Ein Beitrag zum Vergleich von Radialen Basisfunktions-Netzen und Support-Vektor-Maschinen

Möglichkeiten und Grenzen beider Konzepte anhand eines Beispiels aus der visuellen Mustererkennung

vorgelegt als

Diplomarbeit

beim Prüfungsamt Mathematik und Informatik
der FernUniversität Hagen

von

Walter Löser
24983 Timmersiek
Januar 2002

Betreuer und Prüfer:

Prof. Dr. Hermann Helbig, FernUniversität Hagen
PD Dr. Josef Pauli, Christian-Albrechts-Universität Kiel

In memoriam Petra

Zusammenfassung

Nach einer strukturellen und funktionalen Gegenüberstellung von klassischen Radialen Basisfunktions-Netzen und Support-Vektor-Maschinen wird erörtert, ob Radiale Basisfunktions-Netze als Spezialfall von Support-Vektor-Maschinen angesehen werden können. In Konsequenz dieser Erörterung wird ein Fusions-Netz vorgeschlagen und entworfen.

Für einen ersten empirischen Vergleich zwischen Radialen Basisfunktions-Netzen, Support-Vektor-Maschinen und dem entwickelten Fusions-Netz wird ein Gegenstandsmodell aus dem Bereich der Mustererkennung (zur Erkennung von Gesichtern) vorgestellt. Davon ausgehend werden die Untersuchungshypothesen aufgestellt, und der Versuchsaufbau mit den drei Simulatoren wird beschrieben.

Nachfolgend werden neun Vergleichskriterien zur Beurteilung der Lerndaten, zur Bedeutung der Netzparameter und zur Gestaltung der Lernprozesse hergeleitet.

Abschließend werden die Ergebnisse der empirischen Untersuchung referiert, und es werden erste Folgerungen gezogen.

4. Analyse der Vergleichskriterien

5. Diskussion der Ergebnisse und Vergleich

Anhang

1. Der Einsatz von RBF- und SVM-Netzen in der visuellen Mustererkennung

Die Konzepte von neuronalen Netzen bzw. Lernmaschinen entstanden aus dem Bedürfnis, menschliches Denken auf einer subsymbolischen Repräsentationsebene zu simulieren. Der Simulationsanspruch bezieht sich zunächst nur auf das Eingabe/Ausgabe-Verhalten, d.h. es wird nicht von vornherein unterstellt, daß der Verlauf der Simulation auch tatsächlich Abläufen im menschlichen Gehirn entspricht, wenn auch die Idee dieser Strukturen sehr suggestiv auf Einsichten über die Weitergabe elektrischer oder chemischer Impulse von Nervenzellen beruht. Dies gilt nicht nur für Softwaresimulationen, auch spezielle Neuroprozessoren können keinen umfassenden Anspruch an Wirklichkeitsnähe erfüllen. Insofern ist die übliche Graphik eines neuronalen Netzes lediglich eine abstrakte Veranschaulichung einer mehrstufigen mathematischen Abbildung mit graphentheoretischen Interpretationsmöglichkeiten und repräsentiert nicht detailgetreue biophysikalische Gegebenheiten.

In der vergleichsweise kurzen Geschichte dieser Theorie hat es (in Abhängigkeit vom jeweiligen Stand der technischen Evolution) die verschiedensten Ansätze gegeben, je nachdem, welcher Zweck von der Simulation erfüllt werden sollte; z.B. die Bewältigung von Echtzeitproblemen, bei denen eine exakte Lösung entweder wegen Unterbestimmtheit der Parameter nicht möglich ist oder zu einem unvertretbar bzw. unnötig hohen Aufwand führen würde.

Eine der Domänen zur Anwendung neuronaler Netze besteht nach dem Abschied von der GPS-Euphorie der frühen siebziger Jahre neben der Robotik in der Darstellung abgrenzbarer, menschlicher Hirnleistungen wie z.B. Erkennung gesprochener Sprache oder visuelle Wahrnehmung. Eine besondere Rolle spielen dabei Identifikations-Prozesse, die sich mathematisch als Zwei-Gruppen-Klassifikationsprobleme kodieren lassen.

Es könnte intuitiv naheliegend erscheinen, in der vorliegenden Domäne der Mustererkennung vorzugsweise mit KOHONEN- oder HOPFIELD-Netzen zu arbeiten. Hingegen gibt es im wesentlichen drei Gründe, auf der Basis von Radialen Basis-Funktions-RBF-Netzen (RBF) zu arbeiten, von denen jeder für sich genommen nicht ausschließlich auf RBF-Netze verweist, in der Gesamtheit aller Gründe aber gewichtig wird.
(G1) Es gibt ein Analogon zu dem biologischen Modell der rezeptiven Felder, d.h. der Zuordnung bestimmter sensitiver Bereiche des Sehens zu bestimmten Wahrnehmungsleistungen (s. 3.1.)

(G2) Es bietet sich die Möglichkeit, algorithmisch ebenso wie biologisch bottom up-Strategien zu verfolgen; sie sind dabei dynamisch zur Laufzeit; (s. 2.1.)

(G3) Es ist ein Lernen ohne Iterationszyklen möglich, d.h. die Gewichte der verdeckten Schicht sind, auch wenn sie voneinander abhängig bleiben, separat behandelbar. Man kann mit *divide & conquer*-Strategien zu einer Clusterung kommen, ohne die Ausgabe zu kennen (s. 2.2.)

In der praktischen Anwendung von RBF-Netzen wurden allerdings eine Reihe von Schwächen und Grenzen deutlich (s. 2.3.1.), die es durchaus naheliegend erscheinen ließen, nach Verfahren Ausschau zu halten, die Vergleichbares leisten, aber solchen Beschränkungen nicht unterliegen.

In der aktuellen Diskussion gilt das neuere Konzept der Support Vektor-Maschinen (SVM) im Vergleich zu RBF-Netzen gemeinhin als umfassender und erfolgreicher, was aber weder aus der Vielzahl einschlägiger, empirischer Befunde noch theoretisch aus seiner Anlage letztendlich begründbar wäre, da beide Konzepte aus unterschiedlichen Kalkülen abgeleitet werden. Außderdem wurden bislang vorliegende Vergleiche zwischen RBFs und SVMs zumeist unter der fragwürdigen Annahme geführt, RBFs seien funktional ein Spezialfall von SVMs (s. dazu 2.3.1.).

Geht man von einer gemeinsprachlichen Umschreibung des Begriffs 'neuronales Netz' aus, wie er sich etwa in Lehrbüchern findet (vgl. **ZELL** [Zell1]), so kann es plausibel sein, RBF-Netze ebenso wie SV-Maschinen als neuronale Netze aufzufassen. Legt man eine graphentheoretisch formale Definition für 'neuronales Netz' zugrunde (vgl. **LENZE** [Lenze]), ist die Zuordnung für die SV-Maschinen weniger offensichtlich. Dieser Frage wird also nachzugehen sein (s. 2.2).

Die systematische Gegenüberstellung der theoretischen Grundlagen von RBF und SVM findet sich in Kapitel 2 .

Als Gegenstandsmodell für den Vergleich (Kapitel 3) wurde die anwendungsrelevante Domäne der Gesichtererkennung (*face recognition*) gewählt, da man in diesem Bereich der Mustererkennung aufschlußreiche Ansatzpunkte für Gegenüberstellungen von menschlichen und algorithmischen Vorgehensweisen und deren Konsequenzen findet (z.B. Explikation von Entscheidungskriterien, Einfluß von Vorinformationen, Merkmale von Habituierungen).

Aus den Eigenheiten der Domäne (3.1.) werden die Untersuchungshypothesen entwickelt (3.2.) und die Rahmenbedingungen für den empirischen Teil festgelegt (3.3.).

Die einzelnen Vergleichkriterien (Kapitel 4) für die Leistungsfähigkeit beider Netz-Typen lassen sich auf naheliegende Weise drei Ebenen zuordnen:

(1) Kriterien, die systematische Variationen in der Grundgesamtheit der Daten betreffen (s. 4.1)
(2) Kriterien, die systematische Variationen an den Netzparametern betreffen (s. 4.2)
(3) Kriterien, die systematische Variationen an den Lernverfahren betreffen (s. 4.3).

Die Ergebnisse des empirischen Versuchs werden in Kapitel 5 vorgestellt (5.1.) und interpretiert (5.2.; 5.3.) .

Damit beschränkt sich die Untersuchung innerhalb der vielfältigen Anwendungsmöglichkeiten für RBF- und SVM-Netze auf 2-Gruppen-Klassifikationsprobleme und ist explorativ angelegt. Ihre Ergebnisse können daher weder ohne weiteres auf Anwendungen wie etwa Regressionsschätzungen übertragen werden noch können sie im Bereich der Mustererkennung Anspruch auf Repräsentativität erheben.

Eine systematische Exploration beider Konzepte erscheint nicht zuletzt deshalb vielversprechend, weil in verschiedenen Forschungsberichten auffällt, daß bisweilen mit sehr aufwendigen, vielversprechenden Variationen eines Simulationsansatzes kaum Erfolge erzielt wurden, während in anderen Fällen unter eher geringfügiger Manipulation der Vorgehensweise praktische Ergebnisse (deutlich unterhalb von theoretisch herleitbaren Fehlerschranken) verbessert werden konnten, daß u.U. sogar Varianten einer Simulation, die theoretisch überhaupt nicht gedeckt waren, zu einer erstaunlichen Performanz führten.

Dafür können folgende Beispiele benannt werden:

- Extraktion schwacher statt starker Merkmale (vgl. 4.1.1.)
- abnehmender Leistungszuwachs bei zunehmendem polynomialen Grad von Kernen (vgl. 2.2.1.)
- erfolgreiche Heuristiken mit sog. *non mercer* - Kernen (vgl. 2.2.2.)
- Einsatz von *nearest-miss*-Vektoren (vgl. 2.1.3.)

Im vorliegenden Beitrag werden zusammengefaßt drei Ziele verfolgt:

(Z1) Entwurf eines Ansatzes zur **Fusion** beider Konzepte, um unter Nutzung der jeweiligen Vorteile die jeweiligen Schwächen zu mildern oder auszuschalten; (Sicherung von Randbedingungen, um RBF-Netze als einen bestimmten Typ von SVM-Netzen auffassen zu können (s. 2.3);

(Z2) Vorschlag zur **Integration** beider Konzepte durch Herleitung einheitlicher Gütekriterien (s. 3.3);

(Z3) Erkundung einer Möglichkeit zur **Expansion** der Nutzbarkeit der Netze durch explizierte Einbringung von Kontext-Information (s. 4.3.).

2. Architektur und Funktionsweise von RBF- und SVM-Verfahren

2.1. Kennzeichen von RBF-Netzen

2.1.1. Merkmale der Netzarchitektur

Da der Graph eines neuronalen Netzes nur eine rein schematische Veranschaulichung darstellt, die keine detailgetreuen hirnphysiologischen Entsprechungen beansprucht (vgl. 1.), kommt es darauf an, die wesentlichen Eigenschaften der Architektur eines bestimmten Netztyps in mathematischen Kategorien zu beschreiben. RBF-Netze sind feedforward-Netze, d.h. Netze ohne Rückkoppelungen und rekursive Zuordnungen, mit genau einer verdeckten Schicht, insbesondere vollständig vermascht und ohne shortcut-Verbindungen. Die Parameter der mittleren Stufe der Funktionenkette lassen sich demnach in einem Zyklus berechnen (Abb. 2.1.1.).

Da die Netzarchitektur mathematisch gesehen in Matrizen repräsentiert ist, deren Koordinaten die Knoten identifizieren und deren Positionen mit allen Informationen über die Lage, Stärke und Richtung der Verbindungen belegt sind, wird mit dieser Art der Repräsentation auch das methodische Paradigma zur Bearbeitung der Informationen eröffnet.

Die in Frage stehenden Netze sind dynamisch, was bedeutet, daß die verdeckte Schicht während der Trainings um neue Knoten, d.h. im vorliegenden Fall Stützstellen der Approximationsfunktion, erweitert wird. Jedes Trainingsdatum erzeugt in diesem Sinne ggf. einen neuen verdeckten Knoten. Es wird später zu zeigen sein, daß dennoch nicht eine möglichst große Anzahl von Knoten automatisch zu den besten Ergebnissen führt (vgl. 4.1.1.).

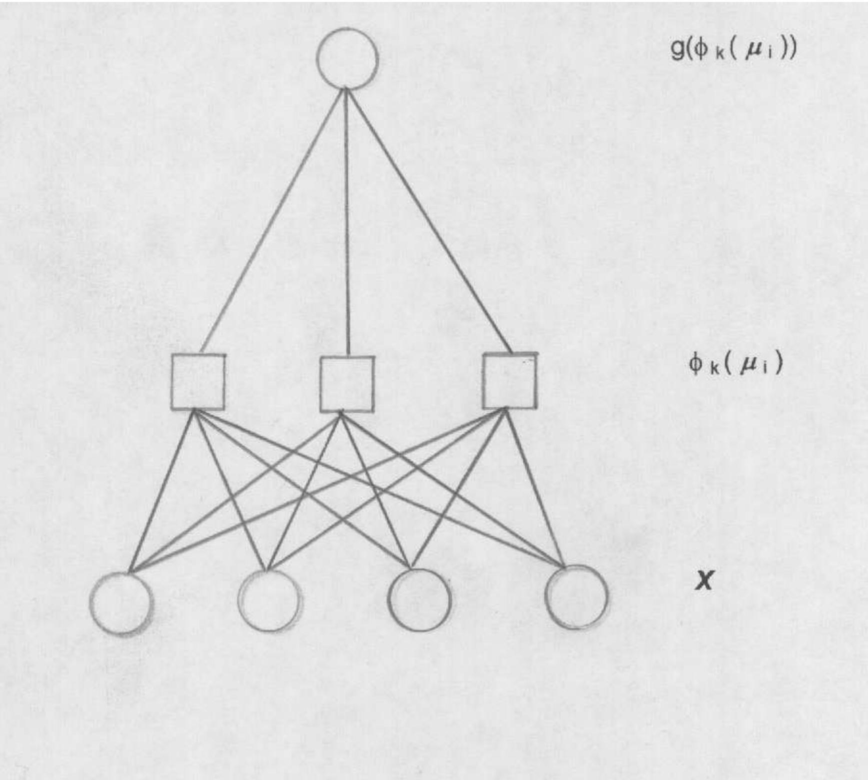


Abb. 2.1.1. Standard-Struktur von RBF-Netzen

2.1.2. Aktivierungs- und Entscheidungsfunktion

Radialsymmetrische Basisfunktionen werden definiert aufgrund bestimmter Annahmen über die Verteilung (Dichte) von Punktmengen in multidimensionalen Räumen (s. 4.1.2.) und haben die Eigenschaft, Beziehungen in diesen Punktmengen zu interpolieren. Um dies exakt zu tun, entspräche die Anzahl der jeweils benötigten Funktionen der Dimension des Raumes.

Ein RBF-Netz modelliert hingegen eine Approximation dieses exakten Verfahrens, die für den betreffenden Fall als hinreichend gut angesehen wird, mit wesentlich weniger solcher Funktionen, indem immanente Zusammenhänge aus bekannten geschätzt werden, d.h. in der Sprache des NN-Kalküls ausgedrückt: indem das Netz allgemeine Zusammenhänge aus einer Menge von Trainingsdaten lernt.

Gängige RBF-Klassen mit $\varphi_i : \boldsymbol{IR}^n \to \boldsymbol{IR}$ sind:

Gaußfunktionen:
$\varphi_i(x,\sigma_i) = \exp(-x^2 / 2\sigma_i)$ für $x = |X - X_i|$

multiquadratische Funktionen:
$\varphi_i(x,\sigma_i) = \mathrm{sqrt}(\sigma_i + x^2)$ für $x = |X - X_i|$

Flach-Spline-Funktionen:
$\varphi_i(x, \sigma_i) = (\sigma_i\ x)^2 \ln(\sigma_i\ x)$ für $x = |X - X_i|$

Besondere Bedeutung haben darunter die GBF-Netze, diejenigen RBF-Netze, bei denen Gaußfunktionen zur Aktivierung angesetzt werden, die dann genauer folgende Struktur aufweisen:

$\varphi_i : \boldsymbol{IR}^n \to \boldsymbol{IR}$ mit

$$\varphi_i(x, \sigma_i) = \exp(-x^2 / 2\sigma_i^2) \text{ für } x = | X - \mu_i |, \tag{F.2.1-1}$$

wobei $|\,|$ eine L2-Norm bezeichnet (im $\boldsymbol{IR}^n$ üblicherweise den euklidischen Abstand); σ_i als vom Anwender steuerbarer Parameter bezeichnet die Varianz (Gaußsches Rauschen) und beeinflußt die Glätte der Approximation.

Mit σ_i verändert sich der Ausstrahlungsradius um die Stützstelle μ_i herum, die ein Muster repräsentiert. Eine Vergrößerung von σ_i dient unter Umständen dazu, Lücken zwischen den ursprünglichen Ausstrahlungsradien zweier φ_i zu schließen.

Kleine Werte von σ_i führen zu steilen, schmalen Kurven von φ_i, große σ_i zu breiten, flachen, wobei aber die max. Höhe an der Stelle $\varphi_i(\mu_i) = e^0$ auf 1 normiert bleibt (Abb. 2.1.2a).

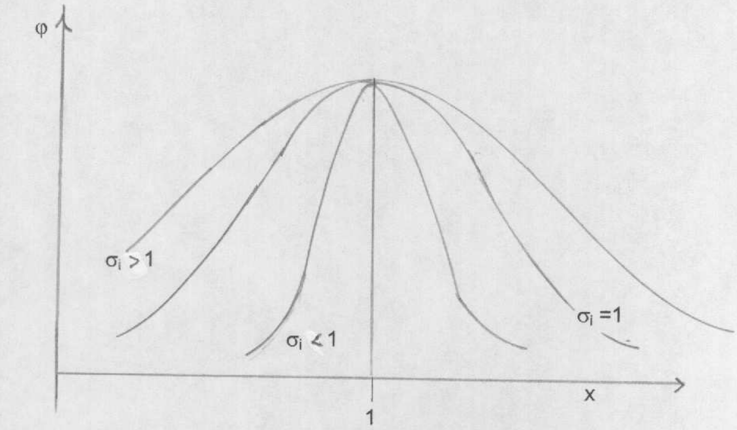


Abb. 2.1.2.a φ : *IR* -> *IR* mit $\varphi(x,1)=\exp(-(x-1)^2 / 2\,\sigma_i)$

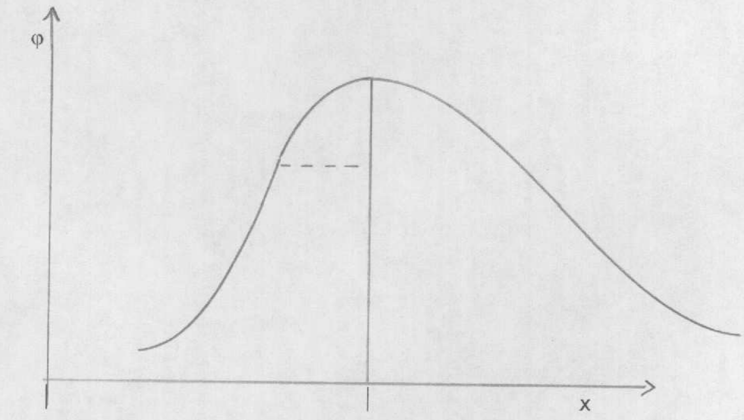


Abb. 2.1.2.b φ : *IR* -> *IR* mit φ(x,1),
normiert auf arbiträre Kovarianzen

Wichtig für eine Beurteilung wird, daß σ_i der einzige vom Anwender beeinflußbare Netzparameter ist, so daß nicht komplexe Wirkungssyndrome mehrerer Parameter durch tupelweise Variation entschlüsselt werden müssen.

Bei allen Unterschieden zwischen Ausprägungen der Funktion (größere σ_i führen u.U. zu gewagteren Approximationen) gelten die RBFs dennoch als robust gegen eine ungünstige Wahl der σ_i. Das bedeutet aber unter Umkehrung der Perspektive auch, daß von einer systematischen Variation der σ_i allein keine markanten Unterschiede im Konvergenzverhalten erwartet werden können.

Nutzt man die radialsymmetrischen Funktionseigenschaften bei der Lösung von 2-Gruppen-Klasifikationsproblemen, so liefern i.a. Identitätsabbildungen von folgender Struktur die Entscheidungsfunktionen des Netzes:

g: ***IR*** -> ***IR*** mit

$g(x, \sigma_i) = \text{ident}\,(\Sigma_i\, w_i\, \varphi_i\,(x)),$ (F.2.1-2)

wobei w_i die Gewichte der einzelnen Knoten beschreiben, also die Intensität der Verstärkung oder Hemmung eines Informationsimpulses bei seiner Weitergabe; i=1...k der Anzahl verdeckter Neuronen.

Der SNNS von **ZELL** [Zell1] bietet neben der oben skizzierten Aktivierungsfunktion ACT_RBF_GAUSSIAN die Variante ACT_RBF_MULTIQUADRATIC an, auf die unter 2.3.2. noch einzugehen sein wird.

Als Entscheidungsfunktion bietet sich im SNNS ACT_IDENTITYPLUSBIAS an, um möglichst ähnliche Bedingungen zur üblichen Trennfunktion bei den SVM zu schaffen. (Eine Alternative wäre etwa Act_Logistic [Zell2] .)

Enthält die Verteilung in der Datengrundgesamtheit Unregelmäßigkeiten, so daß eine radialsymmetrische Ausstrahlung um die Stützstellen die Verteilung nicht mehr zutreffend abbildet, führt dies zu einer u.U. dramatischen Vermehrung der Anzahl der benötigten Stützstellen und damit der benötigten φ_i. Um dem entgegenzuwirken und die Unregelmäßigkeiten durch eine Modulation des Austrahlungsradius aufzufangen, führt man im Argument der φ_i anstelle des Betrages mit dem Quadrat des Abstandes $|X\text{-}\mu_i|$ im Exponenten ein Skalarprodukt aus der Vektordifferenz $|X\text{-}\mu_i|$ und der Inversen der (symmetrischen) Kovarianzmatrix mit $|X\text{-}\mu_i|^T$ ein, also nunmehr

$\varphi_i(x) = \exp(-(|X\text{-}\mu_i|^T\, \text{CoVar}_i^{-1}\, |X\text{-}\mu_i|) / 2)$ mit $x = |X\text{-}\mu_i|$.

Mittels der Matrix-Inversen wird also ein Quotient nachgebildet, d.h. die Mittelwerts-Abstände werden zu Kovarianzen ins Verhältnis gesetzt, was bekanntlich einer Art Normierung gleichkommt. Setzt man alle Werte außerhalb der Diagonalen auf 0, so betrachtet man die Varianzen von μ_i; insbesondere erhält man bei auf 1 normierter Hauptdiagonale gerade die bisher zunächst vorgestellte Norm $|X\text{-}\mu_i|^2$. **BISHOP** [Bisho] S. 169 weist darauf hin, daß man arbiträre Kovarianz-Matrizen zur Normierung verwenden kann, also etwa Kovarianzen zwischen den jeweiligen Eingabevektoren und der Stützstelle. Je nach Höhe der Kovarianzen werden die Funktionswerte $\varphi_i(x)$ größer oder kleiner, d.h. es findet eine abschnittweise Stauchung oder Streckung der Kurve statt (Abb. 2.1.2.b).

2.1.3. Adaption und Validierung

Ein übliches Training für RBF-Netze kann in den folgenden drei Stufen charakterisiert werden:

(1) Festlegung einer Anzahl k von Zentrumsfunktionen (Gaußschen Kernen) für die Initialisierung. Um den sukzessiven Aufbau des Netzes zu kontrollieren, erlaubt es der zur Simulation verwendete SNNS, k=0 zu setzen, d.h. mit einer leeren Netzstruktur aus Eingabe- und Ausgabe-Einheiten ohne verdeckte Schicht zu beginnen. Dies hat zur Folge, daß der Anordnung der Muster im Trainings-Datensatz die gleiche herausgehobene Bedeutung zukommt wie der Entnahme von Mustern aus der Grundgesamtheit bei einer Initialisierung per Hand, da das erste Muster X_1 in jedem Falle den ersten verdeckten Knoten markiert, die erste Stützstelle μ_1, das Zentrum des ersten Gaußschen Kerns φ_1, bildet. Der Knoten erhält das vorläufige Gewicht 1; $\varphi_1 (x_1) =: \varphi_1 (\mu_1) = e^0$.

