\lambda_i \neq 0$ zu dem Normalenvektor beitragen und alle die Trainingsdaten, die folgende Gleichung erfüllen werden als Support-Vektoren bezeichnet.

$$y_i(x_i \cdot w_0 + b_0) = 1 \tag{37}$$

Aus der Gleichung (37) folgt, daß die in (12) gesuchten Extrema, durch die Support-Vektoren bestimmt sind und das der in (16) bestimmte Abstand zwischen den beiden Klassen mit $\alpha = \beta = 0$ berechnet werden kann. Damit ist gezeigt worden, daß der Normalenvektor w aus einer Linearkombination der Support-Vektoren ermittelt wird. Hat man also ein Klassifikationsproblem gegeben, so bleibt nur noch das quadratische Optimierungsproblem (30), welches man lösen muß, um die Klassifikationsfunktion (11) zu ermitteln. Es sei hier noch erwähnt, das das quadratische Optimierungsproblem durch Standardalgorithmen gelöst werden kann, z. B. durch das Verfahren von Wolfe [6], welches nach dem Prinzip des Simplex-Algorithmus arbeitet. Im nächsten Abschnitt wird die SVM auf den realistischen Fall der nicht fehlerfrei separierbaren Trainingsdaten erweitert.

2.3 Nicht-fehlerfrei separierbare Trainingsdaten

In diesem Abschnitt wird die Erweiterung der SVM für nicht fehlerfrei separable Trainingsdaten vorgestellt, welches auch der wahrscheinlichere Fall in der Praxis ist [2]. Die Grundidee bei der Behandlung der nicht fehlerfrei separablen Trainingsdaten liegt darin, diesen Fall zurückzuführen auf den Fall der fehlerfrei separablen Trainingsdaten. Die Erweiterung des Ansatzes der fehlerfrei-separierbaren Trainingsdaten auf nicht-fehlerfrei separierbare Trainingsdaten wird durch die Einführung sogennanter Slack-Variablen (Korrekturwerte) $\xi_i \geq 0$ erreicht, die gewissermaßen die nichterfüllte Nebenbedingung (7) entsprechend korrigieren. Es gibt jedoch Fehler die nicht korrigiert werden können, da ihre Abweichung zu groß sind. Das Ziel besteht also darin, die Anzahl der Fehler zu minimieren, deshalb werden die Abweichungen wie folgt in einer Funktion summiert.

$$\Phi(\xi) = \sum_{i=1}^{l} \xi_i \tag{38}$$

Um die Anzahl der Fehler zu minimieren, ist die Funktion (38) unter den folgenden Nebenbedingungen zu minimieren.

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \qquad i = 1, ..., l \tag{39}$$

$$\xi_i \geq 0, \qquad i = 1, ..., l \tag{40}$$

Ist die Funktion (38) unter den Nebenbedingungen minimiert, dann gibt es eine Teilmenge von Trainingsdaten die nicht korrigiert werden kann.

$$(x_{i_1}, y_{i_1}), ..., (x_{i_k}, y_{i_k}) \tag{41}$$

Nimmt man diese Teilmenge aus der Menge der Trainingsdaten heraus, so kann der Rest der Trainingsdaten wie im Fall der fehlerfrei separierbaren Trainingsdaten durch eine optimale Hyperebene getrennt werden. Diese Idee kann formal wie folgt ausgedrückt werden: Minimiere die Funktion

$$\frac{1}{2}w^2 + C \sum_{i=1}^{l} \xi_i \tag{42}$$

unter den Nebenbedingungen

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \qquad i = 1, ..., l \tag{43}$$

$$\xi_i \geq 0, \qquad i = 1, ..., l \tag{44}$$

mit:

- $C \in \Re$: Gewichtsfaktor
- $\xi_i \in \Re_+, \; w, x_i \in \Re^d, \; b \in \Re, \; y_i \in \{1, -1\}$