(2) Mit dem nächsten Eingabe-Vektor (Muster) wird geprüft, ob der Wert von $\varphi_1 (X_2)$ noch oberhalb der Aktivationsschwelle liegt (hängt von der Summe der Abweichungsquadrate über alle Merkmale und von σ_1 ab); wenn ja, bleibt alles beim alten, wenn nein, wird ein neuer Knoten für X_2 generiert usw.

(3) Ab 2 Knoten werden die Gewichte angepaßt, wann immer ein neuer Eingabe-Vektor hinzukommt.

Während die Verwendung eines random-basierten Algorithmus etwa den Vorteil hat, daß man nicht unbemerkt biasbehaftete Fehler bei der Initialisierung machen kann, ist er für den angestrebten Vergleich gänzlich ungeeignet, weil kein Lernzyklus gewollt identisch reproduziert werden kann. Unter den im SNNS implementierten, spezifischen Algorithmen zur Gewichtsberechnung wurde der Dynamic-Decay-Algorithmus DDA für RBF-Netze ausgewählt [Zell2] S. 152ff . Die wesentliche Idee besteht darin, 2 Schwellenwerte für die Aktivation eines Knotens einführen, eine obere Schranke $\theta+$, oberhalb derer der Knoten planmäßig durchlaufen wird, und eine untere Schranke $\theta-$, unterhalb derer der Knoten planmäßig blockiert ist. Im einzelnen wird folgendermaßen vorgegangen:
Fall 1: ein eingeführtes Muster wird über die vorhandenen Knoten mit korrekter Ausgabe durchgeschleust.
> Fall erledigt.
Fall 2: ein eingeführtes Muster führt zu einer unkorrekten Ausgabe (die nicht der Klassenmarke entspricht).
Fall 2.1. Es gibt einen Knoten, der zwischen $\theta+$ und $\theta-$ dieses Muster abdeckt.
> Das Gewicht dieses Knotens wird erhöht. Fall erledigt.

Fall 2.2. Es gibt keinen Knoten, der zwischen $\theta+$ und $\theta-$ dieses Muster abdeckt.

> Es wird ein neuer Knoten eingerichtet.

Fall 2.2.1. Der radiale Ausstrahlungsbereich des neuen Knotens hat zwischen $\theta+$ und $\theta-$ Überschneidungen mit dem Ausstrahlungsbereich eines Knotens der Komplementärklasse.

> Die überlappenden Bereiche werden gekappt.

Fall 2.2.2. *backpropagation*: Überschneidungen mit dem Radius gegenläufiger Knoten kommen nicht vor.

> Fall erledigt.

Abb. 2.1.3. zeigt beispielhaft ein Stadium eines Netztrainings, bei dem bereits je ein Knoten für Klassenzugehörigkeit zu A und ein Knoten für Klassenzugehörigkeit zu A^c mit unterschiedlichen Gewichten entstanden sind. Wäre nunmehr das nächstfolgende Muster *X* auf der *X*-Achse an der Stelle (1) zu markieren, so würde es das Gewicht des A-Knotens erhöhen, an der Stelle (3) das Gewicht des A^c-Knotens. An den Stellen (2) oder (4) würde das nächstfolgende Muster zur Stützstelle eines neuen Knotens führen, bei (2) deshalb, weil es im Konfliktbereichbereich der Ausstrahlung zwischen A und A^c liegt.

Dieses Verfahren zeitigt nach Zell u.a. folgende Eigenschaften:
(1) Das Verfahren konvergiert.
(2) Nach Beendigung des Trainings gilt für sämtliche Trainingsmuster: unkorrekte Klassifikationen landen unterhalb von $\theta-$, korrekte oberhalb von $\theta+$.

Zell weist daraufhin, daß brauchbare Werte für $\theta+$ und $\theta-$ weitgehend auf Erfahrung beruhen. Der genannte Wert für $\theta+$ =.4 erscheint zwar auf den ersten Blick sehr niedrig; dieses Niveau stellt aber sicher, daß auf keinen Knoten zu viel Gewicht gelegt wird, wodurch das Netz unflexibel würde. Für die untere Schranke gibt Zell $\theta-$ =.2 an. Zu beachten bleibt, daß die Werte für $\theta+$ und $\theta-$ die Wirksamkeit der Werte für die Netzparameter σ_i verändern.

Es wurde bereits darauf hingewiesen, daß aufgrund der einzelnen verdeckten Schicht die Gewichte zwar nicht unabhängig voneinander sind, aber separat berechnet werden können. Die Reihenfolge wird im Rahmen einer SNNS-Simulation durch eine *update*-Funktion festgelegt. Hier bietet sich für den angestrebten Vergleich an, die Variante SYNCHRONOUS_ORDER anzuwählen, weil man sich damit am ehesten dem gänzlich simultan orientierten Optimierungskonzept der SVM annähern kann.

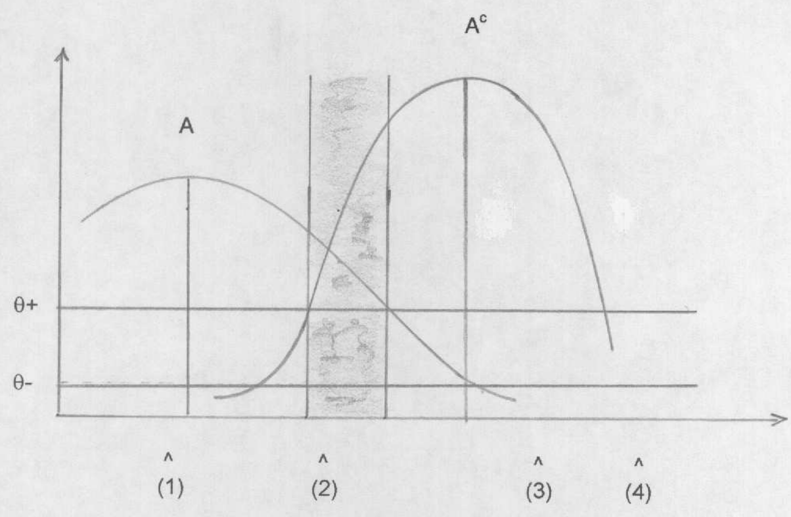


Abb. 2.1.3. Ausstrahlungsbereich und Wirkungszone beim DDA

2.2. Kennzeichen von SVM-Netzen

2.2.1. Merkmale der Netzstruktur

Support-Vektor-Maschinen (oder Support-Vektor-Netze) dienen außer zur Approximation von Funktionen zur Regressions-Schätzung und finden Anwendung bei der Lösung von (insbesondere 2-Gruppen)-Klassifikationsproblemen, d.h. Fragestellungen, bei denen für eine vorgelegte Instanz einer Grundgesamtheit entschieden werden soll, ob sie ein bestimmtes Eigenschaftsmuster besitzt oder nicht.

Die Eingabevektoren beschreiben die Merkmale der Eigenschaftsmuster auf der ersten Ebene (*input space*). Daraus wird in einem Optimierungsprozeß eine Anzahl von Support-Vektoren selektiert, die als Vergleichsfälle zur Abgrenzung dienen.

Aus ihnen wird (im Bedarfsfalle über nicht-lineare Transformationen) auf der nächsten Ebene, dem sog. Merkmalsraum (*feature space*), eine optimale (d.h. mit möglichst breitem Rand versehene) Trenn-Oberfläche zwischen den gegensätzlichen Mustern konstruiert auf der Grundlage der Idee, den strukturellen Fehler innerhalb dieses Optimierungsprozesses durch gezieltes Abtasten der Ränder (*margins*) zu minimieren. Die Lage der Trenn-Oberfläche spiegelt sich wider in den Gewichten der Netzknoten und bestimmt, ausgedrückt in der Trennfunktion die Zugehörigkeit der Eingabe-Muster zu den Clustern (Abb. 2.1.1.).

Es ist demnach im Vergleich zu Abb. 2.1.1. nicht offensichtlich, ob es sich bei dieser Konstruktion überhaupt um NNs handelt.

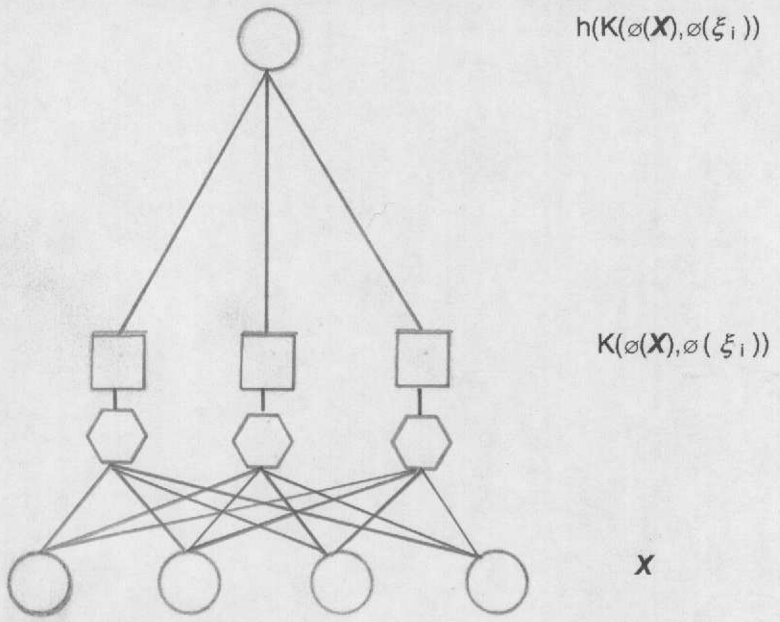


Abb. 2.2.1. Standard-Struktur von SVM-Netzen

Um aber dafür zu sorgen, daß mögliche Unterschiede weder aus dem Blick geraten noch unterschätzt werden, soll nachfolgend durchgängig eine konsequente Begriffstrennung vorgenommen werden: Während bzgl. der klassischen RBF-Netze von einer Entscheidungsfunktion sowie von Lernen als Training mit Anpassungs- und Validierungsdaten gesprochen wird, sollen bzgl. der SVM-Netze die Begriffe Trennfunktion sowie Stabilisierung (in Anlehnung an ein Homöostase-Pinzip; s. 2.2.3.) und Optimierung verwendet werden.

RBF-Netz		***SVM-Netz***
	Netzstruktur	
Stützstelle μ_i		Support-Vektor ξ_i
verdeckte Schicht		Merkmalsraum
Aktivierung		Transformation
Glättungsparameter σ		Kern-Parameter γ
Entscheidungsfunktion g		Trennfunktion h
	Lernen	
	Training	
Adaption		Stabilisierung
	Testung	
Validierung		Optimierung
	Kosten-/Nutzen-Abschätzung	
Gütefunktion		Fehlerfunktion

2.2.2. Transformations- und Trennfunktion

Die Funktion

f: $\boldsymbol{IR}^n \rightarrow 0$ mit $f(X) = X * W + b$ (F.2.2-1)

beschreibt die Lage der gesuchten trennenden Hyperebene, wobei W als Normalenvektor die Hyperebene aufspannt.

Dabei gilt diejenige Hyperebene als optimal, die den breitesten Rand zwischen den beiden Klassen von Mustern erzeugt. Das gesuchte W wird man als Lösung eines Extremwertproblems unter Nebenbedingungen durch quadratische Optimierung erhalten (s. 2.2.3.). Daraus ergibt sich

$f(X) = X * (\sum_i \alpha_i \; y_i X_i) + b$ (F.2.2-2)

mit den Lagrange-Koeffizienten α_i und den Klassenmarkierungen y_i.

Da im allgemeinen die beiden gegensätzlichen Musterklassen im Eingaberaum nicht separierbar sind, wird das Problem dergestalt transformiert, daß die Gegebenheiten des Eingaberaumes in einen höher-dimensionalen Merkmalsraum abgebildet werden:

$\phi_i : X \rightarrow \phi_i(X)$, $\boldsymbol{IR}^n \rightarrow \boldsymbol{IH}$

Diese Kartierung ermöglicht, mit Hilfe der hinzugewonnenen Dimension(en) die Musterstrukturen so zu abstrahieren und anzuordnen, daß innerhalb des Merkmalsraumes eine lineare Separierung möglich wird. Damit geht f über in

$f(\phi_i (X)) = \sum_i \alpha_i \; y_i * \phi_i (X_i) * \phi_i (X) + b$ (F.2.2-3)

Das Eigenartige dieser Konstruktion besteht nun darin, daß zwar die Existenz dieser ϕ_i theoretisch gesichert ist, ihre tatsächliche Beschaffenheit aber i.a. nicht bekannt ist. Sie müssen allerdings auch gar nicht explizit rekonstruiert werden. Vielmehr nutzt man eine Eigenschaft aus dem Kalkül der Funktionalanalysis von Integraloperatoren (Kernen) aus, wonach die komplette Struktur $\phi_i (X_i) * \phi_i (X)$, die nur in eben dieser Verknüpfung als Skalarprodukt vorkommt, ersetzt werden kann durch $\phi_i (X_i) * \phi_i (X) = K(X_i, X)$.

Man erreicht damit eine wesentliche algorithmische Vereinfachung, indem statt der aufwändigen Linearkombination aus dem Skalarprodukt $\phi_i (X_i) * \phi_i (X)$ die geschlossene Form $K(X_i, X)$ berechnet werden kann.

Die für den Ausgangsfall hergeleitete trennende Hyperebene wird auf diese Weise durch eine räumlich gekrümmte Trennfläche im Originalraum substituiert und dargestellt.

Grundsätzlich kommen alle Funktionen als Kerne in Betracht, die der *mercer condition* (s. ANHANG (3)) genügen; gängig sind K: $\boldsymbol{IR}^n * \boldsymbol{IR}^n \rightarrow \boldsymbol{IR}$

polynomiale mit $K(X, Y) = (X * Y)^d$

Solche Netzwerke führen z.B. schon zu einer guten Performanz bei relativ niedrig-gradigen polynomialen Kernfunktionen, während höhergradige Polynomkerne nicht mehr zu einem wesentlichen Sicherheitsgewinn führen.

radialsymmetrische mit $K(X, Y) = \exp(-|X-Y|^2 / 2\gamma^2)$

und sigmoide Kerne mit $K(X, Y) = \tanh(\gamma\ (X * Y) + c)$
(als typische Aktivierungsfunktion für NNs im engeren Sinne)

Da K apriori gewählt wird, während $\varphi = (\phi_1, \phi_{2,\dots}\phi_k)$ i..a. nicht bekannt ist, wird in diesem Bereich eine Transformation vorgenommen, die zwar - wie oben dargelegt - algebraisch motiviert und funktionalanalytisch legitimiert ist, die aber gerade geometrisch nicht unmittelbar plausibel begründet werden kann.

Man kann demnach für eine SVM die Kern-Funktion K mit

$$K(X, X_i) = \exp(-|X - X_i)|^2 / 2\gamma^2) \qquad \text{(F.2.2-4)}$$

festlegen (Abb. 2.2.2.a), ohne daß es etwa aus der Grundgesamtheit der Daten motiviert wäre, mit einem Gaußschen Verteilungsmodell für den Eingaberaum zu arbeiten (vgl. 2.3.1.).

Es verbleibt (außer der Vorgabe einer Fehlerschranke) auch hier nur *ein* wahlfreier Parameter für den Anwender: γ. Das hat wiederum den Vorteil, daß man keine systematische Variation mehrerer Parameter in tupelweiser Kombination durchführen muß, um die Wirkungsanteile jedes einzelnen sichtbar zu machen.

BURGES weist ergänzend darauf hin, daß ***IH*** i.a. unendlich-dimensional sein wird, wenn ein Kern dieser Art die adäquate Ersetzung bildet, und dennoch eine endlich-dimensionale Lösung berechnet werden kann. [Burge] S. 17

Mittels der Abbildung
h: ***IH*** -> {-1,1}

$$h(X, b) = \mathrm{sign}\left(\sum_i \alpha_i\ y_i * K(X, X_i) + b\right) \qquad \text{(F.2.2-5)}$$

im vorliegenden Fall mit $K(X, X_i) = \exp(-|X - X_i|^2 / 2\gamma^2)$

erhält man dann schließlich die gewünschte Trennfunktion (bzw. statistische Entscheidungsregel).

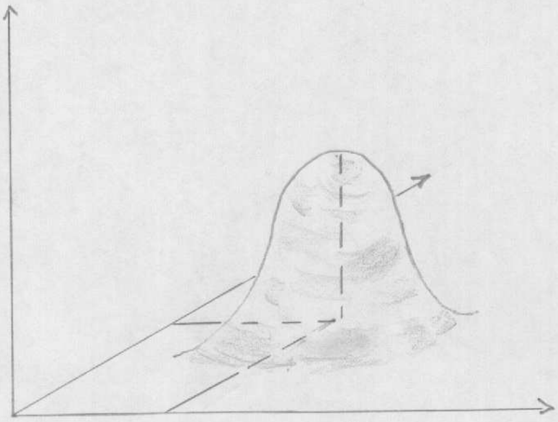


Abb. 2.2.2.a K: ***IR*** -> ***IR*** für $K(\boldsymbol{X},(1,1)) = \exp(- |\boldsymbol{X}-(1,1)|^2 / 2\gamma^2)$
mit $\gamma = 1$

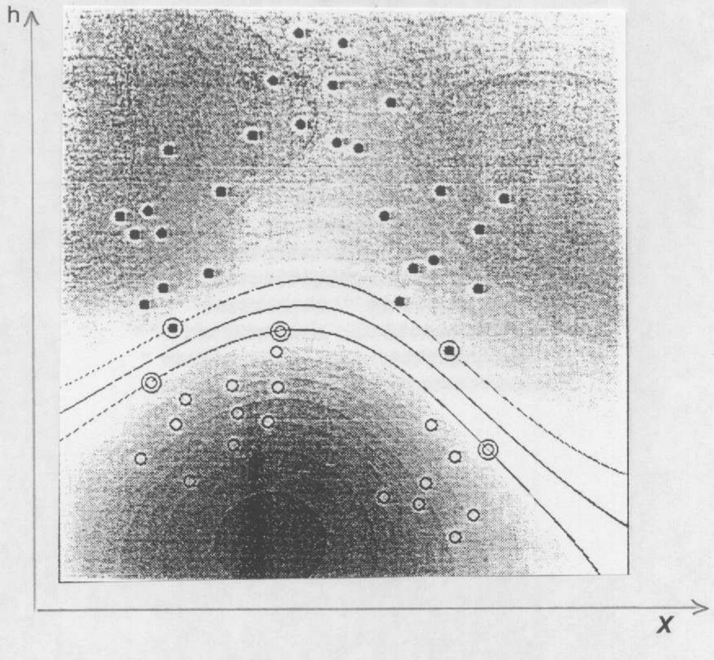


Abb. 2.2.2.b Gestalt der Trennoberfläche h für einen Gauß-Kern
aus [Schö1 S. 44]

Die skizzierte Herleitung wirft die Frage auf, ob es sich bei SVM-Netzen für alle Typen von Kernen um neuronale Netze handelt. **BURGES** sieht innerhalb des Modells einen Paradigmenwechsel, indem mit der Kartierung im Merkmalsraum eine Trennvorschrift expliziert wird, die eine Verkettung von Funktionen im Sinne eines NN realisiert und die ferner durch Stabilisierung zwischen Eingabe und Ausgabe justiert wird [Burge] S. 21 . Gegen diese Sichtweise spricht, daß die Argumentmenge dieser Trennfunktion nicht mehr die Punktmenge ist, aus der die Eingabevektoren (wie bei RBF) entnommen wurden.

Andererseits spricht für **BURGES**' Sichtweise, daß der Ansatz der linearen Separierbarkeit gerade ein Äquivalent in dem Paradigma 'ist durch ein Perceptron berechenbar' findet, also einer Klasse von NNs entspricht.

Da nun die Identität der Argumentmenge über die gesamte Verkettung nicht kategorischer Bestandteil der graphentheoretischen Definition des Begriffes NN ist, soll in diesem Punkt der Argumentation **BURGES**' gefolgt werden, da dies die Vergleichsgrundlage verbreitert.

2.2.3. Stabilisierung und Optimierung

BURGES' Deutung, nach der SVMs sozusagen ab der Testphase als NNs aufzufassen sind, wird durch eine genauere Betrachtung der Lernphase gestützt.

Während das Lernen in RBF-Netzen gewissermaßen einen unipolaren, iterativen Prozeß von einem wählbaren Ausgangspunkt hin zu einem Zielwert, einer Näherungslösung unterhalb einer vorgegebenen Fehlerschranke, beschreibt, vollzieht sich die Konstruktion der optimalen Trennoberfläche als bipolarer Prozeß einer Justierung, für den es eine exakte Lösung gibt (auch wenn in der Praxis eine iterative Näherungslösung implementiert wird). Das quadratische Optimierungsproblem, das die Lagrange-Koeffizienten bestimmt, besteht aus einer Extremwertberechnung unter Nebenbedingungen. Die Netzgenerierung ist nicht für sich genommen steuerbar, sondern ergibt sich sozusagen als Nebenergebnis des Anpassungsprozesses.

Auch dieses Verfahren kann in drei Stufen charakterisiert werden:

(1) Wahl eines Kernes aufgrund von Vorerfahrungen oder Vorannahmen über die Beschaffenheit der Klassifikationsaufgabe: im vorliegenden Fall im Hinblick auf das Vergleichsziel $K(X,X_i) = \exp(-|X - X_i)|^2 / 2\gamma^2)$; Festlegung des Kern-Parameters γ.

(2.1) Festlegung des Stabilisierungs-Datensatzes mit N_d Elementen.

(2.2) Berechnung der N_d Lagrange-Multiplikatoren für

$$W(\alpha_i) = \sum_i \alpha_i - 1/2 \sum_{ij} \alpha_i \alpha_j y_i y_j * K(X,X_i) \qquad \text{(F.2.2-6)}$$

(Abb. 2.2.3. zeigt den dualen Charakter des Extremwertproblems; s.a. ANHANG (4)).

(2.3) Gewinnung der $N_s <= N_d$ Support-Vektoren als diejenigen Elemente aus dem Stabilisierungs-Datensatz, für die $\alpha_i > 0$.

Theoretisch ergibt sich N_s aus der benötigten Dimension des Merkmalsraumes und damit aufgrund der *mercer condition*.