Die zu minimierende Funktion Φ ist, im Vergleich zu den fehlerfrei separierbaren Trainingsdaten, um ein weiteres Ziel erweitert. Zum einen soll die optimale Hyperebene mit einem maximalen Abstand bestimmt werden, dazu ist der erste Teil der Funktion (42) zu minimieren, und gleichzeitig soll die Anzahl der Fehler minimiert werden, wozu der zweite Teil der Funktion (42) minimiert werden muß. Die Minimierung der Gesamtfunktion (42) beinhaltet damit ein „Trade-Off“ zwischen der Maximierung des Abstandes der beiden Klassen und der Minimierung der Anzahl der Fehler. Möchte man erreichen, daß möglichst wenig Fehler gemacht werden, so muß ein relativ großes C gewählt werden, wodurch aber dann der Abstand der beiden Klassen vernachlässigt wird. Es ist an dieser Stelle zu bemerken, daß es wünschenswert wäre direkt die Fehler (also 38) zu minimieren, jedoch führt dieses Vorgehen zu einem NP-vollständigen Problem [1][2]. Der Vorteil bei der Minimierung der Funktion (42) liegt darin, daß dieses Optimierungsproblem fast identisch zu berechnen ist, wie die zu optimierende Funktion in (18). Die Lagrange-Funktion hat im Falle der nicht-fehlerfrei separierbaren Trainingsdaten, unter der nicht-linearen Funktion (42) und den Nebenbedingungen (43,44), folgende Form:

$$L(w, \xi, b, \Lambda, R) = \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \lambda_i [y_i(x_i \cdot w + b) - 1 + \xi_i] - \sum_{i=1}^{l} r_i \xi_i \tag{45}$$

mit:

- $w, x_i \in \Re^d, \; \xi_i, C \in \Re_+, \; b \in \Re$
- $\lambda_i, r_i \in \Re_+$: Lagrange-Multiplikatoren [4]

Es ist Analog, wie im Falle der fehlerfrei separablen Trainingsdaten, der Normalenvektor w_0 gesucht, und dazu muß der Sattelpunkt der Lagrange-Funktion bestimmt werden. Um den Sattelpunkt der Funktion zu bestimmen, müssen wir ein Minimum hinsichtlich der Variablen w_i, b und ξ_i, und ein Maximum bzgl. der Variablen λ_i und r_i finden. Zunächst werden die notwendigen Ableitungen bestimmt:

$$\frac{\partial L(w, \xi, b, \Lambda, R)}{\partial w} |_{w=w_0} = w_0 - \sum_{i=1}^{l} \lambda_i y_i x_i = 0 \tag{46}$$

$$\frac{\partial L(w, \xi, b, \Lambda, R)}{\partial b} |_{b=b_0} = \sum_{i=1}^{l} \lambda_i y_i = 0 \tag{47}$$

$$\frac{\partial L(w, \xi, b, \Lambda, R)}{\partial \xi_i} |_{\xi_i=\xi_i^0} = C - \lambda_i - r_i = 0 \tag{48}$$

Substituiert man nun die Ausdrücke für w_0 (46), b_0 (47) und C (49) in die Lagrange-Funktion, dann ergibt sich eine zu (28) identische Gleichung, da sich die Parameter ξ_i und r_i auflösen.

$$W(\Lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \lambda_i \lambda_j y_i y_j x_i \cdot x_j \tag{49}$$

Somit ergibt sich im Grunde ein zu Gleichung (30) fast identisches quadratisches Optimierungsproblem, daß nur um eine Nebenbedingung erweitert ist.

$$\text{Max !} \quad W(\Lambda) = \Lambda^T 1 - \frac{1}{2} \Lambda^T D \Lambda \tag{50}$$

unter den folgenden Nebenbedingungen

$$\Lambda \geq 0 \tag{51}$$
$$\Lambda^T Y = 0 \tag{52}$$
$$\Lambda \leq C1 \tag{53}$$

mit:

- $Y = (y_1, ..., y_l) : y_i \in \{1, -1\}$
- $1 = (1_1, ..., 1_l)$: l-dimensionaler Einheitsvektor

Aus der Bedingung (53) folgt, daß jedes λ_i kleiner oder Gleich C sein muß. Ist das Maximum der Funktion (50) berechnet, so ist der gesuchte Vektor Λ_0 bestimmt, womit dann der Normalenvektor w_0 und demzufolge der Bias b_0 berechnet werden können. Die gesuchte Hyperebene wird, wie auch bei den fehlerfrei separierebaren Trainingdaten, als Superposition der Support-Vektoren berechnet. Im Falle der nicht-fehlerfrei separierbaren Trainingsdaten ist auch festgelegt, welche Trainingsdaten nicht korrigiert werden können. Diese sind also genau die, die mit der berechneten Hyperebene und damit der Klassifikationsfunktion (11) falsch klassifiziert werden.

Damit ist das konzeptionelle Problem der Support-Vektor-Maschine gelöst. Der nächste Abschnitt widmet sich dem technischen Problem, wo gezeigt wird, wie die Berechnung der Skalarprodukte im hochdimensionalem Raum durchgeführt wird und wie diese Berechnungen effizient durchgeführt werden können.

2.4 Berechnung der Skalarprodukte

In Abschnitt 2.1 wurde beschrieben, daß Trainingsdaten nicht immer linear separierbar sind. Die Idee zur Lösung dieses Problems, liegt darin, daß die Trainingsdaten aus dem Eingaberaum in den hochdimensionalen Merkmalsraum transformiert werden, denn im hochdimensionalem Raum ist die Wahrscheinlichleit linearer Separabilität wesentlich größer. Bisher wurde die Konstruktion der trennenden Hyperebene nur im Eingaberaum vorgestellt, um das Prinzip der SVM besser verstehen zu können. Um eine Hyperebene im hochdimensionalen Raum zu konstruieren, müssen die Trainingsdaten zunächst durch eine geeignete Abbildung ϕ nicht-linear in den hochdimensionalen Raum transformiert werden. Allgemein sieht die Transformation (3) wie folgt aus:

$$\phi : \Re^d \rightarrow \Re^D \qquad \text{mit } D >> d \tag{54}$$

Die Bestimmung der Hyperebene ist prinzipiell wie im Eingaberaum, d. h. die notwendigen Schritte für die Konstruktion einer linearen Entscheidungsfläche werden genauso wir vorher durchgeführt, jedoch nun auf den transformierten Trainingsdaten. Die Klassifikationsfunktion (11) bekommt dadurch die folgende Form:

$$g(t) = \text{sign}(w \cdot \phi(t) + b) \qquad t \in \Re^d, \;\; w \in \Re^D, \;\; b \in \Re \tag{55}$$

Die Berechnung des Normalenvektors w erfolgt damit auch im Merkmalsraum, deshalb ist die Superpositionierung (Linearkombination) des Normalenvektors auch wie folgt modifiziert.

$$w_0 = \sum_{i=1}^{l} \lambda_i y_i \phi(x_i) \tag{56}$$

Wird der im Merkmalsraum berechnete Normalenvektor (56) in die Klassifikationsfunktion (55) eingesetzt, so ergibt sich für den Klassifikator:

$$g(t) = \text{sign}\left(\sum_{i=1}^{l} \lambda_i \, y_i \, \phi(t) \cdot \phi(x_i) + b_0\right) \tag{57}$$

mit:

- $t \in \Re^d$: Testdatum
- $\phi(t) \cdot \phi(x_i)$: Skalarprodukt im Merkmalsraum

Es ist zu bemerken, daß die Berechnung einer Entscheidung durch die Klassifikationsfunktion nur noch von dem Skalarprodukt abhängt, aber gleichzeitig kann diese Berechnung sehr komplex werden. Man wird in den Beispielanwendungen sehen, daß die Dimension des Merkmalsraumes sehr groß werden kann und Dimensionen im Bereich von 10^{12} nicht selten sind. An dieser Stelle wird das schon in der Einleitung erwähnte technische Problem deutlich, denn die Komplexität bei der Berechnung eines Skalarproduktes, steigt mit der Dimension des Merkmalsraumes. Dieses technische Problem ist im Jahre 1992 gelöst worden und die eigentliche Idee lag darin, die Reihenfolge der Operationen zu vertauschen und damit die Skalarprodukte schon im Eingaberaum zu berechnen. Dazu wurden die Eigenschaften des Hilbert-Raumes ausgenutzt [4][5].

Definition: *Im Hilbert-Raum ist das Skalarprodukt definiert durch die folgende Äquivalenz*

$$\phi(u) \cdot \phi(v) \equiv K(u,v) \qquad\qquad u,v \in \Re^d \tag{58}$$

mit:

- $K(u,v)$: K geeignete Kernfunktion

Nach dem Hilbert-Schmidt Theorem berechnet eine Kernfunktion $K(u,v)$ das Skalarprodukt $(\phi(u) \cdot \phi(v))$ im Merkmalsraum genau dann, wenn sie symmetrisch ist und wenn sie die Bedingungen des Mercer's Theorems erfüllt:

Theorem von Mercer:
Damit eine symmetrische Kernfunktion $K(u,v) \in L_2$ das folgende Skalarprodukt im Merkmalsraum berechnet,

$$K(u,v) = \sum_{i=1}^{l} \lambda_i \phi(u) \cdot \phi(v) \tag{59}$$

wobei die Koeffizienten $\lambda_i > 0$ positiv sind, ist es notwendig und hinreichend, daß die folgende Bedingung

$$\int\int K(u,v) g(u) g(v)\, du\, dv \; > \; 0 \tag{60}$$

durch alle $g \neq 0$ erfüllt wird, für die gilt:

$$\int g^2(u) \mathrm{d}u \; < \; \infty \tag{61}$$

Ersetzt man das Skalarprodukt durch eine geeignete Kernfunktion,die die Bedingungen von Mercer erfüllt, so ergibt sich für die Klassifikationsfunktion die folgende Form:

$$g(t) = \mathrm{sign}\left(\sum_{i=1}^{l} \lambda_i y_i K(x_i, t)\right) \tag{62}$$

mit:

- $x_i \in Re^d$: Support-Vektor im Eingaberaum
- $t \in \Re^d$: Testdatum im Eingaberaum
- λ_i : Gewicht (im Merkmalsraum bestimmt)

Die Konstruktion einer linearen Entscheidungsfläche (Hyperebene) im Merkmalsraum bleibt im Prinzip gleich. Die Gewichte λ_i werden genauso wie vorher durch Lösung des quadratischen Optimierungsproblems bestimmt. Eine Änderung ist lediglich in der Matrix D notwendig. Diese bekommt bei Konstruktion einer Hyperebene im Merkmalsraum im Vergleich zu (29) die folgende Form:

$$D_{ij} = y_i y_j K(x_i, x_j), \qquad i, j = 1, ..., l \tag{63}$$

Geeignete Kernfunktion die die Bedingungen des Mercer's Theorem erfüllen sind z. B. die Funktionen mit Polynomialen Kern: $K(x,t) = (x \cdot t)^d$ mit $x, t \in \Re^d$

Beispiel: Sei eine Abbildung ϕ gegeben durch

$$\begin{array}{rcl} \phi \;:\; \Re^2 & \longmapsto & \Re^3 \\ (x_1, x_2) & \longmapsto & (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2) \end{array}$$

dann gilt:

$$\begin{array}{rcl} (\phi(x) \cdot \phi(y)) & = & (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)(y_1^2, \sqrt{2}\, y_1 y_2, y_2^2)^T \\ & = & ((x_1, x_2)(y_1, y_2)^T)^2 \\ & = & (x \cdot y)^2 \\ & =: & K(x, y) \end{array}$$

Allgemein: $\quad K(x,y) = (x,y)^d \qquad x, y \in \Re^d, \;\; d \in N$

Es ist in diesem Abschnitt gezeigt worden, wie die komplexen Skalarprodukte im Merkmalsraum durch einfache Kernfunktionen ersetzt werden können. In Abschnitt 2.5.2 wird noch eine spezielle Kernfunktionen vorgestellt. Im nächsten Abschnitt wird auf die generellen Eigenschaften einer SVM eingegangen.

2.5 Generelle Eigenschaften von Support-Vektoren

2.5.1 Effizienz der Support-Vektor-Maschine

Um eine Entscheidungsfunktion durch eine Support-Vektor-Maschine zu konstruieren, muß das folgende quadratische Optimierungsproblem

$$W(\Lambda) = \Lambda^T 1 - \frac{1}{2} \lambda^T D \Lambda$$

unter den Nebenbedingungen,

$$0 \le \Lambda \le C\,\vec{1}$$
$$\Lambda^T Y = 0$$

gelöst werden, wobei die Matrix D wie unter (63) definiert ist. Das Optimierungsproblem wird durch die Trainingsdaten bestimmt und $K(u,v)$ ist die Funktion welche das komplexe Skalarprodukt im Eingaberaum berechnet. Das Optimierungsproblem kann effizient gelöst werden, denn die gesuchten Support-Vektoren werden aus einer endlichen Menge von Trainingsdaten bestimmt und die zu ermittelnde Entscheidungsfunktion ist eindeutig. Das Optimierungsproblem kann durch Standard-Algorithmen gelöst werden [6].

2.5.2 Universalität der Support-Vektor-Maschine

Die Support-Vektor-Maschine ist eine universelle Maschine, denn die Kernfunktion $K(u,v)$ kann einfach wie ein Modul ausgetauscht werden, wodurch man verschieden Lernmaschinen erzeugen kann. Dadurch ergeben sich andere Klassifikatoren (wie Polynomialklassifikatoren) als Spezialfall einer SVM. Einen Polynomialklassifikator läßt sich beispielsweise durch eine Kernfunktion der folgender Form realisieren,

$$K(u,v) = (u \cdot v)^d$$

womit Polynome vom verschiedenen Grad d behandelt werden können.
Ein weiterer Spezialfall einer Entscheidungsfunktion ist die Radial-Basis-Funktion:

$$f(x) \;=\; \text{sign}\left(\sum_{i=1}^{n} \alpha_i \,\exp\left\{\frac{|x-x_i|^2}{\sigma^2}\right\}\right)$$

mit:

- α_i : Gewichtsfaktor des i-ten Mittelwertes
- x_i : i-ter Mittelwert (Zentrum)
- σ_i : Varianz i-ten Mittelwertes

die mit folgender Kernfunktion implementiert werden kann,

$$K(x, x_i) \;=\; \exp\left\{-\frac{|x-x_i|^2}{\sigma^2}\right\}$$

In diesem Fall werden durch die Support-Vektor-Maschine zum einen die Zentren x_i und die Gewichte α_i bestimmt. In Abschnitt 3 wird, zum Vergleich der SVM mit radialer Kernfunktion, die klassische Form der Radial-Basis-Funktion kurz vorgestellt. Die Beispielanwendungen zeigen Ergebnisse von beiden Verfahren.
Es ist möglich spezielle Kernfunktionen durch a-priori Wissen zu konstruieren, womit die Qualität der SVM weiter verbessert werden kann. Support-Vektor-Maschinen sind für solche Erweiterungen sehr geeignete Lernmaschinen, denn allein durch Austauschen der Kernfunktion wird eine neue Entscheidungsfunktion generiert.

2.5.3 Generalisierungsfähigkeit

Die Generalisierungsfähigkeit einer Support-Vektor-Maschine hängt im allgemeinen von zwei Faktoren ab. Es gilt (Vapnik, 1982):

$$R_{strukt} \leq R_{emp} + \text{Konfidenzintervall} \tag{64}$$

Es gilt das strukturelle Risiko R_{strukt} zu minimieren, das neben dem empirischen Risiko R_{emp} noch ein Konfidenzintervall Φ (abhängig von der VC-Dimension h und der Größe des Lernsets L) berücksichtigt - wobei die beiden Größen gewissermaßen ein gegenläufiges Verhalten aufweisen: eine wachsende Mächtigkeit des verwendeten Klassifikators verringert das empirische Risiko, erhöht aber den Einfluß des Konfidenzterms (d. h. die Generalisierungsfähigkeit wird eventuell schlechter) und das gleiche gilt reziprok für den Stichprobenumfang. Gewöhnlich wird durch die Auswahl eines Klassifikators das Konfidenztintervall festgelegt und dann das empirische Risiko soweit wie möglich reduziert. Bei der Support-Vektor-Maschine wird das empirische Risko festgehalten (im einfachsten, fehlerfrei separierbaren Fall z. B. ist es gleich Null) und durch die Wahl der Klassifikatorstruktur in Abhängigkeit von der zur Verfügung stehenden Lernstichprobe wird das Konfidenzintervall minimiert.

3 Klassifikation mittels radialer Basis-Funktionen

Es wird in diesem Abschnitt, in etwas kürzerer Form, ein RBF-Netzwerk vorgestellt, um zu zeigen, wie ein klassisches System im Vergleich zur Support-Vektor-Maschine konstruiert wird.

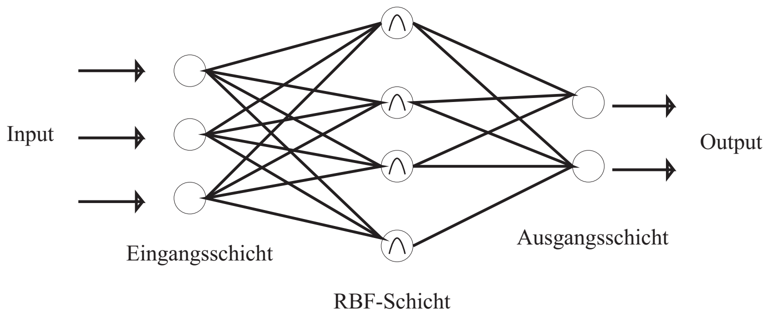

Abbildung 5: Struktur eines RBF-Netzwerks

3.1 Allgemeine Form

Radiale-Basisfunktions-Netze (RBF) sind Spezialfälle eines Neuronalen-Netztes. In Abbildung 5 ist die Struktur eines RBF-Netzwerkes dargestellt. Dahinter verbirgt sich eine typische feed-forward-Struktur. Radiale-Basisfunktion-Netze besitzen nur eine Schicht verdeckter Neuronen, die im folgenden mit RBF-Schicht bezeichnet wird. Diese Neuronen haben spezielle, radialsymmetrische (radiale) Aktivierungsfunktionen. Diese Aktivierungsfunktionen sind mathematisch gesehen die Basisfunktionen eines Funktionensystems zur Approximierung mehrdimensionaler Funktionen anhand von Funktionswerten an Stützstellen. Die Stützstellen sind im einfachsten Fall durch die Trainingsmuster, die dem Netzwerk präsentiert werden (Eingangsmuster und korrespondierendes Ausgangsmuster), vorgegeben. RBF-Netzwerke haben gegenüber anderen Netzwerkmodellen zwei entscheidende Vorteile:

1. Die Aktivierungsfunktion der Neuronen der RBF-Schicht liefert nur dann große Werte, wenn sich das Eingangsmuster in der Nähe einer Stützstelle befindet. Für Testmuster außerhalb des Bereichs, der durch die Trainingsmuster abgedeckt wurde, liefert das Netzwerk somit auch nur geringe Aktivierung am Ausgang. Dies steht im Gegensatz zu Netzen mit sigmoiden Aktivierungsfunktionen, wo für Testmuster, die außerhalb des durch die Trainingsdaten abgedeckten Bereichs unvorhersagbare und u.U. nicht interpretierbare Netzwerk-Ausgaben auftreten können.

2. Die einfache Struktur der RBF-Netze mit nur einer verdeckten Schicht erlaubt, eine direkte und damit nichtiterative Berechnungen der Gewichte des Netzwerks. Dies steht im Gegensatz zu den meist langwierigen iterativen Trainingsverfahren anderer Netztypen. Es ist ebenfalls möglich, RBF-Netze mit den direkt berechneten Gewichten zu initialisieren und mit Verfahren wie Backpropagation nachzutrainieren.

3.2 Gauß'sche Radiale-Basis-Funktion für 2-Klassen Problem

Eine klassische Gauß'sche Radiale-Basis-Funktion mit I Zentren hat die folgende mathematische Form:

$$g(x) = \sum_{i=1}^{l} w_i N_i(x) + b \tag{65}$$

$$= \sum_{i=1}^{l} w_i \frac{1}{(2\pi)^{d/2}\sigma_i^d} \exp\left(-\frac{\|x-\mu_i\|^2}{2\sigma_i^2}\right) + b \tag{66}$$

mit:

- I : Anzahl der Gauß-Verteilungen
- N_i : i-te Gauß-Verteilung

- μ_i : Mittelwert der i-ten Gauß-Verteilung
- σ_i^2 : Varianz der i-ten Gauß-Verteilung
- w_i : Gewichte
- b : Bias

Eine Klassifikationsentscheidung erfolgt dann (für 2-Klassen-Probleme) über ein Schwellenwert, d. h.

$$g(x) \geq T \quad \Longrightarrow \quad x \in k = 1 \tag{67}$$
$$g(x) < T \quad \Longrightarrow \quad x \in k = -1 \tag{68}$$

Die Konstruktion eines RBF-Netzes besteht in der Trainingsphase aus den folgenden Schritten:

1. Bestimmung einer geeigneten Anzahl Gauß-Verteilungen pro Klasse k.
2. Berechnung der Mittelwerte μ_i und der Varianzen σ_i für jede Klasse.
3. Bestimmung der Gewichtskoeffizienten w_i und dem Bias b, etwa durch den „error-backpropagation“ Algorithmus

Eine genauere Abhandlung über die RBF-Netze würde den Umfang dieser Ausarbeitung sprengen. Interessenten seien daher auf [3] verwiesen.

4 Beispielanwendungen

In diesem Abschnitt werden die Experimente vorgestellt, die mit der Support-Vektor-Maschine durchgeführt wurden. Die SVM wurde bei der Ziffernerkennung in zwei verschiedenen Standard Datensammlungen eingesetzt. Diese waren zum einen die kleinere Datensammlung, die „US-Postal-Service-Datensammlung“ und die größere Datensammlung, die “NIST-Datensammlung“ [2].

4.1 Die US-Postal-Service Datensammlung

Die US-Postal-Datensammlung ist ein standardisierter Corpus für die Bilderkennung. Es werden handgeschriebene Ziffern (US-Postleitzahlen) klassifiziert, welche eine Auflösung von 16×16 Pixel haben. Die Datensammlung beinhaltet etwa 7200 Traningsdaten, womit die Support-Vektor-Maschine konstruiert wird und etwa 2000 Testdaten, womit die Fehlerrate der konstruierten SVM herausgefunden wird.

Tabelle 1 — Ergebnisse unterschiedlicher Verfahren

Klassifikator	Fehlerrate
Human performance	2.5 %
Decision tree, CART	17 %
Decision tree, C4.5	16 %
Neuronales Netz (2-Schicht)	6.6 %
Neuronales Netz (5-Schicht)	5.1 %
Gauß’sche Mischverteilungen	3.7 %

In Tabelle 1 sind die Fehlerraten von verschieden Klassifikationsverfahren zu sehen. Es ist zu bemerken, daß die menschliche Fehlerrate bei 2.5% liegt [7].