Kleinere α_i ergeben einfachere Trennfunktionen, was Überanpassungen (und damit mangelnde Generalisierbarkeit) vermeidet, schränken aber die Genauigkeit der Lösung ein; größere α_i führen zu Trennfunktionen höherer Validität, aber mit der Gefahr mangelnder Flexibilität gegenüber Erweiterungen in der Grundgesamtheit der Muster [Schö8] (vgl. auch 4.2.3.).

$$f(b,\boldsymbol{W}; \alpha_i) = \tfrac{1}{2}\,|\boldsymbol{W}|^2\ \Sigma_i\, \alpha_i\, y_i\, (\boldsymbol{X}_i * \boldsymbol{W} + b) +\ \Sigma_i\, \alpha_i$$

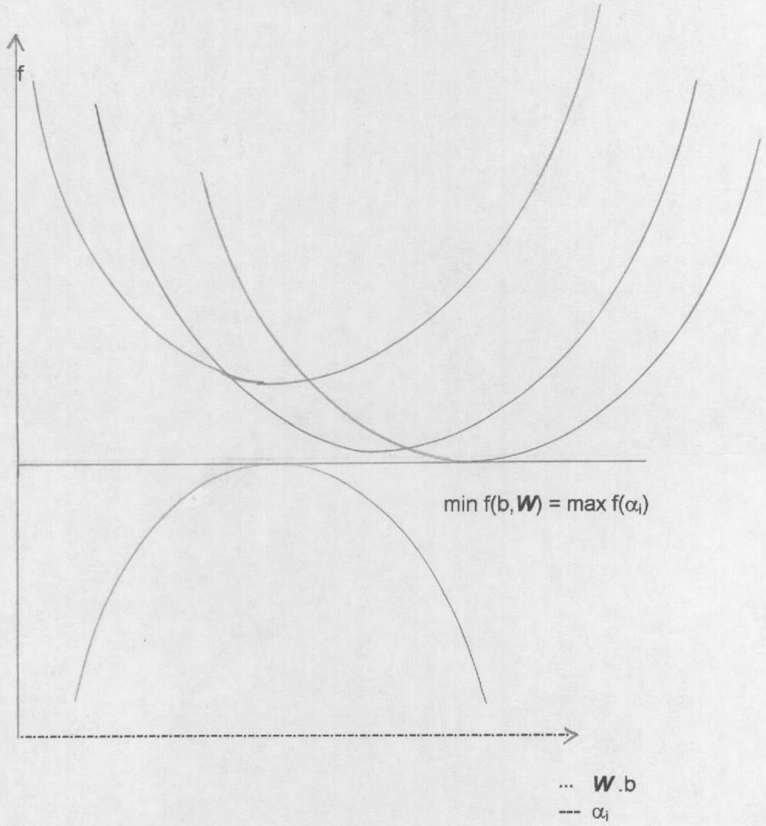


Abb.2.2.3. Duales Optimierungsproblem

(3) Berechnung der Funktionswerte der Anwendung derTrennfunktion auf die Eingabeelemente mittels der Support-Vektoren und den Gewichten entsprechenden Lagrange-Koeffizienten mit den Klassenmarkierungen.

Im SSVM-Simulator werden der LOQO-Optimierer nach der *interior point code* - Methode von **VANDERBEI** [Vande] oder der BOTTOU-Optimierer nach der *conjugate gradient* - Methode von **BOTTOU**, einem der Autoren des SSVM-Manuals [Saund] S. 8f, für diese Aufgabe angeboten.

Wenn **SCHÖLKOPF** beobachtet, daß die Mehrzahl der Support-Vektoren unabhängig von der Kern-Wahl identisch bleibt [Schö3] S. 4, so spricht dies dafür, daß der relevante Abschnitt der Trennoberfläche durch eine relativ einfach gekrümmte Form bereits angemessen modelliert werden kann. Die Beobachtung steht in Einklang mit dem Befund, daß Polynom-Kerne niederen Grades bereits Anpassungen erzielen, die von Polynomkernen höheren Grades kaum noch überboten werden können [Schö3] S.5; d.h. man hat mit SVM ein robustes, universelles Verfahren, das demnach aber möglicherweise nicht die Filigran-Leistungen von RBFs erbringen kann.

2.3. Möglichkeiten, RBF- und SVM- Ansatz zu verbinden

2.3.1. Begründung einer Fusion

Während die spezifischen Stärken beider Verfahren in den vorangehenden Abschnitten bereits benannt wurden, gilt es nun, das Augenmerk auf die spezifischen Schwächen zu richten.

Klassische RBF-Netze verhalten sich empfindlich gegen eine ungünstige, d.h. zu wenig repräsentative Wahl der Stützstellen, was sich insbesondere bemerkbar machen kann, wenn sukzessive die verdeckten Knoten gebildet werden, da klassische RBFs dazu neigen, viel Gewicht auf einzelnen Knoten zu sammeln [Zell1] S. 227, was die Flexibilität beeinträchtigt. Sie arbeiten relativ behäbig unter sehr großen Datenmengen; d.h. die Zahl der benötigten Stützstellen wächst unter Umständen überproportional zur Anzahl der angebotenen Lerndaten. Der Einsatz von RBF-Netzen setzt überdies eine zumindest bereichsweise gaußverteilte Datenmenge voraus, und es ist nicht sicher, daß stets eine globale Optimierung erreicht wird.

Während schwindende Effizienz bei großen Datensätzen auch bei SVM-Netzen zu beklagen bleibt [Burge] S. 35, könnte man erwarten, daß die übrigen genannten Probleme durch die Einbettung eines RBF-Kerns in eine SVM (nachfolgend abgekürzt als SVM(RBF) bezeichnet) eliminiert werden könnten. Es besteht ein Vorbehalt gegen ein solches Vorgehen insofern, als die Entscheidung für ein klassisches RBF-Netz zur Realisierung einer bestimmten Aufgabe auf o.g. grundlegenden Annahmen über die Beschaffenheit und Verteilung der Datengrundmenge im Eingaberaum beruht, während die Wahl eines RBF-Kerns für ein SVM-Netz nicht durch starke, eindeutige Vorbedingungen abgesichert wird, sondern lediglich auf vergleichsweisen Annahmen (z.B. Vorerfahrungen bei ähnlichen Aufgaben) über eine effiziente Umsetzung des in 2.2.2. geschilderten Rechenverfahrens basieren. Dieser Einwand zeitigt umso eher Folgen, als nach der Wahl des Kernes (abgesehen von dem bereits eingeführten Parameter γ) keine internen Steuerungsmöglichkeiten für den Anwender mehr bestehen. Lediglich über die Wahl der Fehlerschranke ist noch Einfluß auf das Verfahren gegeben. - Bei diskreten Daten treten überdies unter Umständen Skalierungsprobleme auf.

Die vorteilhafte Robustheit universeller Verfahren offenbart auf der anderen Seite eben Schwächen in ihrer Daten-Sensitivität. Auch hierzu räumt **BURGES** erheblichen Forschungsbedarf ein [Burge] S. 35 .

Nicht ohne Folgen für die vergleichende Beurteilung bleibt darüber hinaus der Wechsel der Argumentmenge innerhalb der Trenn- Abbildung vom Eingaberaum in den Ausgaberaum, d.h. die Substitution mittels des Kerns wird innerhalb des gänzlich virtuellen Merkmalsraumes angewandt und besitzt (über die Lagrange-Transformation) nur einen mittelbaren Bezug zum ursprünglichen Datenraum, so daß insbesondere die

Eingabevektoren nicht die Argumente im Inneren der verketteten Trennfunktion sind.

Ein weiterer Unterschied besteht darin, daß es in einem klassischen RBF-Netz k Gaußsche Zentrumsfunktionen mit Parametern σ_i ,... σ_k gibt, in einem SVM-Netz nur einen Gauß-Kern mit dem Parameter γ.

Die Wirkung des Kerns im Merkmalsraum ist demnach auf E/A-Ebene im Gegensatz zu einem klassischen RBF-Netz gar nicht ohne weiteres geometrisch interpretierbar. Bei polynomialen Kernen setzt sich f(X) aus nachvollziehbaren Anteilen von $K(X,\mu_1)$,... $K(X,\mu_k)$ zusammen, da die Summe von Polynomen eines bestimmten Grades wieder ein Polynom dieses Grades ergibt. Im Fall eines RBF-Kernes leisten jedoch die $K(X,\mu_i)$ zu f (ϕ_i (X)) einen numerischen Beitrag, der nicht im einzelnen geometrisch darstellbar ist.

Daß die Schaubilder für Trennoberflächen dennoch so suggestiv wirken, hängt zweifellos damit zusammen, daß für die zumeist gezeigten Polynom-Kerne gilt: Die Summe von Polynomen n-ten Grades ergibt wieder ein Polynom n-ten Grades. Analoges gilt eben nicht für die Exponentialfunktion.

Dies läßt **SCHÖLKOPF**s These, RBFs (z.B. mit Gauß-Kernen) seien Spezialfälle von SVMs [Schö2] S. 1, sehr fragwürdig erscheinen, da die Gauß-Kerne weder von der Konzeption her noch in der praktischen Handhabung hier wie dort geometrisch dasselbe repräsentieren. **CAMPBELL** räumt immerhin ein, daß

the method for determining the number of nodes and their centres is quite different from standard RBF networks
[Campb] S. 4f,
dennoch beharrt auch er darauf:

for an SVM with RBF kernels the resulting architecture is an RBF network
[a.a.O.]

Wenn **SCHÖLKOPF** sogar an anderer Stelle für SVM(RBF)-Ansätze in der Performanz signifikante Unterschiede zu klassischen RBFs aufzeigt [Schö3], so erscheint dies ebenfalls als ein weiteres Argument dafür, daß man RBFs nicht als Spezialfall von SVMs betrachten sollte.

Auf dieser Grundlage ist zu prüfen, wie eine Fusion mit Elementen aus dem SVM-Paradigma diese Schwächen ausgleichen bzw. überbrücken kann.

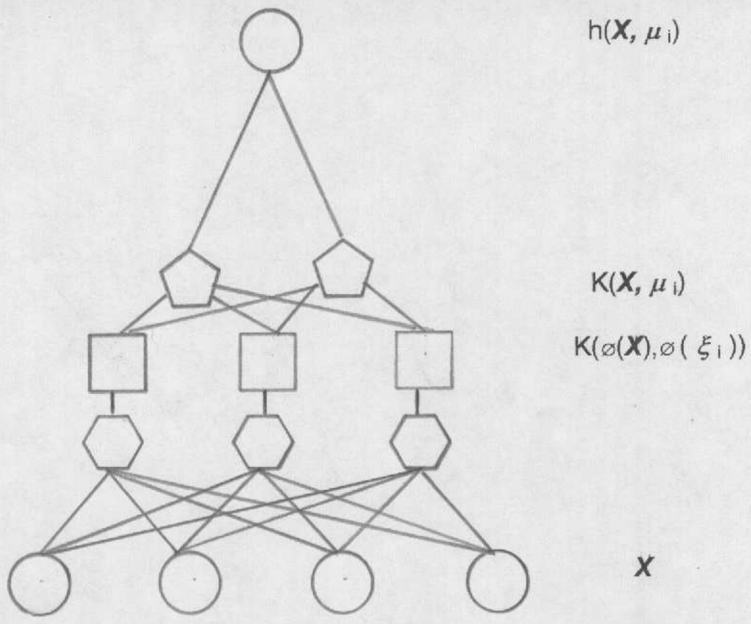


Abb. 2.3.1. Struktur der modifizierten SVM(RBF)-Netze

Man erkennt in Abb. 2.3.1. im Vergleich zu Abb.n 2.1.1. und 2.2.1., daß die grundlegenden drei Stufen Netzstruktur dabei unverändert bleiben, aber im Bereich der verdeckten Schicht wesentlich erweitert werden. Bedeutung und Konsequenzen dieser Erweiterung werden in den nachfolgenden Abschnitten stufenweise vorgestellt.

Um SVM(RBF) als eigenheitlichen Problemlösungs-Ansatz herzuleiten, sei daran erinnert, daß zwischen linearer Separierbarkeit, wie sie SVMs ermöglichen, und Berechenbarkeit durch ein Perceptron eine Äquivalenzbeziehung besteht. **BISHOP** wiederum erwähnt, daß

"radial basis function networks and multi-layer perceptrons play very similar roles in that they both provide techniques for approximating arbitrary non-linear functional mappings between multidimensional spaces"
[Bisho] S. 182

Die einschränkende Ergänzung, daß beide im Detail nennenswerte Unterschiede aufweisen, ist für die hier in Frage stehende Grundsatzentscheidung nicht erheblich. Im Prinzip sichert **BISHOP**s Hinweis die Homogenität beider Verfahren als notwendige Voraussetzung für die Fusionsfähigkeit.

2.3.2. Die verdeckte Schicht des Fusions-Netzes

Trotz der vorgetragenen Einwände kann man aus dem Artikel von **SCHÖLKOPF** u.a. dennoch eine Vorstellung gewinnen, in welcher Richtung eine Verbindung möglich und sinnvoll wäre. Die Autoren stellen ein dreistufiges Verfahren vor, um die Wirkung klassischer RBFs mit SVMs zu vergleichen [Schö2] S.4f . Dabei besteht die **zweite Stufe** der Vergleichskette zwischen RBF und SVM aus einem Hybrid-System, bei dem Support-Vektor-Maschinen die Stützstellen für ein RBF-Netz erzeugen.

Wendet man nun dieses Konzept intern auf eine SVM an, so können die Support-Vektoren einer SVM auf einer **zweiten Ebene** eingesetzt werden, um die Stützstellen eines RBF-Kerns zu bilden (s. dazu 2.3.3.).

Ziel einer solchen Fusion ist es, die beschriebenen datenstruktur-spezifischen Stärken von RBF-Netzen bei der Musterkennung (s. Abschnitt 1.) und die verfahrens-spezifischen Stärken des universellen SVM-Ansatzes (s. Abschnitt 2.) zusammenzuführen, so daß

(>1) eine globale Lösung garantiert wird, weil das Rahmenverfahren, die Bestimmung der Lagrange-Koeffizienten durch quadratische Optimierung, dies gewährleistet;
(>2) die Performanz bei großen Datenmengen verbessert wird, weil die Netz-Anpassung auf zwei Phasen aufgeteilt wird (einem Basisschritt (Ebene I) und einem Ausbauschritt (Ebene II) s.u.);
(>3) die Flexibilität erweitert wird, weil die Anzahl vom Anwender arbiträr beeinflußbarer Parameter unter Verbleib in der standardmäßig benötigten Komplexitätsklasse erweitert wird.

Indem man auf diese Weise die Stärken beider Verfahren nutzt, erreicht man als Komplementäreffekt, daß die unter 2.3.1. aufgewiesenen Schwächen isolierter RBF-Netze oder reiner Standard-SVM-Verfahren neutralisiert oder zumindest abgemildert werden.

Die Fusion wird in folgender Weise realisiert:

I Basisschritt (simultaner Lern-Daten-Durchlauf)

(I.1) In einen SVM-Rahmen wird in der üblichen Weise ein Standard-Kern $K(X, X_i) = X^*X_i$ eingesetzt, um dafür die Support-Vektoren zu erzeugen. Dies ist unter Berücksichtigung des *Mercer*-Theorems möglich und als rein rechnerische Problemtransformation innerhalb der Transformationsfunktion (s. 2.2.2.) zunächst an keine weiteren Bedingungen geknüpft.

II Ausbauschritt (sukzessiver Lern-Daten-Durchlauf)

(II.1) Aus den Support-Vektoren ξ_i werden regionale Zentren (Zentroide)

μi rekonstruiert, auf denen ein RBF-Kern $K(X, \mu_i) = \exp(-|X-\mu_i|^2 / 2\gamma_i^2)$ simuliert wird (s. i.e. 2.3.3.).

(II.2) Der Gauß-Kern wird für jedes Zentroid abschnittweise auf Kovarianzen normiert: $K(X, \mu_i) = \exp(-(X-\mu_i)^T \mathrm{CoVar}_i^{-1} (X-\mu_i)) / 2)$ Dies verändert ihre *Mercer*-Eigenschaft nicht.

Was bedeutet dies im einzelnen?

zu (I.1): Der Ansatz entspricht **SCHÖLKOPF**s Vorgehen: Man nimmt einen SVM-Rahmen und setzt zunächst für die spätere Simulation einer Approximationsfunktion einen einfachen Standard-Kern vom Funktionstyp $K(X,X\mathrm{i}) = X^*X_i$ im Merkmalsraum als Platzhalter ein. Damit wird vorab so wenig wie möglich über die endgültige Beschaffenheit der Trennoberfläche vor-angenommen. Der Kern dient zunächst nur einer vorläufigen Binnenstrukturierung der Trennfunktion; dennoch kann lt. Schölkopfs Beobachtung erwartet werden, daß viele Support-Vektoren identisch denen erzeugt werden (s. 2.2.3.), die auch in der späteren Feinstruktuierung mit komplexeren Kernen das Ergebnis bilden.
Die Zuordnungsvorschrift für die spätere Entscheidungs-/Trenn-Funktion lautet:

Entscheidungsfunktion unter der RBF-Perspektive:
g: ***IH*** -> {-1,1}

$$g(X,w_0) = \mathrm{sign}\,(\textstyle\sum_i w_i\ y_i * K(X,X_i) + w_0) \qquad \text{(F.2.3-1)}$$

bzw.
Trennfunktion unter der SVM-Perspektive:
h: ***IH*** -> {-1,1}

$$h(X,b) = \mathrm{sign}\,(\textstyle\sum_i \alpha_i\ y_i * K(X,X_i) + b) \qquad \text{(F.2.3.-2)}$$

d.h. die Variablen X_i tragen innerhalb dieser Struktur nur indirekt Bedeutung innerhalb der Problemtransformation von $\varnothing$ nach K.

zu (II.1): Die Rolle der Support-Vektoren bleibt allerdings die kritische Schlüsselstelle für den Einsatz von RBF-Kernen, da sie vom Konzept der SVM her gerade den Rand der zu trennenden Cluster markieren, während die Stützstellen eines RBF-Netzes im Gegensatz dazu einen prinzipiellen Bezug zu den (regionalen) Zentren der Cluster besitzen (vgl. 2.2.2. und 2.1.2.).
Zur Überwindung dieses Dilemmas sind verschiedene Ansätze publiziert worden. **HERMES** verwendet zur Lösung eines Mehrgruppen-Klassifikationsproblems ein hierarchisches 2-Gruppen-Klassifikationsverfahren, bei dem er aus Sicherheitsgründen Support-Vektoren konstruiert, die deutlich vom jeweiligen, vorläufigen Teilgruppen-Rand entfernt liegen sollen, um bei weiteren Teilungen nicht zu Konflikten zu führen [Herme] S. 2 . Die Ergebnisse werden später mit Hilfe eines

Bayesschen Klassifikators nachkorrigiert. **ZHANG** beschreibt ein Support-Vektor-Verfahren, das mit regionalen Zentren arbeitet und in der Folge singuläre Ausfransungen an den Rändern von Clustern unberücksichtigt läßt [Zhang] S.7. **TIPPING** schägt ein Verfahren vor, mit Hilfe eines externen Algorithmus sog. relevante Support-Vektoren “R-SV” zu bestimmen, die zu einer allgemeinen Begradigung der Ränder führen [Tippi] S. 6 .

Alle genannten Autoren nehmen bei ihrem Vorgehen zwei Nachteile billigend in Kauf:

> Durch vorgeschaltete oder nachgeführte Sortier- oder Selektions-Algorithmen wird zwar das aufgezeigte Problem erfolgreich unterlaufen, aber die verwendeten Algorithmen sind nicht aus dem ursprünglichen SVM-Basiskonzept abgeleitet, sondern ausschließlich durch ihre praktische Wirksamkeit legitimiert, und behalten insoweit eine gewisse Beliebigkeit.

> Indem der ordnende Zugang zu der Datenverteilung von den Rändern der Cluster wegorientiert wird, wird nicht mehr zu Beginn allen Merkmalssyndromen die gleiche Gültigkeit, das gleiche Gewicht innerhalb der Grundgesamtheit zugestanden, sondern es wird (z.T. explizit) unterstellt, daß die Ränder der Cluster als eher zufällig angesehen werden können, damit weniger interpretierbare Bedeutung besitzen und begradigt, korrigiert oder vernachlässigt werden können. Diese Annahme steht der stillschweigenden Voraussetzung der Gleichverteilung der Daten und einer kontinuierlichen Dichtefunktion entgegen (s. a. 3.2.3.).

Eine Randbemerkung in **SCHÖLKOPF**s Dissertation von 1997 motiviert an dieser Stelle einen anderen Weg, wie man die tragenden regionalen Zentren zu den konstruktiven markanten Randpunkten plausibel in Beziehung setzen kann. **SCHÖLKOPF** führt aus:

"*It is interesting to note that the solution has a simple physical interpretation [..]. If we assume that each Support Vector z_j exerts a perpendicular force of size α_j and sign y_j on a solid plane sheet lying along the hyperplane $w * z + b = 0$, then the solution satiesfies the requirements of mechanical stability*." [Schö1] S. 39

Er weist besonders darauf hin, daß die Summe der Drehmomente (und damit auch die Summe der einwirkenden Kräfte selbst) insgesamt = 0 sein muß (was in den Nebenbedingungen der Trennfunktion auch gesichert ist), so daß diese Analogie die Bedeutung des Begriffes 'Support Vektor' augenfällig veranschaulicht. Schölkopf unterlegt also den Clusterwolken von Merkmalssysndromen physikalische Masseeigenschaften und interpretiert die Bedeutung der Projektionen von Clusterpunkten auf die Hyperebene als das Einwirken mechanischer Kräfte, die gerade so zueinander in Beziehung stehen, daß eine stabile Lage der (Hyper)Ebene erreicht wird.

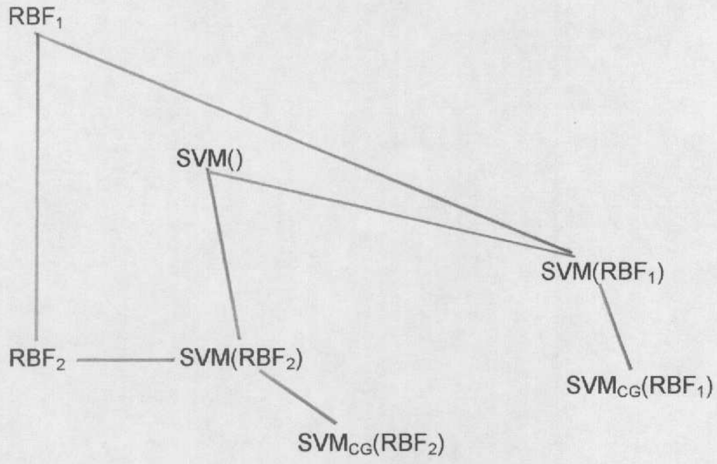


Abb. 2.3.2a Bausteine der Fusion

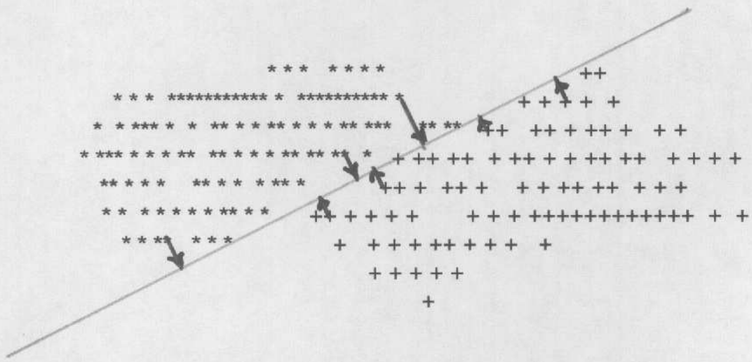


Abb.2.3.2.b Hyperebene und mechanische Stabilität

Eine solche Deutung erscheint umso mehr als plausibel und hilfreich, wenn man bedenkt, daß in der schließenden Statistik häufig entsprechende metaphorische Anleihen üblich sind, wenn man etwa von 'auslösenden Impulsen', 'Einflüssen von Gegenkräften' oder dem 'Gewicht von Faktoren' gesprochen wird.

Beide Paradigmen berühren sich gewissermaßen im Begriff der Dichtefunktion, der eine kontinuierliche Verteilung der physikalischen Masse bzw. der statistischen Grundgesamtheit voraussetzt und dann deren Verteilungs-Eigenschaften beschreibt.
Daraus ergibt sich wiederum anschaulich korollar, daß stets $N_S >= N_Z$ sein wird, da man jeden bekannten Randpunkt über den Vergleich der Varianzen *within* und *between* genau einem Zentrum zuordnet, welches aber für verschiedene Randpunkte natürlich identisch sein kann. Hier kommt man demnach zu einem anderen Ergebnis als bei **SCHÖLKOPF**s Ansatz von SVM(RBF), bei dem prinzipiell

the number of nodes [is] equal to the number of support vectors and the centres of the RBF nodes identified with the support vectors themselves
[Campb] S. 5

Denkt man die Analogie fort, nach der in der statistischen Grundgesamtheit wie in einer physikalischen Masse Kräfte zur Geltung kommen, so entsprechen die regionalen Zentren der Cluster regionalen Massenschwerpunkten von unregelmäßigen Körpern (etwa wie von Gesteinsbrocken), die von den zugänglichen Rändern her ebenso experimentell bestimmt wie theoretisch identifiziert werden können (s.2.3.3.). Die einwirkenden Schwerkräfte hängen dabei insbesondere quadratisch vom Abstand zu den Rändern ab, was eine Entsprechung in der statistischen Varianz findet, die als Maß für den Grad der Abweichung vom Mittelwert ebenfalls quadratisch vom Abstand zum Zentrum der Verteilung abhängt. Man brauchte somit lediglich den Kehrwert der Varianz als äquivalentes Maß für die Mittelwertsbindung zu definieren, um eine unmittelbare Entsprechung zur Wirkung der Schwerkraft zu beschreiben.

Selbst wenn man die vorausgehende Plausibilitätsbetrachtung als nicht zwingend für das angestrebte Vorgehen betrachtet, bleibt dennoch als hinreichendes Argument seine Zulässigkeit. In einem Lehrbuch der Physik heißt es dazu:

Natürlich weiß man noch nicht, ob das ganze Verfahren [zur Bestimmung eines Massenschwerpunktes] physikalische Bedeutung hat. Selbst wenn es keinen tieferen physikalischen Sinn hätte, so ist es doch jedenfalls in sich zulässig und einwandfrei; auf alle Fälle haben wir einen [interpretierbaren] geometrischen Punkt im System festgelegt.
[BerSc] S. 90

Entscheidend ist also, daß an die Stelle von Schölkopfs SVM(RBF)-Kern

$$K(X, \xi_i) = \exp(-|X - \xi_i|^2 / 2\gamma^2)$$

jetzt der SVM_{CG} (RBF)-Kern

$$K(X, \mu_i) = \exp(-|X - \mu_i|^2 / 2\gamma^2)$$

treten wird.

zu (II.2): Die Normierung auf Kovarianzen erhöht gemäß den Ausführungen von Bishop (s. 2.1.2.) den Spielraum für Vorgaben seitens des Anwenders und verringert potetiell die Anzahl der benötigten Stützstellen, weil die Ausstrahlung einer Stützstelle genauer und spezifischer modelliert werden kann als beim radialsymmetrischen Ansatz.

Damit entstehen aber nicht analog zu einem RBF-Netz eigenständige Kerne $K_1,...K_k$, die jeweils eine Stützstelle repräsentieren würden, sondern der universelle Kern K wird über die Kovarianzen gewissermaßen abschnittweise definiert. Damit muß auch nicht mehr vorausgesetzt werden, daß sich die Ränder der Clusterwolke in der Umgebung eines jeden Support-Vektors als Stützstelle in gleicher Form abgrenzen lassen.

Eine Rückabbildung in den ursprünglichen Eingaberaum findet nicht statt. Aus **SCHÖLKOPF**s Artikel “Input Space vs. Feature-Space in Kernel-Based Methods” wird deutlich, daß ϕ_i nicht umkehrbar ist (vgl. [Schö7] *'the pre-image problem'* S. 7). Sie ist aber auch für die weitere Interpretation nicht erforderlich, da für die isolierte Teilbetrachtung des eingebetteten RBF ja gilt: Eingaberaum = Ausgaberaum (= Merkmalsraum).

2.3.3. Die Schritte der Lernphase

Wie gewinnt man also die (unbekannten) Zentroide μ_i aus den berechneten Support-Vektoren ξ_i ?

Man vergegenwärtige sich noch einmal den inneren Bezug zwischen ξ_i und μ_i anhand des prinzipiellen Unterschiedes in der Fehlerbehaftung von RBF-Netzen und SVM-Netzen. Während der strukturelle Fehler gerade auf dem Ansteuern einer näherungsweisen Lösung beruht, kommt der empirische Fehler durch die Randbedingungen des Trainings zustande. Die Vorgehensweise bei RBF-Netzen ist dabei ein Spezialfall des Konzeptes, das Vertrauensintervall fest zu halten und auf dieser Grundlage den empirischen Fehler über die Varianz zu minimieren. Bei den SVM-Netzen hingegen wird gerade der empirische Fehler festgehalten mit der Maßgabe, das Vertrauensintervall zu minimieren.

Der unterschiedliche Ansatz hat u.a. zur Konsequenz, daß die Stützstellen eines RBF-Netzes und die Support-Vektoren eines SVM-Netzes unter einer unterschiedlichen Perspektive generiert werden: Während die Support-Vektoren randständige Trennstellen markieren (Prinzip der Wasserscheide), repräsentieren die Stützstellen gerade charakteristische, zentrale Merkmalspunkte einer Klasse (Prinzip der Quelle). Es ist daher nicht zu erwarten, daß bei der Gegenüberstellung eines RBF-Netzes und einer SVM(RBF) für ein und dieselbe Klassifikationsaufgabe die Stützstellen μ_i und die Support-Vektoren ξ_i identisch bestimmt werden, zumal wenn dies im klassischen RBF-Netz sukzessive und in der SVM simultan geschähe.

Das vorgesehene Lernverfahren für die Fusion SVM_{CG} (RBF) läßt sich (in der Numerierung von 2.3.2.) folgendermaßen beschreiben:

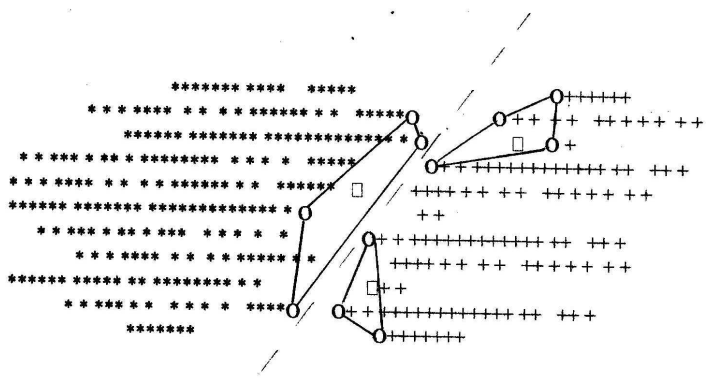
(I) Es wird ein Rahmen für einen Standard-Kern gebildet.

(I.1) Im vorliegenden Fall zunächst $K(X, X_i) = X^* X_i$;

(I.2) Bestimmung der Gewichte durch die Lagrange-Multiplikatoren für
$W(\alpha_i) = \sum_i \alpha_i - 1/2 \sum_{ij} \alpha_i \alpha_j y_i y_j * K(X,X_i)$
(Lösung des exogenen Extremwertproblems)

(I.3) Gewinnung der Support-Vektoren ξ_i aufgrund der $\alpha_i > 0$.

(II) Zu den N_S Support-Vektoren werden mittels des Algorithmus **cent_grav**() (s. ANHANG (5)) regionale Zentroide μ_i berechnet. Mittels **cent_grav**() wird Schölkopfs Hinweis auf die mechanische Analogie weitergehend ausgeführt (Abb. 2.3.3.):



O möglicher Supportvektor ξ_i; in der Umgebung *** : ξ_i^+
in der Umgebung +++ : ξ_i^-

□ mögliches Zentroid
(Konvergenz-Grenzwert von **cent_grav()**)

Abb. 2.3.3. Polygonzüge durch Clusterwolken

(II.1) Aus den ermittelten Support-Vektoren bildet man einen Polygonzug, den man in alle permutativ möglichen Teildreiecke zerlegt.

(II.2) Für jedes Teildreieck wird der geometrische Schwerpunkt, also der Schnittpunkt der Seitenhalbierenden berechnet.

(II.3) Aus den auf diese Weise gewonnenen Schwerpunkten wird ein neuer Polygonzug gebildet, mit dem gemäß (II.1) weiterverfahren wird.

Zunächst läßt sich feststellen, daß dieses Verfahren konvergieren muß, da der Schnittpunkt der Seitenhalbierenden eines Dreiecks (etwa im Gegensatz zum Schnittpunkt der Mittelsenkrechten) stets per Definition der Seitenhalbierenden innerhalb des Dreiecks liegen muß.

Zur physikalischen Deutung dieses Prozesses läßt sich folgendes feststellen: Für jedes Teildreieck eines Polygonzuges gilt, daß das räumliche Gravitationszentrum orthogonal zur Dreiecksfläche auf einer Geraden durch den Schnittpunkt der Seitenhalbierenden liegt. Die orthogonalen Geraden zu den verschiedenen Teildreiecken werden sich i.a. im n-dimensionalen Raum nicht schneiden.

Konvergiert der Polygonzug aber, so schneiden sich im Grenzfall alle Orthogonal-Geraden in eben diesem Konvergenzpunkt, so daß dieser das gesuchte Gravitationszentrum μ darstellt.

Im Sinne der Stützstelleneigenschaft wird man sich nun i.a. nicht für das globale Gravitationszentrum der Clusterwolke von Merkmalsvektoren interessieren, sondern für partielle, regionale Zentren (Zentroide) von besonders ausgeprägten Bereichen der Clusterwolke. Um solche (virtuellen) Zentroide zu gewinnen, ist es für **cent_grav**() möglich, die Grundmenge aller Supportvektoren in k disjunkte Teilmengen aufzuspalten, aus denen dann k Ausgangs-Polygonzüge gebildet werden, mit denen gemäß (II.1) -(II.3) verfahren wird.

Auf diese Weise enstehen nach Konvergenz k Zentroide μ_i.

Geht man also in (II.1.) nicht von *einem* Polygonzug aus, der sämtliche Support-Vektoren erfaßt, sondern bildet mehrere Polygonzüge aus disjunkten Untermengen der Support-Vektoren, so nähert man (virtuelle), lokale Zentroide μ_i (potentielle spätere Stützstellen) anstelle des absoluten Gravitationszentrums; d.h. ähnlich wie beim *chunking* für den Lern-Daten-Satz kann die Klassifikationsaufgabe in Teilaufgaben zerlegen, von denen jede potentiell in einer späteren Stützstelle repräsentiert wird.

Welche Support-Vektoren ξ_i man dabei zu einer dieser Untermengen zusammenfaßt, bleibt exogenen Vorannahmen über lokale Besonderheiten in der Menge der Lerndaten vorbehalten. Insbesondere ist es nicht prinzipiell erforderlich, positive ξ_i^+ und negative Support-Vektoren ξ_i^- getrennten Untermengen zuzuweisen, da durchaus ein positives Zentroid μ_i^+ des A-Clusters Wirkung auf eines negatives μ_i^- des A^c-Clusters ausstrahlen kann und umgekehrt (s. dazu die Fallunterscheidung zum DDA-Algorithmus 2.1.3.). Man hat also die Wahl, diese mögliche

Ausstrahlung zu berücksichtigen oder die Vorinformationen, die aus der in dieser Phase bereits ermittelten Zuordnung von ξ_i^+ und ξ_i^- hervorgehen, als Begründung dafür zu nutzen, diese möglichen Ausstrahlungen nunmehr auszublenden. In beiden Fällen bedarf es aber keiner u.U. ergebniskritischen Annahmen zur Initialisierung wie bei anderen Clusterverfahren (vgl. KMEANS).

Da man mindestens ein Zentroid μ_i^+ und ein Zentroid μ_i^- benötigt, ist das Verfahren für $N_s >= 4$ sinnvoll einsetzbar.

(II.3) Die μ_i i=1...$N_z <= N_s$ ($N_s>=4$) bilden den virtuellen Daten-Satz (in iterativen *work-sets*) , für das der Gauß-Kern $K(X, \mu_i) = \exp(-|X-\mu_i|^2 / 2 \gamma^2)$ simuliert wird. Alternativ werden die Argumente wahlweise auf Kovarianzen normiert: $K(X, \mu_i) = \exp(-(|X-\mu_i|^T CoVar_i^{-1} |X-\mu_i|) / 2)$.

(II.4) Bestückung der Trennfunktion h mit den neu gewonnenen Support-Zentroiden und den zugehörigen Lagrange-Koeffizienten.

Das additive Korrekturglied b läßt eine zweifache Interpretation zu, wenn man die Analogie in Trennfunktion und Entscheidungsfunktion nutzt:
+ w_0 : additiver bias-Term, der eine durchgängige Differenz zwischen Modell und Daten-Realität markiert;
+ b : geometrischer Abstand vom Ursprung, dem gesetzten Bezugspunkt für das System.

3. Kennzeichen des Gegenstandsmodells Gesichtererkennung

3.1. Identifikation von Gesichtern als Beispiel für visuelle Mustererkennung

3.1.1. Wahrnehmungs- und Unterscheidungsleistungen

In der Einleitung wurde bereits dargelegt, daß für die Erkennung von Gesichtern gute Möglichkeiten bestehen, aus dem menschlichen Verhalten bei eben dieser Aufgabe Einsichten in das grundsätzliche Leistungsvermögen von NNs zu gewinnen. Um die Arbeit eines Netzes, das menschliche Gesichter erkennen soll, zu beurteilen, ist es hilfreich, sich zunächst zwei grundlegende wahrnehmungspsychologische Voraussetzungen zu vergegenwärtigen, die zum einen das Zusammenwirken von biologischem Wahrnehmungsapparat und zerebraler Kognition sowie zum anderen Stufen evolutionärer Optimierungen in diesem Bereich betreffen.

Es gehört offenbar zu den Leistungen der Evolution, die Verarbeitung optischer Informationen je nach ihrer mutmaßlichen Bedeutung für ein Individuum auf verschiedenen zeitlichen und hierarchischen Stufen zu organisieren, etwa die unter Umständen bedrohliche Geschwindigkeit eines beweglichen Objektes vor den Einzelheiten seiner Form zu analysieren (vgl. *low level vision* vs. *high level vision*). In diesem Zusammenhang wäre zu fragen, ob unterschiedliche Netztypen wie RBF und SVM diese jeweiligen Teilaufgaben unterschiedlich gut bewältigen können.

Da in der Entwicklung von Verfahren zur visuellen Objekterkennung sehr unterschiedliche Wege beschritten worden sind, ergeben sich aus dem Vergleich der Ergebnisse weitreichende Rückschlußmöglichkeiten. Während sich das natürliche visuelle Objekterkennen evolutionär verändert hat, d.h. nicht auf jeder Stufe alle biologischen Resourcen (z.B. Zapfen und Stäbchen) verfügbar waren, die heute beim Sehen und Wahrnehmen zusammenwirken, sind in den einzelnen Stadien mutmaßlich auch die jeweiligen Optimierungsprozesse des Gehirns anders verlaufen, als sie bei kompletter Verfügbarkeit aller heutigen Resourcen auf allen evolutionären Stufen verlaufen wären.

Insofern ist es nicht verwunderlich, daß in der Robotik bei Ansätzen ohne natürliche Resourcenbeschränkung sich bisweilen Systeme als besonders effizient erwiesen haben, die nicht den Ergebnissen der biologischen Evolution entsprechen, so zum Beispiel, daß Roboter mit mehr als zwei Kameraaugen und damit entsprechend mehr Bildinformation aufgrund geringeren Verrechnungsbedarfs zu schnelleren Ergebnissen im Bereich des räumlichen Sehens gelangen als Roboter mit zwei Kameraaugen.

Was das Zusammenwirken des Wahrnehmungsapparates und kognitiver Verarbeitung betrifft, sind dabei ähnlich wie in der Neurologie Fälle aufschlußreich, bei denen genetische Einschränkungen, pathologische Defizite und umweltbedingte Prägungen im Vergleich Einsichten in die normalen Funktionalitäten offenbaren. Zu diesem Feld gehört in der vorliegenden Domäne die Analyse von optischen Täuschungen (Abb. 3.1.1.).

Sie zeigen, daß das Gehirn anderes oder auch mehr wahrnimmt, als die Augen tatsächlich sehen, d.h. daß Vorinformationen verarbeitet werden. Ein alltägliches und gänzlich unpathologisches Beispiel hierfür bildet im Bereich der visuellen Wahrnehmung die interne 3D-Rekonstruktion verdeckter Oberflächen.

Beachtenswert erscheint in diesem Zusammenhang, daß bei der Arbeit eines NNs keine optischen Täuschungen 1. Ordnung (z.B. die Neigung zu umgebungsbedingter Komplementärfarben-Kontrastierung oder die Neigung, Längen umgebungsbezogen zu proportionieren) auftreten können, sehr wohl aber optische Täuschungen höherer Ordnung (vgl. [ZekiS] S. 62; [CriKo] S.151), bei denen semantische Interpretationsleistungen zu erbringen sind (z.B. bei Vexierbildern). Das Netz kann an derartigen Leistungen systematisch scheitern und auf diese Weise erkennen lassen, an welchen Stellen und in welcher Form Vorwissen explizit oder implizit eingesetzt wird und auf welcher Grundlage die Interpretationsleistung erbracht wird.

Derartige Schwierigkeiten erinnern wiederum an vielfältige Kompensationsmechanismen, wie sie auf einer anderen Ebene in der neurologischen Rehabilitationsmedizin immer wieder zu beobachten sind. Es gibt etwa Raupen mit sog. Lichtfleckaugen, deren unmittelbares Sehvermögen sich darauf beschränkt, anhand eines Lichtstrahls Helligkeitswerte zu unterscheiden, die aber durch Bewegung und die Auswertung der daraus gewonnen Unterschiede im Raster der Helligkeitseindrücke erstaunliche Leistungen im Bereich der Mustererkennung erbringen können.

Da derartige Spezifikationen evolutionär stets dem Ziel dienen, das Überleben einer Population der Art zu sichern, kann in der Entwicklung eine Optimierung von Leistungen erwartet werden, so z.B. wenn die Sehwerkzeuge von Amphibien darauf abgestimmt sind, vorzugsweise solche Objekte zu erkennen, die sich kontrastiv vom jeweiligen Hintergrundschema abheben. Man vergleiche hierzu NN-Verfahren zur vorgezogenen Kantendetektion.

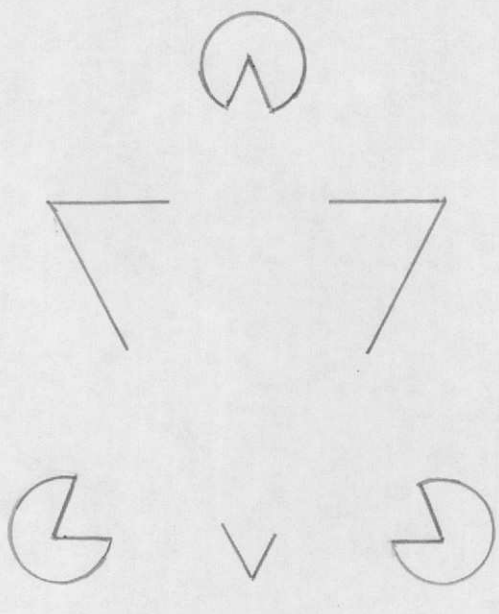


Abb. 3.1.1. Variante des KANIZSA-Dreiecks

3.1.2. Rahmenbedingungen von Erkennung

Über die prinzipiellen Überlegungen von 3.1.1. hinaus gibt es wahrnehmungspsychologische Indizen dafür, daß die Verarbeitung visueller Information im menschlichen Gehirn nicht einem einheitlichen Konzept folgt, sondern in Abhängigkeit von Lebensalter (bzw. Seh-Erfahrung), Vertrautheitsgrad mit der Umgebung, aber auch aktueller mentaler Disposition oder Antizipation von Erwartungen (selektive Wahrnehmungen einer Situation bei Männern und Frauen, soziale Erwünschtheit einer Handlung) auf ganz unterschiedliche Weise geleistet wird. Eine solche Annahme würde wiederum nahelegen, unterschiedliche Netzstrukturen je nach Bedarf in der unter 2.3. und 4.3. vorgeschlagenen Weise zu kombinieren, zumindest jedenfalls im Versuchsaufbau dafür zu sorgen, daß verschiedene Konzepte unter wechselnden Vorgehensweisen im Vorfeld gegeneinander getestet werden.

Eine Schattierung auf einem ebenen Blech wird vom Betrachter nicht unbedingt sofort als erhabene Beule identifiziert werden, eine Schattierung in der Mitte eines Gesichts jedoch sofort als Erhebung der Nase attribuiert werden.

Ein Beispiel aus der in Forschungsberichten häufig zitierten Identifikation handgeschriebener Ziffern, wobei dem Sinne nach zehn Schablonen gelernt werden, von denen die wahrscheinlichste als Zuordnung gewählt wird: Es kann sehr wohl zu einer Fehlklassifikation zwischen 8 und 9 kommen, wenn der untere Ziffernbogen nahezu geschlossen wirkt, es ist aber vom Ansatz her unmöglich, eine 8 mit einem B zu verwechseln (vgl. [CorVa] S.279-282). Abb. 3.1.2. zeigt, daß selbst eine mit einem zeitweise aussetzenden Kugelschreiber erstellte, deutlich verunglückte Handschriftversion eines Musters zur Klassifikation in einem auf Zahlen fixierten Umfeld nicht, wie es naheliegend wäre, als “b” interpretiert werden kann, sondern trotz der vergleichsweise geringen Übereinstimmung nur mit Vorlage (1) als “5” oder mit Vorlage (2) als “3”.

Um die Grenzen zwischen “eher gleichartig” und “eher nicht gleichartig” unter den jeweiligen Rahmenbedingungen zu lernen (Störeinflüsse, sog. Rauschen), brauchen RBF-Netze in besonderer Weise Varianten eines in Frage stehenden Musters, in der vorliegenden Domäne Bilder mit unterschiedlichen Gesichtsausdrücken einer Person.

Abb.3.1.2 Beispiel Erfassungsraster Ziffernschema

3.1.3. Kontext-Implikationen

Offensichtlich gehen in alle Entscheidungen, die im Zusammenhang mit einer Netzsimulation getroffen werden, von der Aufbereitung des Datenmaterials bis zur Auswahl eines Lernverfahrens, von der Formulierung einer Testhypothese bis zum Festlegen der Netzparameter bewußte oder unbewußte Annahmen oder Vorkenntnisse über den Untersuchungsgegenstand ein. Je umfassender es gelingt, diese Implikationen offenzulegen oder zu rekonstruieren, desto angemessener werden die Ergebnisse ausfallen und desto sicherer werden darüber hinausgehende Schlüsse ausfallen können. Als Beispiel sei die Analyse der Unterschiede zwischen α- und ß-Fehlern (Fehlern 1. und 2. Art) angeführt, die für viele Anwendungen von zentraler Wichtigkeit ist (s.. 4.3.1.)

Betrachtet man derartige Implikationen unter der Perspektive der Simulations***phase***, zu der sie wirksam werden (diachroner Schnitt), so ergibt sich naheliegend eine Unterscheidung in

- **apriori**-Wissen
grundsätzliche Annahmen über Invarianzen unter geometrischen Operationen, 3D-Sichten u.ä. linear zusammenhängende Bereiche, typische Schattenbildungen, Hell- Dunkel-Verteilungen usw.
> findet seinen Niederschlag in Parametern der Klassifikationsfunktion, Wahl einzelner Funktionsparameter, des Lernalgorithmus o.ä.

- **kontemporär**-Wissen
betrifft empirisches Vorwissen über die Beschaffenheit der Trainingsdaten, Trennfunktion
> findet seinen Niederschlag in deren Auswahl und Anordnung

- **aposteriori**-Wissen
etwa Plausibilitätsüberlegungen zu vorläufigen Ergebnissen, kritische Zuordnungen, Interpretation von Fehlern

d.h. die CPU-Zeiten für NN-Simulationen im Millisekundenbereich sollten nicht den Blick dafür verstellen, daß es sich dabei nicht um eine Momentaufnahme eines Netzzustandes handelt, sondern daß jede Simulation mehrere Zustände durchläuft, auf die von außen Einfluß genommen werden kann und die eine unterschiedliche Interpretationsperspektive ermöglichen. So sind auch die in 2.3.2. eingeführten zwei Ebenen unter der diachronen Perspektive eigentlich als zwei Phasen zu interpretieren.