Für die Experimente mit der Support-Vektor-Maschine sind die folgenden Optionen gewählt worden:

- Polynomiale Kernfunktion
- Es sind Polynome vom verschiedenen Grad eingesetzt worden
- Die Input-Vektoren haben 256 Merkmale ($x_i \in \Re^{256}$)

In Tabelle 2 werden die Ergebnisse gezeigt.

Tabelle 2 — Ergebnisse der SVM

Polynomgrad	Fehlerrate [%]	SV	Dim. Merkmalsraum
1 (lin. Klassif.)	8.9	282×10	256
2	4.7	227×10	$\sim$33000
3	4.0	274×10	$\sim 1 \times 10^6$
4	4.2	321×10	$\sim 1 \times 10^9$
5	4.3	374×10	$\sim 1 \times 10^{12}$
6	4.5	377×10	$\sim 1 \times 10^{14}$
7	4.5	422×10	$\sim 1 \times 10^{16}$

Es ist zu bemerken, daß das die beste Fehlerrate bei einem Polynomgrad von drei erreicht wird. Diese Ergebnisse sind 1996 ermittelt worden und ein aktuelleres Ergebnis ist, daß die SVM auf der US-Postal-Datensammlung eine Fehlerrate von 3.0% erreicht. Ein hochdimensionler Merkmalsraum führt nicht immer zu einer besseren Fehlerrate, dieser muß passend gewählt werden. Es ist zu erkennen, daß die Anzahl der Support-Vektoren in Bezug auf den Polynomgrad sehr langsam steigen. Der Polynomgrad 7 hat lediglich 35% mehr Support-Vektoren, als der Polynomgrad 3. Dabei ist die Dimension des Merkmalsraumes bei einem Polynomgrad von 7 etwa 10^{10}-mal größer, als die Dimension bei einem Polynomgrad von 3. Die hohe Anzahl an Support-Vektoren bei dem linearen Klassifikatior kommt daher zustande, daß die Trainingsdaten nicht separabel sind. Die Zahl 200 beinhaltet neben den bestimmten Support-Vektoren auch Trainingsdaten, die einen Faktor ξ ungleich Null haben. Der lineare Klassifikator macht im Durchschnitt etwa 34 Fehler. Ein Klassifikator vom Polynomgrad 2 macht dagegen im Durchschnitt nur noch 4 Fehler. Es ist zu bemerken, daß sogar eine SVM vom Polynomgrad 2 besser abschneidet, als ein neuronales Netz mit 5-Schichten.

Klassische Radiale-Basis-Funktionen erreichen auf der US-Postal-Datensammlung eine Fehlerrate von 6.7%. Die Support-Vektor-Maschine mit radialem Kern erreicht dagegen eine Fehlerrate von 4.1%.

4.2 Die NIST-Datensammlung

Ein weiteres Experiment wurde auf der größeren NIST-Datensammlung durchgeführt. Die Datensammlung von NIST ist wie folgt aufgebaut:

- handgeschriebene Ziffern
  (28×28 Pixel-Bilder)
- 60.000 Trainingsdaten
- 10.000 Testdaten

Es wurde dabei folgende die Support-Vektor-Maschine konstruiert:

- Polynomiale Kernfunktion vom Grad 4
- Dimension des Eingaberaumes ist 784 (pro Pixel ein Merkmal)

Die Ergebnisse der NIST-Datensammlung sind in Abbildung 6 zu sehen.
Auch in der NIST-Datensammlung erzielt die Support-Vektor-Maschine eine sehr gute Fehlerrate mit 0.8%. Es ist zu bemerken, daß neuronale Netze (LeNet1, LeNet4) speziell für die Ziffernerkennung konstruiert worden sind, aber ihre Fehlerraten mit 1.7% und 1.1% immer noch schlechter sind als die SVM. Allerdings erreicht auf NIST bereits ein 3-nearest-Neighbour Verfahren eine Fehlerrate von 2.4%.