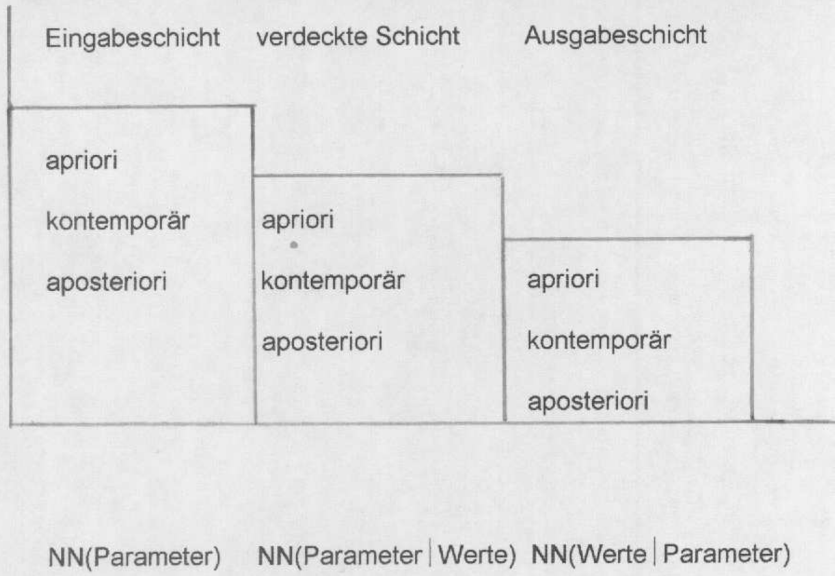


Abb.3.1.3. Diachrone vs. synchrone Perspektive

Diese diachrone Sicht auf die Simulation erinnert natürlich nicht zufällig an die Definition von apriori-, bedingten und aposteriori-Wahrscheinlichkeiten, womit im Rahmen des BAYES-Theorems ein übergreifendes Interpretations-Paradigma eröffnet wird (vgl. dazu 4.1.2.).

Unter der Perspektive der jeweils aktuellen Handhabung und Qualität der Implikationen (synchroner Schnitt) sollen zwei Arten von Kontext-Implikationen unterschieden werden, die als *Intern*Wissen und *Extern*Wissen bezeichnet werden sollen. *Intern*Wissen ist danach explizit oder implizit im Netz vorhanden, z.B. durch die jeweilige Beschaffenheit grundsätzlich wahlfreier Parameter oder in der Semantik der Algorithmen; *Extern*Wissen wird von außen explizit oder implizit implementiert, z.B. in Form von Initialisierungen des Lernprozesses oder durch die Selektion des Trainingsdatenmaterials.

Diese Unterscheidung orientiert sich also an den Möglichkeiten einer Einflußnahme auf den 'Wissenserwerb'. Die Vorstellung von *Intern*Wissen des Netzes ist damit in gewisser Weise angeborenem Wissen im biologischen Bereich vergleichbar, Externwissen gelehrtem bzw. unspezifisch erlerntem.

Beim *Extern*Wissen wäre gelehrtes Wissen eher symbolisch codiert, diffus gelerntes Wissen subsymbolisch. Bei der visuellen Objekterkennung wären etwa Erkenntisse über die übliche Grundform eines Gesichts und die erwartbaren markanten Erhebungen subsymbolisch in den Gewichten der verdeckten Knoten codiert, Schablonen von Vergleichsgesichtern hingegen symbolisch in Form von Vorlagen.

Auf algorithmischer Ebene könnten wiederum sehr unterschiedliche Arten von *Intern*Wissen transportiert werden, indem
- die Bildgewinnnung auf bestimmten *canonical views* basiert wird
- Vereinfachungen aufgrund von Invarianzen unter vektoriellen Operationen vorprogrammiert werde oder mit Hilfe von Invarianzen die Zahl unterschiedlicher Muster gleicher Zuordnung zu erhöhen (VSV, s. dazu 4.1.3.), d.h. die Palette denkbarer Varianten besser zu überdecken
- die Registrierphase für die Bilddaten mit einer vorgeschalteten Kantendetektion beginnt
- die Verwendung virtueller Supportvektoren die Konstruktion der trennenden Hyperebene verfeinert.

Man kann solche Kontext-Implikationen nutzen, indem zu einem Vektor (Muster) virtuelle Zusatzvektoren generiert werden, z.B. wenn man weiß, daß ein Muster nach Drehung, Spiegelung, Translation o.ä. wiederum derselben Klasse angehören würde. Auf diesem Wege geht zusätzliches Vorwissen in das Training ein und verstärkt die Genauigkeit und Generalisierbarkeit der Klassenbildung; in anderer Weise durch Verfahren des Nachtrainings oder der Nachjustierung.

Damit müssen auch die Aussagen zur Güte einer Netzanpassung über die üblichen Fragen nach *model fit* und *model complexity* erweitert werden um die Frage, wie gut die abstrakt erzielte Anpassung dem tatsächlichen Gegenstandsmodell entspricht, d.h. inwieweit ein Befund im Rahmen des Gegenstandsmodells sinnvoll interpretierbar ist. Dieser Bereich sei als *model propriety* bezeichnet (s. 3.3.3. und insbesondere 4.3.2.).

3.2. Der Untersuchungsplan

3.2.1. Beschreibung der Bilddatenbank

Für eine vergleichende Simulation zur Leistungsfähigkeit bei der Gesicht**s**erkennung (*face-detection*) wird naturgemäß eine andere Datenbasis benötigt als für Simulationen zur Gesicht**er**erkennung (*face-recognition*). Zur Kontrolle der Rahmen-Parameter erschien es jedoch wünschenswert, für alle Fragestellungen einen übereinstimmenden, gemeinsamen Datenkern zu haben, um verdeckte Einflüsse aus dem Datenformat ausschließen zu können. Dazu wurden folgende Grundannahmen herangezogen:

(1) Eine Anzahl von 40-80 Bildern (Mustern) gilt als vielversprechende Grundgesamtsamt für die Exploration eines Verfahrens (vgl. **ZELL1**).

(2) Die Anzahl von Frauen und Männern unter den Subjekten sollte identisch sein; die Zugehörigkeit zu den vordefinierten Altersgruppen sollte gleichmäßig verteilt sein.

(3) Die Bilder sollten bzgl. der Darbietungskriterien normiert sein, und zwar in Anlehnung an Überlegungen von **BELHUMEUR U.A.** [Belhu] S. 712 in folgender Weise:

(3.1) gleichbleibende Kameraposition und Justierung (Abstand, Aufnahmewinkel (angle of view), Focus, Blende, Belichtungszeit); daraus folgend gleiche Größe (Skalierung) und Lokalisierung des Gesichtes im Bild, ebenso gleiche pixel-Zahl pro Bild (Körnung)
(3.2) normierte Blickrichtung der aufgenommenen Person (2D frontal) [PonVe] S. 640f unter zufälligen Varianten eines emotional neutralen, entspannten Gesichtsausdrucks (Paßfotostandard)
(3.3) identische Beleuchtungsquelle(n) [Illumination] (Stärke, Farbtemperatur, Intensität, Positionierung) und Hintergrund (Textur), daraus folgend unter Berücksichtigung von (3.2) gleichartige Schattenformationen
(4) Von jedem Subjekt sollte es mindestens 6 Bildvarianten im Sinne von (3.2.) geben, die jeweils ein *set* bilden, das in Trainings- und Testungs-Datensatz aufgeteilt werden kann. Den plausiblen Erfahrungen erkennungsdienstlicher Handhabung folgend, können sämtliche Bestandteile der Frisur nur bedingt berücksichtigt werden.

Bei Brillenträgern müssen konsequenterweise Bilder mit und ohne Brille abgeglichen werden.

In Bezug auf die in Frage stehenden 2D-Momentaufnahmen waren *event*-abhängige Überlegungen zu Bewegung und Visibilität, wie sie in der Objekterkennung nötig sind, nicht erforderlich.

Für die Aufgabe der Gesichtserkennung werden in identischer Größe graphisch verfremdete und generisch neue Bilder eingebracht.

Für die Anwendungsphase kommen zusätzlich simulierte Verfremdungen wie Spiegelungen, Fehlfarben, Stauchungen oder vergleichbare Manipulationen eines Gesichtes als Vergleichsmuster in Betracht.

Als Datenbasis dient die Olivetti-face-database mit 400 Aufnahmen, bestehend aus je 10 Varianten von 40 verschiedenen Gesichtern [SamHa] S. 3f .

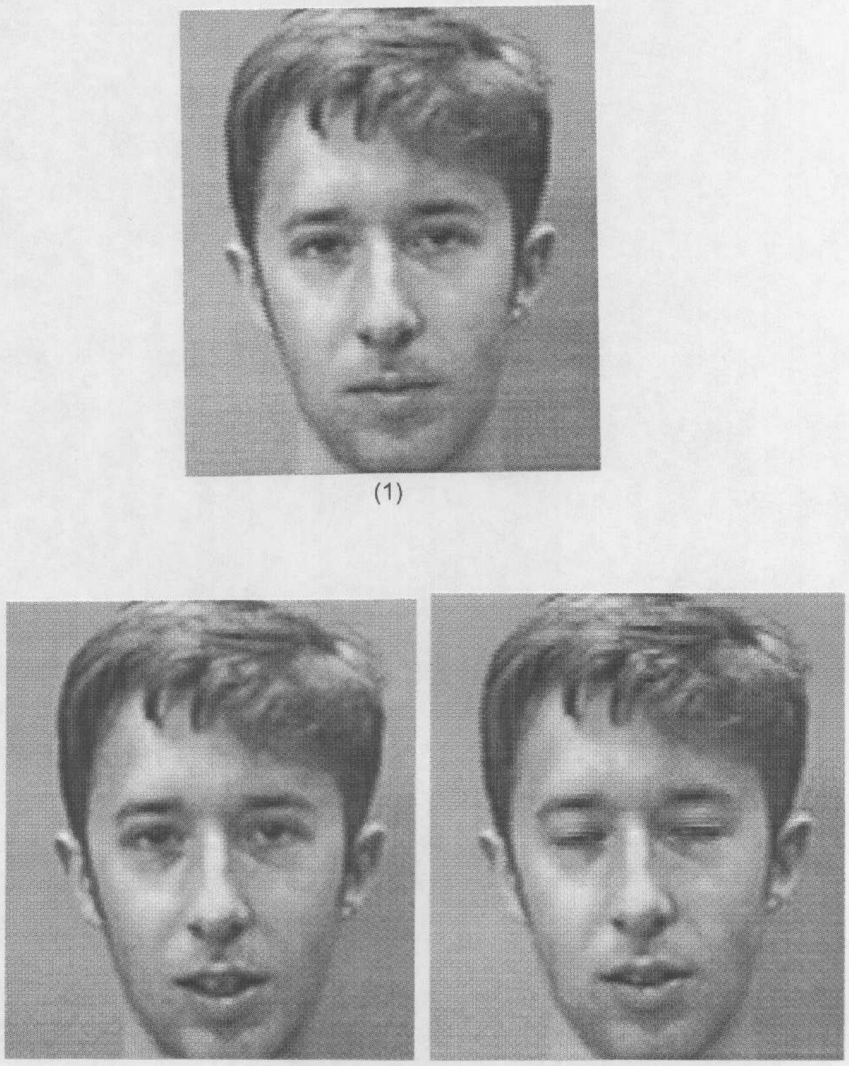


Abb.3.2.1. Beispiel aus der Bilddatenbank

3.2.2. Untersuchungs-Hypothesen

Der empirische Test für den in Frage stehenden Vergleich zwischen RBF und SVM wird in Anlehnung an eine Untersuchung von **SCHÖLKOPF** u. a. [Schö3] S. 5 so konstruiert, daß abgestuft vier Varianten der beiden Basisverfahren zum Einsatz kommen:

[A] klassisches RBF-Netz mit distanz-normierter Gaußfunktion: RBF_1

[B] GBF mit kovarianz-normierter Gaußfunktion: RBF_2

[C] Standard-SVM mit Gauß-*kernel* : $SVM(RBF_1)$

[D] Fusionsmaschine: $SVM_{CG}(RBF_1)$
[$SVM_{CG}(RBF_2)$]

Da zu erwarten ist, daß je nach Beschaffenheit des vorgelegten 2-Gruppen-Klassifikationsproblems, also der gestellten Lernaufgabe, die unter 4.1.-4.3. entwickelten Vergleichkriterien unterschiedlich einwirken oder sich auswirken, werden diese vier Modelle anhand mehrerer Fragestellungen verglichen:

(H.I) $SVM_{CG}(RBF_1)$ kann verschiedene 2D-Aufnahmen ein und desselben, bekannten Gesichts gegen Aufnahmen anderer Gesichter besser unterscheiden als $SVM(RBF_1)$, diese wiederum besser als RBF_2 und RBF_1.

(H.IIa) Bei der Unterscheidung von 2D-Bildern menschlicher männlicher gegenüber weiblichen Gesichtern hat keines der vier Netze $SVM_{CG}(RBF_1)$, $SVM(RBF_1)$, RBF_2 und RBF_1 signifikante Vorteile.

(H.IIb) Bei der Zuordnung von 2D-Bildern menschlicher Gesichter aus vordefinierten Altersgruppen hat keines der vier Netze $SVM_{CG}(RBF_1)$, $SVM(RBF_1)$, RBF_2 und RBF_1 signifikante Vorteile.

(H.III) $SVM_{CG}(RBF_1)$ kann 2D-Bilder von menschlichen Gesichtern von 2D-Bildern mit anderen Abbildungen besser unterscheiden als $SVM(RBF_1)$, $SVM(RBF_1)$ besitzt Vorteile gegenüber RBF_2, aber nicht RBF_2 gegenüber RBF_1.

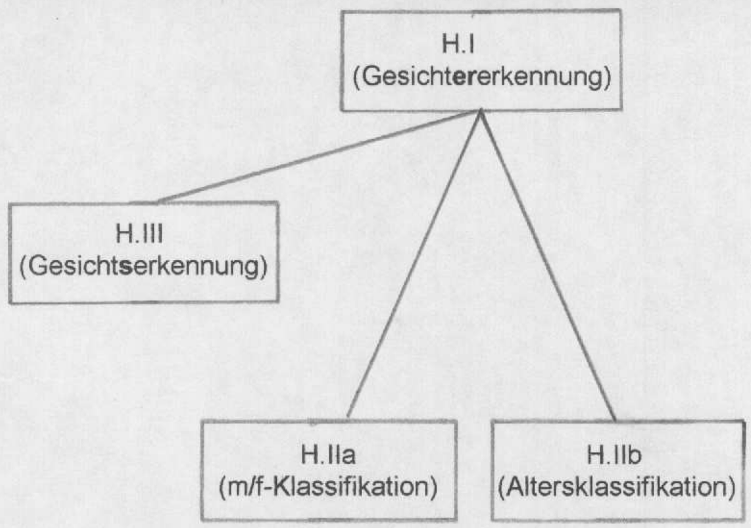


Abb.3.2.2. Bezüge zwischen den Hypothesen

Die Fragen (I) und (III) zielen auf verschiedene Einzelleistungen in der visuellen Mustererkennung ab (*recognition* und *detection*), bzgl. derer die verschiedenen Netztypen unterschiedlich leistungsfähig sein könnten.

Die Fragen (IIa,b) beinhalten Varianten von Fragestellung (I), wobei bestimmte Vorannahmen über unspezifische (mehrdeutige) Identifikationsmerkmale für Frauen und Männer eine Rolle spielen.

Frage (IIb) bietet im Vergleich zu Frage (IIa) eine kontinuierliche Merkmalsskala (gegenüber einer diskreten), der eine natürliche Unsicherheit der Information eigen ist (*fuzzy*-Komponente).

3.2.3. Die Simulations-Netze

Wie in 2.1.2. ausgeführt, soll neben der klassischen Form des RBF-Netzes (RBF_1) vergleichend eine Variante mit Modifikation der Radialsymmetrie (RBF_2) simuliert werden. Da diese Variante im SNNS nicht implementiert ist, waren entsprechende Ergänzungen erforderlich.

Innerhalb des Moduls **functions.h** werden alle Aktivationsfunktionen für den SNNS definiert, u.a. auch die hier benötigte **Act_RBF_Gaussian(sum2,bias)** durch (exp(-sum2 * bias**)**. Aus der Definition wird erkennbar, daß der Parameter bias in diesem Zusammenhang für den reziproken, quadrierten (in 2.1.2. eingeführten) Rausch-Parameter σ steht und sum2 für die Abstandsnorm $|\boldsymbol{X}\text{-}\boldsymbol{X}_i|^2$.

Die Bausteine für diese beiden Argumente, also σ, $\boldsymbol{X}$, $\boldsymbol{X}_i$, werden aus der Eingabe in den *header* templates.h übernommen. Dort wird $|\boldsymbol{X}\text{-}\boldsymbol{X}_i|^2$ zur Übergabe erzeugt.

An dieser Stelle wurde zur Realisierung der Variante RBF_2 die Prozedur zur Erzeugung von sum2 = $(\boldsymbol{X}\text{-}\boldsymbol{X}_i)^T$ * $CoVar^{-1}(\boldsymbol{X}, \boldsymbol{X}_i)$ * $(\boldsymbol{X}\text{-}\boldsymbol{X}_i)$ implementiert. Der StörParameter wird dabei gewissermaßen absorbiert (s. 2.1.2.) und muß auf σ =.5 festgesetzt werden (Abb. 3.2.3.a **ACT_RBF_GAUSSIAN(sum2,bias)**).

Zur Umsetzung des Fusionsnetzes $SVM_{CG}(RBF_2)$ aus 2.3.* waren die Programm-Module von SSVM entsprechend zu verändern.

Der RBF-Kern für SVMs wird innerhalb des Programmteils **KERNEL_RADIAL_BASIS_FUNCTION_C.{H,C}** aufbereitet. Die Routine **calcK()** gibt dazu in den Parametern gamma und result die Argumente für $K(\boldsymbol{X}, \boldsymbol{X}_i)$= exp (-gamma * result) zurück. Demnach kann gamma genau so behandelt werden wie bias in RBF_2, result wie sum2 (Abb. 3.2.3.b).

Zu beachten bleibt, daß $CoVar(\boldsymbol{X}, \boldsymbol{X}_i)$ nicht regulär sein muß, so daß $CoVar^{-1}(\boldsymbol{X}, \boldsymbol{X}_i)$ ggf. nicht unmittelbar existiert. Für diesen Fall kann man über die Pseudo-Inverse (mittels Aufmultiplizieren der Transponierten) versuchen, $CoVar^{-1}(\boldsymbol{X}, \boldsymbol{X}_i)$ zu konstruieren. Ansonsten bleibt die Alternative, in einer Fallunterscheidung immer dann, wenn $CoVar^{-1}$ nicht berechnet werden kann, auf die Normierung über die euklidische Distanz zurückzugreifen.

Die Berechnung der N_Z Zentroide aus den N_S Supportvektoren wird auf folgende Weise implementiert: Das Modul **chunk_c.{h,c}** verwaltet eine Liste der N_D Trainingsdaten sowie Funktionen zur Bearbeitung dieser Liste, darunter die Routine **calculate_support_vectors**, in der zu den jeweils durch Optimierung gewonnenen Werten alpha (für die Lagrangekoeffizienten α_i) der jeweilige Support-Vektor ξ_i unter **svs.insert_after(chunk_elem)** in die Liste eingestellt wird. Am Ende dieser Prozedur können folglich die N_S Support-Vektoren ξ_i für die Eingabe in den **cent_grav()**-Algorithmus (s. 2.3.3.) abgefragt werden. Die dort

```
if (RBF₁)
        define Act_RBF_Gaussian(sum2,bias) (exp(-sum2*bias));

if (RBF₂ )
          { sum2=( X - Xᵢ )ᵀ * CoVar(X,Xᵢ)⁻¹ * ( X -Xᵢ);
            bias=.5;
           }
return(exp(-sum2*bias));
```

Abb.3.2.3.a Struktur des Quellcodes SNNS(RBF_1/RBF_2)

```
if (RBF₁)
        {
         result=( X - Xᵢ )²;
         gamma=new_gamma;
         kernel_radial_basis_function=exp(-gamma*result);
        }

if (RBF₂)
        {
         result=( X - Xᵢ )ᵀ * CoVar(X,Xᵢ)⁻¹ * ( X -Xᵢ);
         gamma=.5;
         kernel_radial_basis_function=exp(-gamma*result);
        }
return (kernel_radial_basis_function);
```

Abb.3.2.3.b Struktur des Quellcodes SVM(RBF_1/RBF_2)

berechneten N_Z Zentroide μ_i werden sodann in die Routine **calcK()** reintegriert (zu den Details s. ANHANG (5)).

Darüber hinaus wurde die Implementierung eines PCA-Algorithmus zur Bearbeitung der Lerndaten benötigt. Zu diesem Zweck wurden die Module MatMult($\boldsymbol{X}$, $\boldsymbol{X}_i$), CoVT($\boldsymbol{X}$, $\boldsymbol{X}_i$), Eigen($\boldsymbol{X}$, $\boldsymbol{X}_i$) MatMult($\boldsymbol{X}$, $\boldsymbol{X}_i$) durch eine entsprechende Schnittstelle zusammengefügt (s. dazu ANHANG (7)).

Durchlauf-Struktur

main-Block	Aufruf	Wahl-Kern	Chunking	Ergebnis	File
(I)	trainsvm.asc	semi-lokal	-	KSI	akt_svs.asc
(II.1)	akt_svs.asc	dot	posh	KSI_CHUNK/	
				> **cent_grav**()	
				>MUE	akt_zen.asc
(II.2)	akt_zen.asc	RBF1/RBF2			

3.3. Versuchsaufbau für das Gegenstandsmodell

3.3.1. Benötigte Eingabe-Parameter

Der SNNS wird danach in folgenden Schritten zur Simulation von RBF_1 und RBF_2 verwendet:

(1a) Separate Definition eines *.pat-Files mit den formatierten und etikettierten N_D Trainingsdaten

(1b) ***Simulation SNNS.RBF***

Erstellung eines *.net-Files:	*enthält die Kenndaten der Netzstruktur*	
Festlegung der Aktivierungs-Funktion:	RBF_1 / RBF_2	[s. 2.1.2.]
Festlegung der Entscheidungs-Funktion:	Act_IDENITYPLUSBIAS	[s. 2.1.2.]
Wahl der Gewichtsinitialisierung:	RBF_WEIGHTS	[s. 2.1.3.]
Wahl des Update-Verfahrens:	SYNCHRONOUS_ORDER	[s. 2.1.3.]
Entscheidung für Lernalgorithmus:	RBF_DYNAMIC_DECAY	[s. 2.1.3.]

Der SSVM-Simulator wird danach zur Realisierung von SVM(RBF_1) bzw. von SVM_{CG} (RBF_2) in folgenden Schritten verwendet:

(2a) separate Erzeugung eines *.dat-Files mit den formatierten und etikettierten N_D Lerndaten

(2b) ***Simulation SSVM***

Maschinentyp: Mustererkennung
gewählter Kern: RBF_1 / RBF_2
Kern-Parameter: $\gamma = \{.1,.5, 1, 2,5\}$
allgemeine Parameter: *default*
Strukturierung der Lerndaten
chunking (Portionierung): *im Standardfall nicht erforderlich [s. 5.1. Standard-Datensatz]*
scaling: *wg. Vergleichbarkeit nicht angewandt*
Sonder-Parameter: - - -

Anzahl Lerndaten: $n = N_D$
Anzahl Merkmale: m = 10 304
gewählter Optimierer: LOQO
BOTTOU
α-Schranke: C=0 (*d.h. unbeschränkt*)
zero-Schranke: 1.0 e-16
zero-Toleranz: 1.0 e-07 } *Angaben zu numerischen Toleranzen*

Vorversuche mit polynomialen Kernen anstelle von RBF-Kernen lassen vermuten, daß RBF-Kerne unter Umständen nicht die optimale Wahl für die 2-Gruppen-Klassifikation bei der Mustererkennung darstellen (s. dazu [Schö7] S.n 1 und 7). Das wäre aber für den angestrebten Vergleich nicht unbedingt hinderlich, da etwa bei der Testung von technischen Geräten das Verhalten unter extremen und widrigen Bedingungen mehr Aufschluß über deren Beschaffenheit liefert als Standardbedingungen.

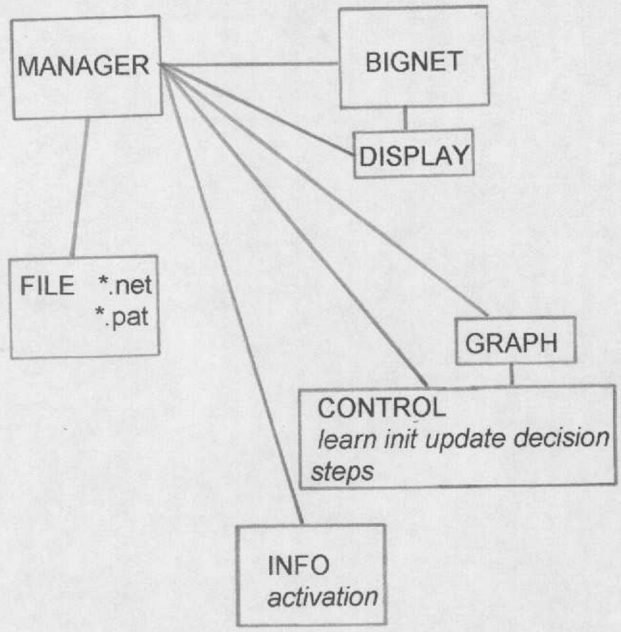


Abb.3.3.1.a Komponenten des Eingabebildschirms SNNS

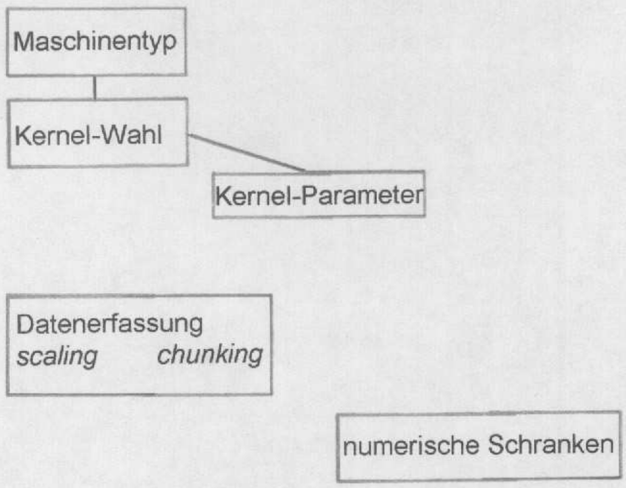


Abb.3.3.1.b Komponenten des Eingabebildschirms SSVM

3.3.2. Interpretierbare Parameter-Ausgaben

Als Grundlage für Komplexitätsberechnungen (s. 3.3.3.) wurden in die Lern-Algorithmen DDA von SNNS sowie LOQO und BOTTOU von SSVM diverse Iterationszähler implementiert. Dabei ist zu bedenken, daß die Iterationen von DDA theoretisch begründet sind, während die Iterationen in LOQO und BOTTOU sich lediglich aus den Bedingungen maschineller Datenverarbeitung ergeben.

(1) ***Simulation SNNS.RBF***

Mit den Bezeichnungen:
p = Anzahl zu schätzender Parameter (Gewichte) ,
SSE = Summe der quadrierten Fehler,
TSS = das Total der Quadrate, korrigiert gegen den Mittelwert der abhängigen Variablen,
werden u.a. folgende Maße als Gütekriterien bestimmt:
Pearson-Koeffizient (Proportion der Varianz)
$R^2 := 1 - (SSE/TSS)$
Angemessenheit des Modells in der Population
$PC := 1 - [(n+p)/(n-p)] * (1 - R^2)$
Schwarzsches BAYES-Kriterium
$SBC := n * \ln (SSE/n) + n * \ln p$ (s. dazu 4.1.2. und 4.2.1.)
Für kleine n, insbesondere wenn nicht n>>p erreicht werden kann, müssen diese Maße allerdings als wenig aussagekräftig bzw. unsicher gelten.

Daten aus dem *.log-File:

außer obigen Maßen zu Fehlern und Güte des Trainings werden die Trainings-Parameter, die Anzahl der Iterationszyklen und die Inhalte des INFO-panels zur Aktivation (s.a. Abb. 3.3.2.a) protokolliert.

(2) ***Simulation SSVM***

Die Standard-Parameter-Ausgabe von SSVM ist in folgender Weise ergänzt:

+ -Support-Vektoren (A):
- -Support-Vektoren (A^c):
begrenzte Support-Vektoren:
max α:
min α:
margin from hyperplane b0.risk.emp:

Wert der Zielfunktion f.objective:
Anzahl Iterationen IT:
STABILISIERUNGS-Set:
Fehler-Rate F%:

Anzahl + -Muster (A):
davon falsch zugeordnet (α-Fehler):
lfd. Nrn.:
Anzahl - -Muster (A^c):
davon falsch zugeordnet (ß-Fehler):
lfd. Nrn.:

OPTIMIERUNGS-Set:

Anzahl + -Muster (A):
davon falsch zugeordnet (α-Fehler):
lfd. Nrn.:
Anzahl - -Muster (A^c):
davon falsch zugeordnet (ß-Fehler):
lfd. Nrn.:

Liste: Nr.(SV) α(Nr.)

|w| =
w[0]= ... w[m-1]= [*nur für kleine m sinnvoll*]

max |x| =
VC_{dim} <=
Selektions-Koeffizient N_S / N_D =

Bei der Interpretation der Netz-Ausgaben bleibt zu bedenken, inwieweit Ergebnisse von Besonderheiten der verwendeten Simulatoren beeinflußt sein könnten und inwieweit sie allgemeine Gültigkeit beanspruchen können. Als Beispiel sei genannt, daß eine A/B-Klassifikation ebenso mit einer wie mit zwei output-units gelöst werden könnte, je nachdem ob man B als A^c auffassen will oder auch ($\neg$A,$\neg$B)-Fälle offenhalten möchte.

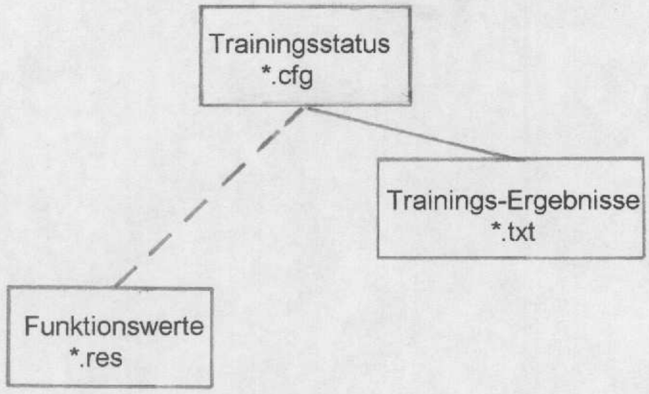


Abb.3.3.2.a Schema der Ausgabedaten SNNS

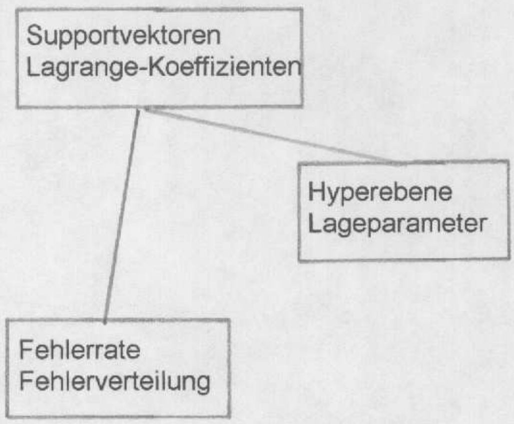


Abb.3.3.2.b Schema der Ausgabedaten SSVM

3.3.3. Gütekriterien

Unter den drei bereits genannten Gütekriterienklassen für die Performanz eines Netzes, *model fit* (Risiko: Trennsicherheit, Zuverlässigkeit), *model complexity* (Kosten: Rechenzeitdauer, Speicherplatzbedarf) und *model propriety* (Nutzen: Zweckmäßigkeit, Wiederholbarkeit), ergibt sich insofern eine natürliche Hierarchie, als ohne den Anspruch an befriedigend begrenztes Risiko (*model fit*) weitere Güteabwägungen sinnlos würden.

Ob dieser Anspruch immer in die Forderung nach einer gänzlich fehlerfreien Modellanpassung münden muß, bleibt im Einzelfall einer Risiko-/Kosten-Abwägung überlassen. Dennoch gibt es zwei grundsätzliche Überlegungen, denen jeder Einzelfall in gleicher Weise unterliegt: Einerseits spricht für eine gewisse Fehlertoleranz, daß überangepaßte Modelle u.U. kaum generalisierbar sind, womit etwa auch die willkürlichen Randbegradigungen in Clusterwolken bei **ZHANG** zu rechtfertigen wären; andererseits steht für eine anfänglich strenge 0-Fehler-Toleranz, daß Autoren wie **ZHANG** oder **TIPPING** ein grundsätzliches Risiko eingehen, wenn sie augenscheinlich marginale, singuläre Werte aus Clustern nach Opportunität übergehen oder aussortieren, da diese eben doch Standard-Elemente einer gleichverteilten, aber unvollständig erhobenen Grundgesamtheit sein könnten, so daß sie dann ebenso viel Aussagekraft und Bedeutung besäßen wie die berücksichtigten Clusterelemente auch.

Eindeutiger wird die Abwägung für Anwendungsfälle in sicherheitsrelevanten, aber nicht zeitkritischen Bereichen (vgl. Verkehrslenkung vs. Medizin) ausfallen, bei denen zugunsten besonders hoher Anforderungen an die Trennsicherheit (*model fit*) die modellinterne Redundanz zumeist willentlich derart erweitert wird, daß erhebliche Abstriche in der algorithmischen Effizienz (*model complexity*) dafür in Kauf genommen werden (vgl. 4.3.1.). Die Einführung virtueller Support-Vektoren kann in dieser Weise motiviert sein.

Man kann anhand der Gegenüberstellungen aus 2.1. und 2.2 erwarten, daß RBFs aufgrund ihrer Dynamik leichter aus einem zunächst groben Anpassungsstatus durch nachträgliche Restriktionen verfeinert werden können, während SVMs aufgrund der Möglichkeit, den α-bound-Parameter C zu verändern, leichter im Bedarfsfalle aus einer aus einem zunächst strikten Anpassungsstatus erweitert werden könnten.

Vergleiche der *model complexity* verschiedener Konzepte für das Anpassen der RBF- wie der SVM-Netze zeigen, daß oft Teilaufgaben ausgelagert werden, so daß zunächst bei der eigentlichen SVM-Optimierung ein erheblicher Gewinn an Rechenzeit ins Auge fällt, der aber unter Anrechnung des zusätzlichen Zeitbedarfs für die vorgeschalteten oder nachgelagerten Operationen weitgehend absorbiert wird. Der

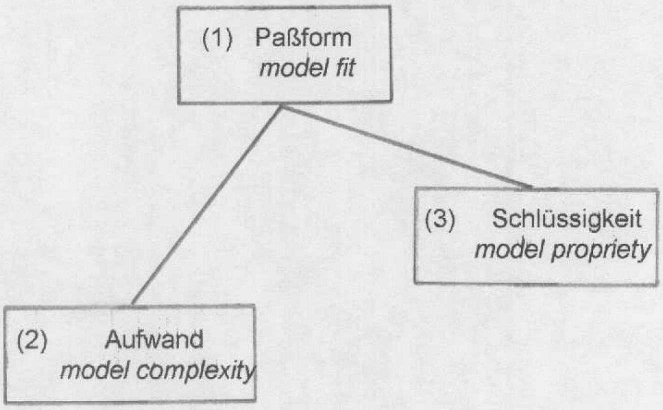


Abb.3.3.3. Klassen von Gütekriterien

entscheidende Gewinn solcher modularer Lösungen dürfte in ihrer Wieder- und Weiterverwertbarkeit liegen. Während aber der vermeintliche Rechenzeitgewinn in den Forschungsberichten eher überbewertet bzw. beschönigt wird, wird letzteres Argument in den vorliegenden Untersuchungen überraschenderweise nicht ins Feld geführt.

Der Einwand gilt entsprechend für eine Bewertung der Vorschaltverfahren zur Strukturierung der Daten, wie zum Beispiel die PCA, die eher die Trennsicherheit erhöhen als zu einem bedeutenden Rechenzeitgewinn führen.

Der Zweckmäßigkeit (*model propriety*) kommt insofern eine untergeordnete Bedeutung zu, als für diesen Bereich nicht bei jeder Fragestellung mittelbar weitere Aspekte ins Spiel kommen (s. hierzu 4.3.2.), die über die unmittelbare Trennsicherheit (*model fit*) hinausgehen (Abb. 3.3.3.).

Im einzelnen stellen sich die Kriterienklassen wie folgt dar:

(1) Die Forderung nach *model fit* bezieht sich auf die Anpassung des Netzes an die Grundgesamtheit der Daten und realisiert sich über den Vergleich zwischen dem wahrscheinlichkeitstheoretischen Erwartungswert und den empirischen Beobachtungswerten während der Anpassungsphase des Netzes. Der Erwartungswert markiert die Residuale (error-Anteile) der Anpassung $E[(f(x,D)-y)^2 | D]$ mit

$$ED[(f(x,D)-E[y|x])^2] = ED[(f(x,D)-ED[f(x,D)])^2] + ED[(f(x,D)-E[y|x])^2] \qquad \text{(F.3.3-1)}$$

Dem Zusammenspiel der beiden additiven Komponenten kommt eine große praktische Bedeutung zu (s. 4.2.3.).

(2) Die Forderung *model complexity* speist sich aus der kombinatorischen Analyse der algorithmischen Einzelschritte zur Gewinnung eines angepaßten Netzes und muß folglich außer der Komplexität der Entscheidungsfunktion auch die Komplexität der Subroutinen für Gewichtsberechnung und Adaption berücksichtigen. Im pragmatischen Vorgehen können Peripheriealgorithmen die Komplexität entscheidend erhöhen (vgl. OLVQ-Algorittmus zur Optimierung der Stützstellenwahl in RBFs [**VOGTM**] S. 1841).

Für die Optimierungsmodule ist demnach sicherlich ein Iterationszähler zur Bestimmung der Komplexitätsklasse ein geeignetes Maß. Um eine komplette Simulation zu beurteilen, wäre es hingegen wirklichkeitsnäher, die verbrauchte Gesamt-CPU-Zeit aller Algorithmen zu messen, wobei diejenigen Module auszuklammern wären, die der Verwaltung der Benutzeroberfläche dienen.

In diesem Zusammenhang liefert die VC-Dimension ein notwendiges Kriterium für Durchführbarkeit modellfreien Lernens in Netzen. Eine endliche VC-Dimension liefert allerdings noch keine direkten Anhaltspunkte für die Effizienz eines Algorithmus [CorVa] S. 298. Insbesondere gibt es Netze mit endlicher VC-Dimension (so daß modellfreies Lernen möglich ist), für die konkrete Fragestellungen aber als NP-vollständig nachgewiesen werden können (s. Beispiele in [Vidya]).

Ebenso kann die MDL (bzgl. Entropie und Redundanz) eines Klassifikators eher eine Orientierungsmarke als ein pragmatisches Kriterium für einen anwendungs-optimalen algorithmischen Ansatz sein (s. dazu im einzelnen 4.2.1.).

SCHÖLKOPF räumt ein, daß es für SVM kein origniäres, unmittelbares Verfahren gibt, um das Konvergenzverhalten einzuschätzen (Fehlerschranken herzuleiten), sondern daß es sekundärer Annahmen bedarf (vgl. besipielhaft die verschiedenen Versuchsbeschreibungen in [Schö1]).

So gibt es zwar algebraische Abschätzungen, die Parameter aus (1) und (2) zusammenführen [Schö8] [Schö0] S. 1; Sie erscheinen jedoch als wenig zwingend:

Die Analyse des durchschnittlichen Fehlers aus dem Erwartungswert der Wahrscheinlichkeitsverteilung P(x,y) ergibt zunächst

$$\text{Rtheo}(f) = 1/2 \int |f(x)-y| \, \boldsymbol{d} P(x,y) .$$

Dieser Wert ist jedoch mittels der obigen Gleichung nicht berechenbar, da P(x,y) i.a. nicht bekannt ist. Er kann aber über tatsächlich meßbare Beobachtungswerte für empirische Teilmengen der Grundgesamtheit abgeschätzt werden.

$$\text{Remp}(f) = 1/k \, \Sigma_i \, 1/2 \, |f(x_i)-y_i| \quad i=1...k \,..$$

Mit einer gewählten Wahrscheinlichkeit $1-\eta$ gilt die Schranke:

$$\text{Rtheo}(f) <= \text{Remp}(f) + \sqrt{}\,(h\,(\ln\,(2k/h) + 1) - \ln\,(\eta/4)/k)$$

für die Entscheidungsfunktion f mit der VC-Dimension h bei k empirisch berücksichtigten Mustern.

Man erkennt, daß die Breite des Vertrauensintervalls (wünschenswert) kleiner wird, wenn man für ein festes h (VC-Dimension) $1-\eta$ verkleinert (hier setzt PCA an) und k vergrößert. [Schö0] S. 1f, [CorVa] S. 287, [Burge] S. 2f

(3) Die Forderung nach *model propriety* ist letztlich dem Paradigma der schließenden Statistik entlehnt, indem kontextbezogene Restriktionen (z.B. Marken für nicht zur Disposition stehende Nachbarschaftsbeziehungen o.ä.) formuliert werden, die für die Aufrechterhaltung einer Hypothese berücksichtigt werden müssen. Empirische Ergebnisse führen auf dieser Grundlage zu einer Signifkanzaussage unter einer bestimmten Irrtumswahrscheinlichkeit.

Empirisch sollten somit zwei Grenzen exploriert werden:

(1) Ab welchem N_D kann das Netz in welcher Komplexitätsklasse fehlerfrei trainiert werden?
(2) Ab welchem N_D werden nicht mehr sämtliche Lerndaten ∂_i als Supportvektoren ξ_i benötigt?

Um die jeweiligen Ergebnisse zu vergleichen, wurden in den Programm-Modulen von SNNS und SVM folgende Gütekriterien formuliert und abgefragt:

Fehler-Rate	Anzahl Iterationen	Selektions-Koeffizient
F%	IT	N_s / N_d

Also die Fehler-Rate als das zentrale Kriterium für die Modell-Paßform, die Anzahl Iterationen (im Trainingsbereich) als übliches Maß für den Aufwand (unter dem Vorbehalt von 2.2.3., daß die Iterationen in SNNS und in SSVM unterschiedliche theoretische Qualität besitzen) sowie den Selektions-Koeffizienten als grundlegenden Indikator für die Modell-Schlüssigkeit (d.h. inwieweit die benötigten Support-Vektoren - je weniger, desto besser generalisierbar - die beiden Klassifikationsmengen widerspiegeln).

Aus diesen basalen Kriterien könnten nach Bedarf komplexere, bedingte Kennwerte abgeleitet werden.

4. Analyse der Vergleichskriterien

4.1. Aspekte zur Beurteilung der Lern-Daten

4.1.1. Generalisierbarkeit

Für realistische Anwendungen ist es wichtig, abschätzen zu können, wie sich ein Netz gegenüber solchen Daten verhalten wird, die in der Grundgesamtheit der Lerndaten nicht vorkamen.

Ein gut trainiertes Netz ist bestmöglich angepaßt, das heißt, es führt bei Test- bzw. Anwendungsdaten, die den Trainingsdaten ähneln, zu bestmöglichen Ergebnissen, also zu schneller Konvergenz und geringstmöglichen Fehlern. Dies besagt aber noch nichts über seine Generalisierbarkeit, d.h. über die Zuverlässigkeit seines Verhalten gegenüber Daten, die zwar in das gleiche Klassifikationsschema gehören wie die Trainingsdaten, diesen aber z.B. in der Gewichtung der entscheidungsrelevanten Merkmale nur bedingt oder auf einer aggregierten Ebene ähnlich sind.

Gängige Maße zur Beurteilung der Möglichkeiten haben allerdings oft eher theoretisches Gewicht als praktische Bedeutung. So hängt die Breite des Vertrauensintervalls einer Klassifikation von der VC-Dimension des Netzes ab, die die Leistungsfähigkeit einer Menge von Funktionen bzgl. ihrer Separierungseigenschaften beschreibt [Schö0] ebda. . Mittels dieser Kategorie läßt sich die prinzipielle Kapazität eines Netzes einschätzen, etwa, ob für ein Netz modellfreies Lernen möglich ist, sie sagt aber über die aktuell zu erwartende Komplexität nicht unbedingt Greifbares aus.

Besondere Auswirkungen auf die Generalisierbarkeit eines Netzes zeitigt das sog. *bias/variance*-Dilemma. Folgt man dabei der späteren Argumentation von 4.2.3., so ist nicht zu erwarten, daß SVM-Netze gleichwertige Generalisierbarkeitseigenschaften besitzen wie RBF-Netze.

Da RBF-Netze dynamisch anhand von Stützstellen aufgebaut werden, deren zugrundeliegende Musterstruktur eine durch σ näher bestimmte radiale Ausstrahlung erzeugt, erreichen RBF-Netze prinzipiell gute Generalisierungseigenschaften [Zell1] S. 240; [Bisho] S. 184 f. Das jeweilige Netz braucht allerdings "Erfahrungen'" mit verrauschten Daten, um diese "Fähigkeit" zu erlangen. Diesen Effekt nutzt in vergleichbarer Weise der *nearest-miss*-Ansatz bei SVMs.

Im Falle der Gesichtererkennung ist die Klasse A typischerweise durch ein Stamm-Muster (ein Stamm-Zentroid) besetzt, alle weiteren Elemente bestehen aus bewußt konstruierten geringfügigen Abweichungen davon im Gesichtsausdruck und/oder dem Blickwinkel (also gerade dem Rauschen im Sinne des Parameters σ). Die Klasse A^c hingegen wird in natürlicher Weise aus mehreren Stamm-Mustern bestehen, deren Abweichungen untereinander gerade über eine mittels σ beschreibbare Ausstrahlung hinausgehen.

Abb. 4.1.1. zeigt ein Bild-*set*, bei dem die Varianten (1)-(3) in naheliegender Weise als Stamm-Muster in Betracht kämen. Wäre hingegen Variante (4) mit den geschlossenen Augen das zuerst

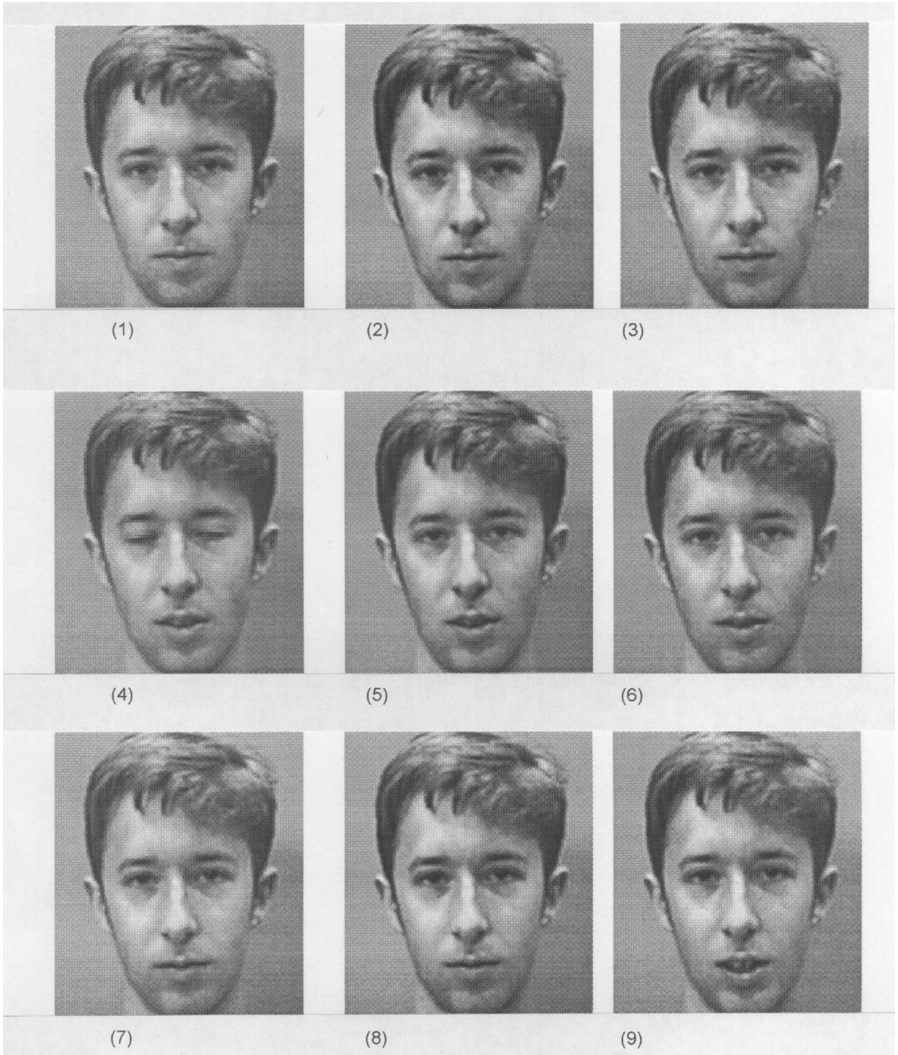


Abb.4.1.1. Bereichsweise Verteilung

präsentierte Stamm-Muster, so könnte man vermuten, daß im Verlauf des Lernens für diese Person zwei Stützstellen generiert werden, insbesondere beim sukzessiven Lernen von RBFs. Ähnliches gilt, wenn bespielsweise Variante (1) mittels der Varianten (5),(6),(9) im Trainingsdatensatz ausschließlich durch Varianten mit geöffnetem Mund ergänzt würden.