5 Zusammenfassung

In dieser Seminararbeit wurde ein neue Art von Lernmaschine für 2-Klassen-Klassifikationsprobleme vorgestellt, nämlich die Support-Vektor-Maschine. Ist ein Mehrklassen-Problem gegeben, so wird dieses durch die SVM auf ein 2-Klassen-Problem abgebildet. Realisiert wird die Support-Vektor-Maschine durch eine lineare Entscheidungsfläche, die zwei gegebene Klassen mit möglichst großem Abstand und möglichst wenig Fehlern voneinander trennt. Es wurde ausfürlich gezeigt, wie eine Entscheidungsfunktion im idealen Fall

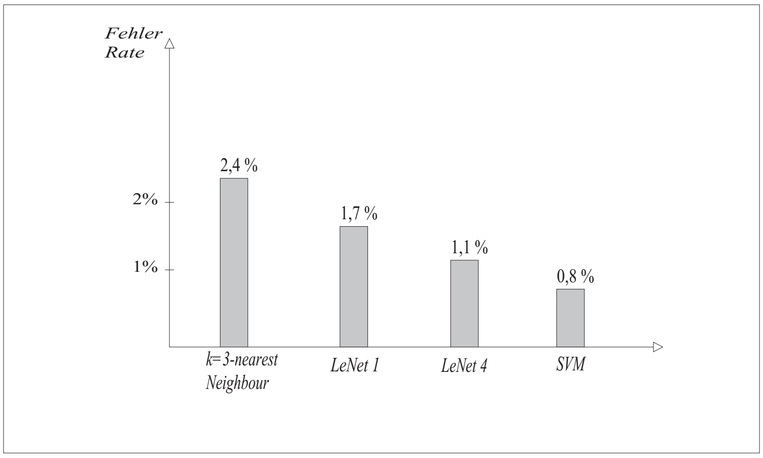

Abbildung 6: Fehlerraten von Klassifikatoren

(fehlerfrei separierbare Trainingsdaten) konstruiert wird. Für die Konstruktion einer trennenden Hyperebene ist das quadratische Optimierungsproblem zu lösen, wodurch die gesuchten Support-Vektoren ermittelt werden. Aus dem Normalenvektor, welcher sich aus der Superpositionierung der Support-Vektoren berechnet, folgen dann schließlich der Bias und damit die Entscheidungsfunktion. Es wurde desweitereren gezeigt, wie der realistische Fall (nicht fehlerfrei separierbare Trainingsdaten) behandelt werden kann. Die komplexen Skalarprodukte im Merkmalsraum, die bei der Konstruktion einer Hyperebene zu berechnen sind, können durch geeignete Kernfunktionen im Eingaberaum ersetzt werden, welche die Berechnungen sehr vereinfachen. Zum Vergleich der SVM, wurde auch die Radial-Basis-Funktion vorgestellt, welches ein Spezialfall eines neuronalen Netzes ist. Die generellen Eigenschaften und auch die Experimente mit einer SVM zeigen einen Klassifikator, welcher hervorragende Ergebnisse liefert.

Die Support-Vektor-Maschine erweist sich als ein mächtiger Ansatz zur Adaption von Klassifikatoren, der zudem einfach zu handhaben ist, da eigentlich nur zwei Parameter (Kosten C und Kernfunktion K) geeignet auszuwählen sind. Es ist jedoch sehr wichtig, die Randbedigungen einer SVM (wie Größe der Lernstrichprobe, Dimension des Merkmalsraumes, Anzahl der Klassen) genau zu beachten, um jeweils einen geeigneten Ansatz (Festlegen eines primären Ziels: Minimale Fehlerrate oder möglichst großen Abstand zwischen den beiden Klassen) auszuwählen. Nachteilig soll hier erwähnt werden, daß eine relativ hohe Rechenzeit in Kauf zu nehmen ist (ähnlich wie bei Nearest-Neighbour-Verfahren), welche die SVM ein wenig zurückwirft. Aber andere Qualitäten wie Universalität und Generalisierung zeigen, das die Support-Vektor-Maschine ein wirklich leistungsstarkes Verfahren ist.

Literatur

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

[2] V. Vapnik, C. Cortes: *Support-Vector-Networks*, Machine Learning, 20, 273-297, Boston, 1995.

[3] B. Schölkopf, V. Vapnik, K. Sung, C. Burges: *Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers*, A.I. Memo No. 1599, C.B.C.L. Paper No. 142, Massachusetts Institute of Technology, Dezember 1996.

[4] I.N. Bronstein: *Taschenbuch der Mathematik*, 2.Auflage, Frankfurt am Main, 1995.

[5] N.I. Achieser: *Theorie der linearen Operatoren im Hilbert-Raum*, Verlag Harri Deutsch Thun - Frankfurt/Main, 1977.

[6] H.P. Künzi: *Nichtlineare Programmierung*, Springer-Verlag, Berlin, 1979.

[7] P. Simard, Y. Le Cun, J. Denker, „Efficient Pattern Recognition Using a New Transformation Distance," S.J. Hanson, J.D. Cowan, C.L. Giles (eds.): *Advances in Neuronal Information Processing Systems 5*, Morgan Kaufmann, San Mateo CA, pp. 50-58, 1993.