Im Rahmen der vorliegenden Simulatoren (SNNS, SSVM) können zwar nur bedingt Anwendungen mit unmarkierten Daten (d.h. ohne Eingabe von y_i) realisiert werden, man könnte aber die Generalisierungseigenschaft dadurch überprüfen, daß man auf ein trainiertes und validiertes bzw. optimiertes und getestetes Netz Daten aus anderen Bilddatenbanken anwendet, die probeweise entsprechend der Validierung bzw. Optimierung markiert sind; d.h. es erfolgt eine Fehlerkontrolle anstelle einer ergebnisoffenen Anwendung.

Ein naheliegender Weg, die Generalsierungsfähigkeit zu verbessern, besteht darin, Kriterien für eine begrenzte Fehler-Toleranz zu entwickeln (vgl. [Campb] und [StrMa]). Diese Ansätze sollten aber nicht mit der Definition der sog. *slack*-Variablen in Zusammenhang gebracht werden, mittels derer man bei SVMs anstrebt, begrenzte Abweichungen zu kontrollieren, die zu einem Verlust der linearen Separierbarkeit führen würden (s. z.B. [Burge] S. 15).

4.1.2. Bereichsweise Verteilung

Wie in Abschnitt 2.2.2 gezeigt wurde, bilden RBF-Netze kein besonderes ursprüngliches Beispiel für neuronale Netze wie das Perzeptron. Ob es sich bei SVMs überhaupt um neuronale Netze handelt, bleibt in der vorliegenden Forschungsliteratur gar umstritten. Deshalb ist die Ebene des Simulationsmodells (RBF vs. SVM) nicht unbedingt geieignet, im einzelnen Vergleichskriterien zu entwickeln. Sie sollen deshalb nachfolgend allgemeiner direkt aus den Verteilungs-Eigenschaften der Grundmenge der zu modellierenden Daten gebildet werden.

Will man Untermengen dieser Datenmenge klassifizieren, so möchte man etwas über die apriori-, die bedingten und die aposteriori-Wahrscheinlichkeiten in der Daten-Grundgesamtheit wissen (vgl. 3.1.3.).

Unter der Laplaceschen Annahme einer Gleichwahrscheinlichkeit aller Ereignisse (Muster) erhält man beim Übergang von diskreten zu kontinuierlichen Skalen gerade die Dichtefunktion der Gaußschen Normalverteilung

φ_i: IR -> IR mit

$\varphi_i(x) = (1/\sigma\sqrt{(2\pi)}) \exp(-(x-\mu)^2 / 2\sigma^2)$, (F.4.1-1)

d.h. die Verwendung von GBF-Netzen zur Klassifikation von Datenmengen liegt dann nahe, wenn die Grundgesamtheit der Daten im Eingaberaum zumindest bereichsweise gleichverteilt bzw. die Zufallsgrößen normalverteilt sind, d.h. zumindest im Binnenbereich einer Klasse $\boldsymbol{X}_i$, so daß im Bereich der Ränder erwartbare Abweichungen tolerierbar sind.

Radiale BasisFunktionen wurden unter der Simulationsperspektive in 1.1. als Verfahren zur Approximation multidimensionaler Verteilungen vorgestellt. Unter der neuen, allgemeineren Perspektive kann man feststellen: RBF-Netze modellieren Entscheidungen für normalverteilte (stetige) Zufallsgrößen.

BISHOP erläutert dazu, daß RBF-Netze für Klassifikationsaufgaben unter der Annahme einer Gleichverteilung der Daten als Verfahren zur Bestimmung der aposteriori-Wahrscheinlichkeiten im Sinne des BAYES-Theorems [s. ANHANG (6)] aufgefaßt werden können [Bisho] S. 179-181:

Setzt man $\phi_i(x) = p(x \mid \boldsymbol{X}_i) \; / \; \sum_i p(x \mid \boldsymbol{X}_i) \, P(\boldsymbol{X}_i)$, (F.4.1-2)

also der bedingten Wahrscheinlichkeit dafür, daß ein Muster x zu $\boldsymbol{X}_i$ gehört, ins Verhältnis zur totalen Wahrscheinlichkeit für x, so ergibt sich daraus, daß $\phi_i(x) * P(\boldsymbol{X}_i)$ gerade $P(\boldsymbol{X}_i \mid x)$ im Sinne des Bayes-Theorems ergibt. $P(\boldsymbol{X}_i)$ wäre in dieser Interpretation das Gewicht, mit dem ϕ_i vom verdeckten Knoten zur Ausgabeeinheit belegt wird.

Diese Deutung ist insoweit idealisiert, als i.a. ein ϕ_i nicht genügen wird, um die Zugehörigkeit zu einer Klasse $\boldsymbol{X}_i$ angemessen zu beschreiben. Folglich betrachte man laut **BISHOP** k Basisfunktionen $\phi\ j_i$, indem man $p(x \mid \boldsymbol{X}_i)$ in nachfolgender Weise aufspaltet:

$$p(x \mid \boldsymbol{X}_i)= \Sigma_i\ p(x \mid j)\ P(j \mid \boldsymbol{X}_i).$$

Daraus ergibt sich für die unbedingte Dichte

$$p(x) = \Sigma_i\ p(x \mid \boldsymbol{X}_i)\ P(\boldsymbol{X}_i) = \Sigma_i\ p(x \mid j)\ P(j)\ , \qquad \text{(F.4.1.-3)}$$

mit der apriori-Wahrscheinlichkeit

$$P(j)= \Sigma_i\ P(j \mid \boldsymbol{X}_i)\ P(\boldsymbol{X}_i).$$

Somit wird $\phi\ j_i\,(x) = p(x \mid j)\ P(j)\ /\ \Sigma_k\ p(x \mid k)\ P(k)$

mit den Gewichten $wij = P(j \mid \boldsymbol{X}_i)\ P(\boldsymbol{X}_i)\ /\ P(j)$,

so daß nunmehr

$$\Sigma_i\ w_{ij}\ \phi\ j_i\,(x) = P(\boldsymbol{X}_i|j) \qquad \text{(F.4.1-4)}$$

die aposteriori-Wahrscheinlichkeit für $\boldsymbol{X}_i$ beschreibt, so daß die Zuordnung von x zur wahrscheinlichsten Klasse $\boldsymbol{X}_i$ mittels ϕ_{ji} dargestellt wird.

Damit können die Aktivationen der ϕ_{ji} interpretiert werden als aposteriori-Wahrscheinlichkeiten für das Vorhandensein von Mustern im Eingaberaum, und die Gewichte können interpretiert werden als aposteriori-Wahrscheinlichkeiten für die Klassenzugehörigkeit zu gegebenen Mustern.

RBFs für 2-Gruppen-Klassifikationen sind also ein Modell für den Spezialfall des BAYES-Theorems mit einer Zerlegung des Ergebnisraumes in 2 Ereignisse und liefert als solches von den Lerndaten her eine neue Perspektive auf die Simulation.

Greift man angesichts des Begriffs der “Dichtefunktion” in diesem Zusammenhang das physikalische Analogon aus 2.3. wieder auf, so könnte man sich die apriori-Wahrscheinlichkeiten als Informationen über den Rauminhalt einer Clusterwolke vorstellen, die aposteriori-Wahrscheinlichkeiten über die Dichtefunktion als Aussagen über bestimmte Massenanteile ebendort (s. Abb. 4.1.2.) .

Könnte man nun auch SVMs in diesen Zusammenhang stellen und als Modelle für Wahrcheinlichkeitsverteilungen deuten, so entstünde nicht nur eine einheitliche Vergleichsgrundlage mit RBFs; es würde insbesondere möglich, die Wahl von Gaußkernen für SVM(GBF) jenseits von Intuition und Vorerfahrungen (vgl. 2.2.2. und 2.3.1.) zu motivieren und zu begründen.

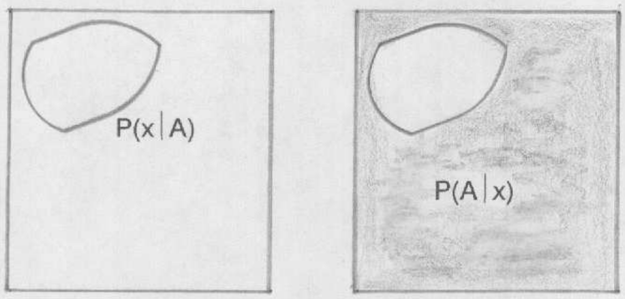
In seinem Artikel “Probabilistic Interpretation and Bayesian Methods for Support Vector Machines” geht **SOLLICH** dieser Frage nach. Ausgehend von der üblichen Darstellung der trennenden Hyperebene im Merkmalsraum mittels $\boldsymbol{w} * \varnothing(x) + b = 0$ zeigt **SOLLICH,** daß die

$$P(A \mid x) = P(A) * P(x \mid A) \,/\, (P(A)\, P(x \mid A) + P(A^c)\, P(x \mid A^c))$$

externe Perspektive: $P(A)$

$P(A^c)$

interne Perspektive:



apriori | aposteriori

Aspekt *Rauminhalt* | Aspekt *Masse*

Abb.4.1.2. BAYES-Theorem für 2 Ereignisse

Funktionswerte $a(\boldsymbol{X}) := \boldsymbol{w} * \varnothing(\boldsymbol{X})$ verbundene Gaußsche Verteilungen besitzen, da eben dies für die Komponenten vom Normalenvektor $\boldsymbol{w}$ gilt.

SOLLICH führt weiter aus, daß die apriori-Wahrscheinlichkeiten in Verteilungen, für die SVMs ein Modell bilden, gerade solchen stochastischen Prozessen über die so definierten Funktionen a entsprechen [Solli] S.2, bei denen alle endlich dimensionalen Verteilungen laplacesch, also gleichverteilt, ausfallen. Und das sind per Definition die Gaußschen Prozesse.

Daraus folgert er, daß
... the SVMs correspond to maximum a posteriori solutions of inference problems with zero mean Gaussian process (GP) priors [Solli] S. 3,

wobei das Optimierungsproblem der SVM
$\min_{w,b} \quad ½ \, | \boldsymbol{w}|^2 + C \, \Sigma_x \; l \, (y_x \, [\boldsymbol{w} * \varnothing(x) + b]$
gerade die Bestimmung der aposteriori-Wahrscheinlichkeit für die Parameter $\boldsymbol{w}$ und b zu einem gegebenen Lerndatensatz D beschreibt.

Eine SVM für eine 2-Gruppen-Klassifikation zu trainieren, kann damit gemäß **SOLLICH** beschrieben werden als die Aufgabe, die maximale aposteriori-Wahrscheinlichkeit von a (und dem offset b) zu bestimmen, d.h. diejenigen Werte zu finden, für die $P(a,b|D) \cong P(a)\ P(D|a,b)$ maximal wird [Solli] S. 2;4, womit wiederum der genannte Spezialfall des BAYES-Theorems abgebildet wird.

Schließlich rechnet er vor, daß die Kovarianzen der $CoVar(a\ (\boldsymbol{X}), a(\boldsymbol{X}_i))$ (mittelwertzentriert) genau durch

$K(\boldsymbol{X},\boldsymbol{X}_i) = \varnothing(\boldsymbol{X}) * \varnothing(\boldsymbol{X}_i)$ repräsentiert werden.

Als Kovarianzfunktionen Gaußscher Prozesse beschreiben die Kerne nunmehr interpretierbare Vorannahmen darüber, welche Art von Funktionen ∅ wahrscheinlich auftreten würden, wenn man sie denn bestimmen müßte: z.B. bzgl. ihrer Glätte, Diferenzierbarkeit und ähnlicher Eigenschaften, wie sie sich in der Entscheidungsoberfläche ablesen lassen.

SOLLICH hebt ebenfalls den wichtigen Effekt der Kern-Ersetzung hervor, daß "*the feature space has disappeared entirely from this formulation*" [Solli] S. 2, wie in der vorliegenden Arbeit bereits unter 2.3. dargestellt und genutzt.

Schließlich ermöglicht die Anwendung BAYESscher Methoden die plausibel herleitbare Abschätzung der Störeinfüsse (des Kern-Parameters) und von Fehlerschranken, was die in 3.2.3. kritisierte Lücke schließen kann (s. a. 4.3.1. und ANHANG (6)).

Es zeigt sich also, daß bei der scheinbar voraussetzungslosen Wahl eines Gauß-Kernes für SVM aufgrund "allgemeiner Erfahrungswerte" sehr wohl Annahmen oder implizite Kenntnisse über die Verteilung der Lerndaten einfließen, die aber üblicherweise in der bisherigen Forschungsliteratur nicht weiter analysiert werden. Die Interpretation im Rahmen des BAYES-Theorems verdeutlicht hingegen, daß die Wahl eines Gauß-Kernes eben nicht voraussetzungslos ist, sondern - nicht unerwartet - eben den Voraussetzungen folgt, die auch für eine begründete Anwendung von SVM(RBF)-Netzen gesichert werden müssen.

4.1.3. Erfassung der Grundgesamtheit

Es wurde bereits als Motiv für approximatives Lernen mit RBFs genannt, die Anzahl der benötigten Stützstellen und Basisfunktionen (entsprechend die Anzahl von Support-Vektoren zur Festlegung der Hyperebene) klein zu halten (2.1.2.). Es wird auf diese Weise gleichzeitig möglich, die zu verarbeitende Datenmenge im Vergleich zur exakten Interpolationslösung drastisch zu verkleinern (je nach Domäne etwa auf 10%). Zwei Ansätze bieten sich an, diese Verkleinerung der Datenmenge in der Praxis zu erreichen: ***Selektion*** von Daten und ***Reduktion*** von Daten aus der Grundgesamtheit.

Sei $D_{(m1,m2)\ n}$ eine Datei (*array*) von n möglichen Lerndaten (Mustern), wobei in der vorliegenden Untersuchung jedes einzelne Muster als eine (m_1,m_2)-Matrix oder strenggenommen als m_1*m_2-Vektor von Grauwerten von m_1*m_2 Pixeln gegeben ist. Je weniger dieser Daten aus der verfügbaren Grundgesamtheit der Dimension n als Muster zum Netz-Training bzw. zur Netz-Optimierung Verwendung finden, desto entscheidender wird es natürlich, treffende Kriterien für die Vorverarbeitung (***Selektion***) zu erkennen, d.h. eine Vorstellung davon zu gewinnen (*data mining*), welche Daten am ehesten die zentralen Kerninformationen der Grundgesamtheit oder die entscheidenden marginalen Grenzwerte abdecken (vgl. Auswahl starker und schwacher Merkmale für die Komponenten). Dieser Aspekt wird in Bezug auf die Verarbeitung von Kontext-Information wieder aufzugreifen sein (vgl. 4.3.2.).

Die theoretische Fehleranalyse (3.2.3.) spricht nicht gegen eine derartige Selektion, da eine Maßgabe, so viel Information wie möglich einzubringen, also die dynamischen Netze so weit wie möglich zu expandieren, nicht fortschreitend zu besseren Ergebnissen führt, sondern vielmehr die Clusterränder ab einem gewissen Punkt unübersichtlich ausfransen und dann die Klassengrenze diffus erscheinen läßt (*overfit*).

Eine Selektion von Daten auf 10% der Grundmenge an Merkmalsvektoren (Mustern), kann aber bei umfangreichen Grundmengen immer noch einen sehr großen Datenbestand übriglassen. Es wird also zu prüfen sein, inwieweit die beiden Netztypen unterschiedlich auf überdurchschnittlich große Datenmengen (bezogen auf die jeweilige Domäne) reagieren. Dies muß nicht nur ein Problem der Modell-Komplexität sein, sondern kann auch die Modell-Paßform günstig oder ungünstig beeinflussen, da in zur Laufzeit dynamischen Netzen auch die Anordnung der Lerndaten von Bedeutung ist. - Weiter ist zu berücksichtigen, daß bei den hier in Frage stehenden Experimenten zur Gesichtererkennung bei größeren Datenmengen die Klassen A und A^c nicht mehr in sinnvoller Weise annähernd dieselbe Anzahl von Elementen enthalten können.

Umgekehrt wäre ebenso zu prüfen, ob unterdurchschnittlich kleine Datensätze für die Performanz beider Netztypen verträglich sind, wenn z.B. nur in geringem Umfang Daten verfügbar sind. Dies könnte dazu führen, daß zum Lernen erforderliche Rausch-Varianten nicht in genügendem Umfang zur Verfügung stehen oder daß *bias*-Einflüsse verstärkt einwirken.

Während man fehlende A^c - Daten u.U. leicht aus anderen Quellen ergänzen kann, besteht für das Aufforsten fehlender A-Daten bei SVM-Netzen in der Erzeugung zusätzlicher sog. virtueller Support-Vektoren (VSV) eine Möglichkeit der Ergänzung, indem man z.B. ausnutzt, daß ein Muster ggf. nach Drehung, Spiegelung oder Translation nach wie vor derselben Klasse angehören würde (Invarianten). In der Forschungsliteratur ist dokumentiert, daß die Verwendung von VSVs die Konvergenz von Modellen verbessert [Saund] S. 12f. Da der genannte Effekt auf der Beschaffenheit der Lerndaten basiert, wäre er natürlich für RBFs grundsätzlich ebenfalls nutzbar.

Eine zweite Möglichkeit, gezielt die Modell-Komplexität zu verringern, besteht darin, die Anzahl der musterbestimmenden Merkmale der Dimension $m = m_1 * m_2$ unter kontextbezogenen Annahmen zu reduzieren. Dabei ist allerdings zu bedenken, daß für die Modell-Paßform eine solche Maßnahme erkennbar ein grundsätzliches Risiko birgt, da mit jedem gestrichenen Merkmal natürlich auch Information (von mehr oder weniger diskriminierender Bedeutung) verlorengeht, sofern man einen echten Unterraum des Eingaberaumes erreicht.

Für die vorliegende Domäne der Gesichtererkennung seien folgende beiden Beispiele für eine mögliche ***Reduktion*** genannt:

Variante 1: Aus der Bild-Matrix wird jede zweite Zeile und/oder jede zweite Spalte gestrichen aufgrund der Überlegung, daß man auch bei relativ grobrasterigen Tageszeitungs-Bildern Gesichter durchaus noch erkennen kann.

Variante 2: Aus der $m_1 * m_2$ - Bild-Matrix werden nur bestimmte Rechtecke (Augen-, Nasen-, Mundpartie) berücksichtigt aufgrund der Erfahrung (vgl. Puzzles, Ausschnitt-Erkennungsspiele), daß bestimmte Bereiche eines Gesichts deutlich mehr Information zur Diskriminierung enthalten als bestimmte andere (vgl. dazu das '*landmark*'-Konzept von **CRAW** u.a. mit einer Normierung der Bilder auf die rechte Iris des Abgebildeten [Crawl] S. 729.

Natürlich wäre auch in umgekehrter Richtung zu prüfen, inwieweit eine Expansion der Merkmale die Modell-Paßform verbessern kann, im vorliegenden Fall etwa durch Verdreifachung von ***dim*** m mittels Berücksichtigung von RGB-Farbwerten anstelle von Grautönen.

Abb.4.1.3.a Reduktionsvariante *landmarks*

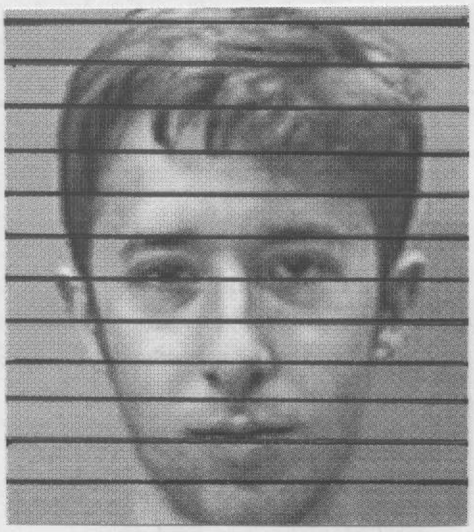


Abb. 4.1.3.b Reduktionsvariante *lawnmower*

4.2. Aspekte zur Bedeutung der Netzparameter
4.2.1. Bewertung der minimalen Beschreibungslänge

Während sich der Abschnitt über die Gütekriterien zunächst lediglich auf eine Bewertung der Ergebnisse einer Simulation bezog, soll an dieser Stelle weitergehend gefragt werden, welcher Aufwand betrieben werden muß, um ein Ergebnis einer bestimmten Qualität zu erzielen.

Unter dem Oberbegriff der Regularisierung bietet sich eine Möglichkeit, RBF-Netze und SVM-Netze mit RBF-Kernen dahingehend zu vergleichen, welches Verfahren unter welchen Bedingungen die geringere Redundanz innerhalb einer angemessenen Beschreibung der in Frage stehenden Anwendung enthält.

Um zu prüfen, wieviel Redundanz ein Netz enthält (von der zumindest ein Teil zur Desambiguierung auch benötigt wird), kann man seinen Graphen (s. Abb. 2.1.1., 2.2.1., 2.3.1.) als Abbild der Transformation von Eingabe-Daten in Ausgabe-Daten ausfassen, bei der jeder mögliche Pfad durch einen eindeutigen Code gekennzeichnet wird. Gefragt ist dann, ob die Länge dieses Codes optimal gewählt ist, um die Information aus den Eingabe-Daten auf die Ausgabe-Daten zu übertragen.

Die Codelänge ist ein Maß, um den Informationsgehalt (bit pro Zeichen) einer Nachricht zu beurteilen, die von einer Quelle (dem Stichprobenraum) durch einen Kanal transportiert wird. Die mittlere Codelänge beschreibt dazu insbesondere den Informationsgehalt in Abhängigkeit von der Wahrscheinlichkeit des Auftretens einer bestimmten Nachricht in der Übertragung, die minimale Beschreibungslänge des Codes (MDL) den optimal erreichbaren Informationsfluß.

Der Code ist die eindeutige (digitale) Beschreibung des Weges, auf dem die Eingabe von der Quelle durch den Kanal (das Simulationsnetz) zur Ausgabe gelangt.

Die totale Codelänge führt **RISSANEN** ein als

$$L(x,\Theta) = -\ln P(x|\Theta) + L(\Theta), \qquad \text{(F.4.2-1)}$$

wovon bzgl. Θ das Minimum gesucht ist.

Θ beschreibt dabei einen Parameter-Vektor, in dessen Komponenten z.B. die Wahrscheinlichkeiten für jeden Pfad gespeichert sind, und stellt insofern eine Aggregation der Netzstruktur mit allen Aktivationspotentialen der Knoten und allen Gewichten der Kanten dar.

$L(\Theta)$ bezeichnet nach **RISSANEN** die Beschreibungslänge dieses Parameter-Vektors, d.h. die Anzahl der benötigten Bits, ihn darzustellen [Rissa] S. 418 .

$P(\boldsymbol{X}|\Theta)$ beschreibt die Wahrscheinlichkeit der Eingabe-Daten ***X*** unter der Bedingung des gegebenen Parameter-Vektors Θ. **RISSANEN** erklärt $-\ln P(x|\Theta)$ als Maß für die ideale Codelänge für besser geeignet als die traditionelle mittlere Codelänge, wie sie im Zusammenhang mit der Entropie einer Quelle (eines Stichprobenraumes) definiert wird, da mit $-\ln P(x|\Theta)$ der Speicherbedarf der Daten (vgl. nicht-stationäre Quellen) mitberücksichtigt werden. - Der Autor zeigt später, daß die Entropie damit zum Spezialfall der MDL wird [Rissα] S.429 .

RISSANEN weist darauf hin, daß die Bestandteile der Formel ebenso eine informationstheoretische wie eine statistische Deutung zulassen [a.a.O.] S.418 und daß die Gleichung insgesamt dabei auffällig an eine logarithmisierte Form der BAYES-Formel erinnere [a.a.O.] S. 419 . Dasselbe gilt für den Schwarzschen Koeffizienten, einem der Gütekriterien aus der SNNS-Ausgabe (s. 3.3.2.). An dieser Stelle zeigt sich abermals, daß ein Paradigmenwechsel eine Zusammenführung verschiedener Einsichten ermöglicht.

Aus den Abbildungen 2.1.1. und 2.2.1. ergibt sich, daß die Eingabe stets einheitlich aus den m Vektorkomponenten, den Graustufenvalenzen der m Bildpixel, besteht, die Ausgabe einheitlich aus dem Endneuron, an dem die Zuordnung, die Klassenzugehörigkeit abgelesen wird.

Geht man wiederum von einer Gleichverteilung der Daten innerhalb der Quelle (im Stichprobenraum) aus, so ist die Codelänge dabei nicht mehr von der Zuordnung zu den einzelnen Elementen des Stichprobenraumes abhängig, d.h. es ist unbedeutend, welchen Elementen kürzere oder längere Codes zugeordnet werden.

Daraus folgt, daß der Code ausschließllich auf die Beschreibung des Weges durch die einzelne, vollständig vermaschte, verdeckte Schicht des jeweiligen Netzes reduziert werden kann, z.B. von den m Eingabe-Neuronen via Knoten 3,17 und 21 zum Ausgabeneuron würde z.B. durch den Code c(X)=00011 10001 10101 eindeutig beschrieben (Abb. 4.2.1.).

Somit ist die minimale Codelänge MDL bei Gleichverteilung der Daten im Fall der Kanäle RBF bzw. SVM ausschließlich von der unterschiedlichen Anzahl der Neuronen der einen verdeckten Schicht abhängig, also der Anzahl der Zentroide, $N_{Z,}$ bzw. der Support-Vektornen N_S. Daraus ergibt sich, daß die MDL für dasjenige Modell günstiger ist, das bei gleicher Ausgangs-Datenmenge weniger verdeckte Knoten besitzt. Diejenigen Forschungsansätze, die darauf abzielen, N_S bzw. N_Z zu reduzieren (R-SVM von **TIPPING,** CoVar-Ansatz bei **BISHOP),** reduzieren demnach auch unmittelbar die MDL.

Gerade deshalb erscheint allerdings die MDL eher als ein notwendiges, aber nicht hinreichendes Vergleichskriterium, weil bei $L_{RBF}(x,\Theta) = L_{SVM}(x,\Theta)$ durch die Verwendung unterschiedlicher Lernverfahren eine deutlich unterschiedliche Modellkomplexität entstehen kann.

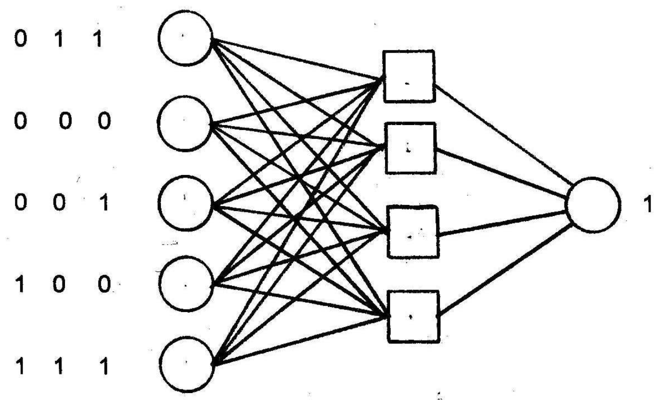


Abb.4.2.1. Struktur Code-Transfer

4.2.2. Filterverfahren für die Netzeingabe

Da die Lauf-Parameter σ_i bzw. γ keine wesentlichen Einflußmöglichkeiten vonseiten des Anwenders auf die Konvergenz der Netze eröffnen, sind für im Sinne von 4.1.2. zusammengestellte Trainings-Datensätze die unterschiedlichsten Filter-Maßnahmen vorgeschlagen worden, um auf der Ebene der Netz-Generierung die Laufzeit-Bedingungen zu verbessern:

Durch Randomisierung der Daten können *bias*-Effekte bekanntlich abgeschwächt oder unterdrückt werden. Mittels Substitution der euklidischen Distanz durch eine andere L2-Norm im Argument der Netz-Funktion können die Merkmale der Muster gewichtet werden. Für die Domäne der Gesichtererkennung ist etwa die Verwendung der Mahabolonis-Norm vorschgeschlagen worden, bei der ein multiplikativer Skalar die Distanzen qualifiziert [Crawl] S. 728.

Oft wird versucht, mittels einer der Simulation vorgeschalteten Bearbeitung der Daten einen ersten Überblick über die Grundgesamtheit zu gewinnen und die Simulationsergebnisse auf diese Weise zu verbessern.

Ein Orientierungsverfahren, das im Zusammenhang mit SVMs häufig erfolgreich Verwendung findet, ist die sog. (K)PCA ((kernel) *principal component analysis*), ein Eigenwertverfahren mit dem Ziel einer spezifischen Merkmals-Varianz-Analyse (Basistransformation im Merkmalsraum). Daraus sind Eigenheiten des Eingaberaumes rekonstruierbar [Schö7] S. 3f.

Zunächst wird für die Daten-Matrix $D_{n,m}$ die zugehörige (quadratische) $CoVar_{m,m}$ - Matrix für die Merkmale erstellt. Danach ist in der üblichen Weise aus der Determinante $|\, CoVar_{m,m} - \lambda E|$ das charakteristische Polynom φ zu bestimmen. Die Nullstellen λ_i des charakteristischen Polynoms sind die Eigenwerte von $CoVar_{m,m}$. $CoVar_{m,m}$ ist genau dann diagonalisierbar, wenn φ in Linearfaktoren zerlegbar ist. Zu jedem Eigenwert λ_i liefert die Lösung der homogenen linearen Gleichung $\boldsymbol{X}^T(CoVar_{m,m} - \lambda_i E) = \boldsymbol{0}$ einen Eigenvektor. Alle Eigenvektoren sind linearunabhängig und können folglich eine Basis des Daten(unter)raumes bilden (s. ergänzend ANHANG (7)).

Anhand des Eigenwertverfahrens wird die Datenmatrix also daraufhin untersucht, welche Vektorkomponenten (Merkmale) am meisten (Ko)Varianz zeigen (am meisten zur Diskriminierung beitragen); entsprechend werden durch Umsortieren Hauptkomponenten gebildet, die dann die Grundlage der Randfindung (Separierung) bilden. Das Verfahren entfaltet seine Wirkung demnach für $N_D >> m$.

Die PCA basiert somit ebenso auf der Idee, die Kovarianzen zwischen Merkmalen zu berücksichtigen wie auf anderer Ebene die Normierung der

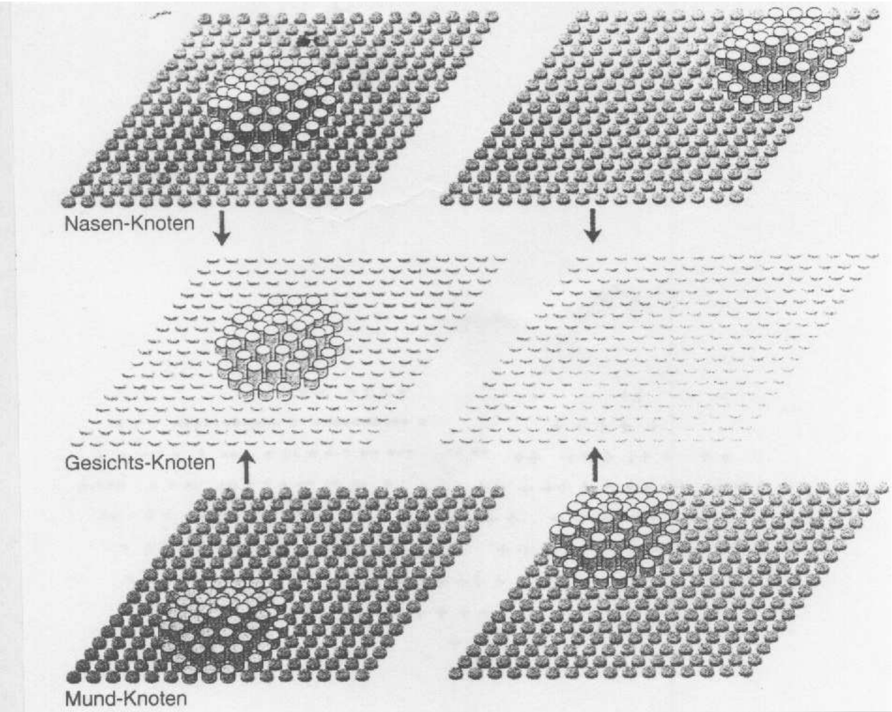


Abb.4.2.2. zu möglichen Wirkungen von PCA
Bild aus [HINTO] S. 143

RBF-Gaußfunktionen auf die Kovarianzen zwischen den Mustern. Ein Unterschied besteht darin, daß die Information im einen Fall im Vorfeld einer Simulation auf der Ebene der Datenmanipulation genutzt wird, im anderen Fall zur optimalen Gestaltung der verdeckten Schicht des Netzes.

Im Gegensatz zur (K)PCA-Variante könnte man das PCA-Verfahren natürlich auch für klassische RBF-Netze entlehnen.

Zur Bestückung von RBF-Netzen kann man etwa durch Cluster-Verfahren wie KMEANS oder ISODATA versuchen, Aufschluß über die mutmaßliche Lage der Zentroide und damit eine günstige Wahl der Stützstellen zu gewinnen. Dies ist für Formen der Netzinitialisierung per Hand vonnutzen, womit im Vergleich zu Formen der automatischen Initialisierung wechselnde Erfahrungen berichtet werden. Die Einschätzung der Ergebnisse wird allerdings dadurch erschwert, daß sich die Lernzuwächse i.a. nicht linear zum Aufwand einstellen und oft auf mehrere Einflüsse zurückzuführen sind.

4.2.3. Das *bias* / *variance* - Dilemma

Berechnet man den totalen Fehler eines Netzes in der üblichen Weise aus dem Erwartungswert (3.2.3.), so zeigt sich, auf welche Weise sich das sog. *bias/variance*-Dilemma auswirkt: Der Einfluß des *bias* (additiver Term w_0 bzw. b in in der Entscheidungs- bzw. Trennfunktion; im Sinne eines durchgängigen Abstands des Modells von der Realität bzw. der Randbreite zwischen den Clustern) eines Trainingsdatensatzes kann dadurch vermindert werden, daß der Umfang des Trainings vergrößert wird, was aber in der Folge die Varianz erhöht, d.h. beide Abweichungen entwickeln sich gegenläufig, so daß man abwägen muß, welches der beiden Probleme am ehesten in welchem Umfang tolerierbar ist.

Wenn man sich verdeutlicht, daß der Balancepunkt dieser Entwicklung, der Nullpunkt des totalen Fehlers, so zu interpretieren ist, daß das Netz "auswendig Gelerntes" exakt reproduziert, zeigt sich, daß der Begriff *bias/variance*-**Dilemma** eigentlich unberechtigterweise ein negatives Konnotat enthält, da das *bias/variance*-Phänomen genau das Wesen jedweden Lernens beschreibt, nämlich die erforderliche Abwägung zwischen Einseitigkeit und Geschlossenheit bei hoher Präzision des Gelernten einerseits gegenüber Ungenauigkeit bei entsprechender Flexibilität und Offenheit für Modifikationen wie Erweiterungen andererseits.

Im vorliegenden Fall ist von Interesse zu prüfen, inwieweit RBF-Netze und SVM-Netze unter gleichwertigen Struktur- und gleichen Trainingsbedingungen das *bias/variance*-Phänomen unterschiedlich absorbieren. Damit wäre es ggf. möglich, je nach Beschaffenheit des vorliegenden Problems die eine oder die andere Netzvariante zu bevorzugen.

Im Bereich der visuellen Mustererkennung etwa könnte ein evolutionsbedingter *bias* von der menschlichen Neigung verursacht sein, Abweichungen von Symmetrieeigenschaften als bedeutsamer einzustufen als Abweichungen, die symmetrisch auftreten. Ein sozialisationsbedingter *bias* wäre im Falle eines unterschiedlich zuverlässigen Erkennens weiblicher gegenüber männlichen Gesichtern vorstellbar oder in Form spezifischer männlicher gegenüber weiblichen Aufmerksamkeits-Dispositionen.

SCHÖLKOPF referiert im SVM-Manual einen modifizierten Kern, der auf einer Idee von **VAPNIK** beruht und mit dem eine globale mit einer lokalen Justierungs-Möglichkeit verbunden wird [Saund] S. 16.

K: ***IH*** -> ***IR***

$K(X,Xi) = ((X * Xi) + 1)^{\delta} \exp(-|X-X_i|^2 / 2\sigma_i^2)$. (F.4.2-2)

Das Lokalitätsproblem von klassischen RBFs erörtert auch Bishop [Bisho] S. 184

Es fragt sich allerdings, ob nicht die globale Justierung analog der lokalen über den Abstand zwischen X und Xi statt über die Länge von Xi (mittels Xi*Xi) normiert werden könnte. Der Wertebreich eines globalen

Faktors λ / $(|X\text{-}X_i|^2 + 1)$ wäre für $\lambda > 0$ ebenso ***IR***+, und für veränderte Werte von λ könnten auch dieselben Monotonieeigenschaften simuliert werden wie mittels $((X \;{}^*Xi) + 1)^{\delta}$.

Dieser Kern könnte geeignet sein, eine gewünschte Umgewichtung zwischen *bias*-Toleranz und Varianz-Höhe zu steuern.

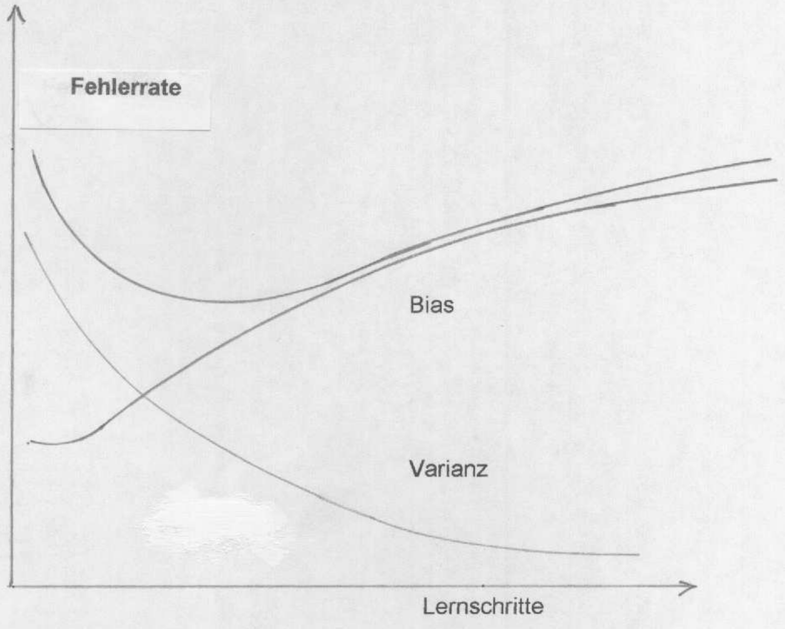


Abb.4.2.3. Das Bias-Varianz-Dilemma
(am Beispiel einer Regressionsschätzung)

4.3. Aspekte zur Gestaltung der Lernprozesse
4.3.1. Überlegungen zur Trennschärfe

Als Ergebnis der Überlegungen von 4.1. und 4.2. kann nunmehr die Übersicht von Abb. 2.3.2.a in folgender Weise vervollständigt werden:

$_{(K)PCA}\ SVM()_{VSV}$

$_{PCA}\ RBF_1$ $_{(K)PCA}\ SVM(RBF_1)_{VSV}$

$_{PCA}\ RBF_2$ $_{(K)PCA}\ SVM(RBF_2)_{VSV}$

$_{(K)PCA}\ SVM_{CG}\ (RBF_1)$ $_{(K)PCA}\ SVM_{CG}\ (RBF_2)_{VSV}$

Es bleibt zu fragen, inwieweit auf dieser Grundlage Bedarf und Möglichkeiten bestehen, mit nachgeschalteten Verfahren die Ergebnisse von $_{(K)PCA}\ SVM_{CG}\ (RBF_2)_{VSV}$ zu verbessern.

Defizite in der Trennschärfe können an den Rändern eines Clusters bekanntlich in zwei Richtungen auftreten: Entweder das Verfahren zeigt eher die Neigung, nicht-zugehörige Instanzen fälschlich anzuerkennen (Irrtum, α-Fehler), wenn Übereinstimmungen in den Eigenschaftsmustern stärker gewichtet werden als Abweichungen, oder eher die Tendenz, zugehörige fälschlich zurückzuweisen (Risiko, ß-Fehler), wenn abweichende Merkmale höher eingestuft werden als übereinstimmende.

Die Zusammenhänge lassen sich wiederum anhand der BAYES-Formel erläutern: Sei Ereignis ***X***1: Muster gehört zu Klasse A; Ereignis ***X***2: Muster gehört zu Klasse A^c; bedeute weiter Ereignis X: Muster ist fehlerhaft zugeordnet und wie üblichP(***X***1) = Wahrscheinlichkeit, daß Muster zu A gehört, P(***X***2) = Wahrscheinlichkeit, daß Muster zu A^c gehört, P(X|***X***1)= Wahrscheinlichkeit für ein fehlerhaftes Muster aus A, P(X|***X***2)= Wahrscheinlichkeit für ein fehlerhaftes Muster aus A^c.

Dann kennzeichnet P(***X***1|X) gerade die α-Fehler (Irrtümer) einer 2-Gruppen-Klassifikation, d.h. die Wahrscheinlichkeit für ***X***1 unter der Bedingung, daß x fehlerhaft ist;
P(***X***2|X) gerade entsprechend die ß-Fehler (Risiko) einer 2-Gruppen-Klassifikation.

Im Falle der Gesichtererkennung könnte man mit Effekten selektiver Wahrnehmung rechnen oder der Antizipation eines gewünschten Ergebnisses: Sucht man einen Bekannten in einer Zuschauermenge, wird man eher bei fremden Personen glauben, ihn ausgemacht zu haben (α-

Fehler) als ihn nicht zu erkennen, obwohl man ihn sieht (ß-Fehler). Soll man als Zeuge in einem Kriminalfall aussagen, wäre eher zu erwarten, daß jemand aus Vorsicht oder Angst einen Täter nicht wiedererkennt (defensives Verhalten; ß-Fehler), während bei der Aussage eines Opfers womöglich wiederum eher α-Fehler (offensives Verhalten) zu befürchten sind. - Am Rande sei daran erinnert, daß man durch Problemtransformation α- und ß-Fehler tauschen kann.

Da nun SVM als Konzept gelten, den strukturellen Fehler zu minimieren, während RBF-Netze den empirischen Fehler minimieren, könnte man erwarten, daß sich beide Verfahren gegenüber α- bzw. ß-Fehlern unterschiedlich anfällig zeigen.

Ja nachdem, ob eher eine offensive oder eher eine defensive Tendenz erwünscht ist, ob eher α- oder eher ß-Fehler tolerierbar erscheinen, könnte man
bei mehreren parallel implementierten Netz beider Typen zu einer bestimmten Aufgabe die Entscheidungssicherheit erhöhen, indem man jede Klassifikationsentscheidung mehrheitlich von *zwei* Netzen des einen Typs (mit unterschiedlichen Kernfunktionen oder unterschiedlicher Initialisierung) und *einem* Netz des alternativen Typs tendenziell defensiv oder tendenziell offensiv treffen lassen (stat. 2von3 Term).

Während diese Verfahren unter Komplexitäts- und Kostenaspekten eher skeptisch zu sehen ist, könnte es im Hinblick auf die Entscheidungssicherheit u.U. einen Gewinn bedeuten.

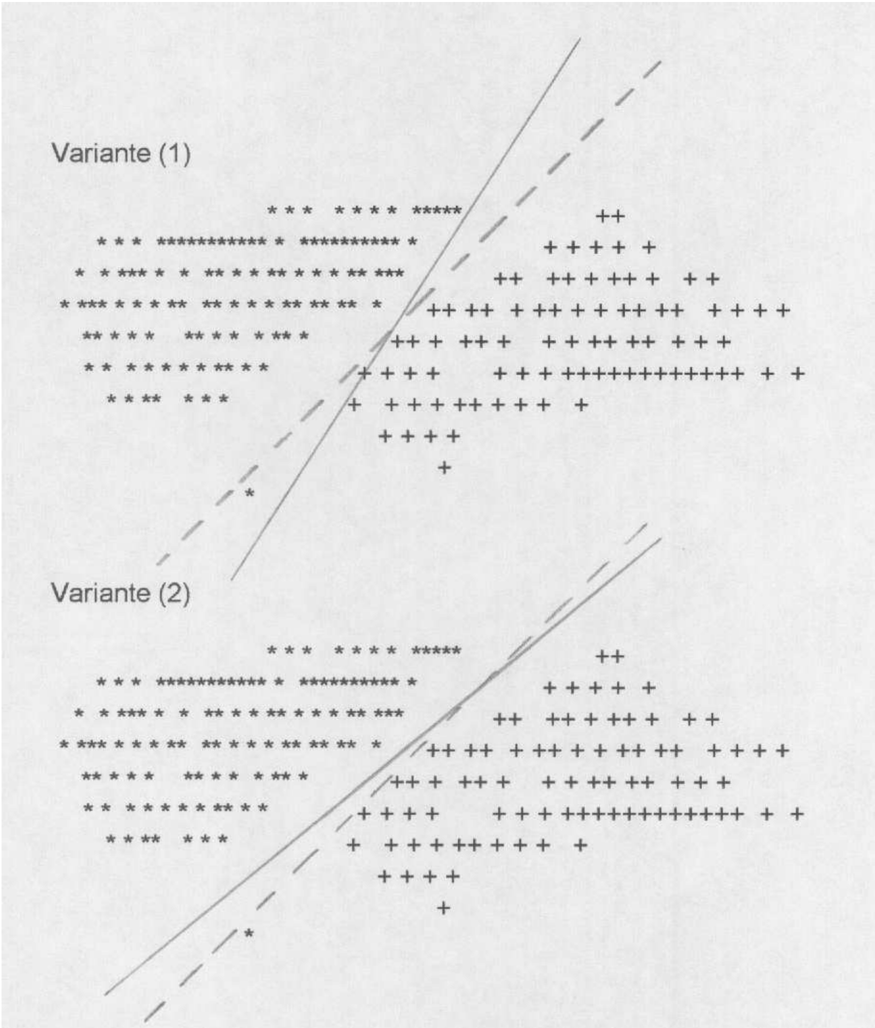


Abb.4.3.1. Behandlung isoliert auftretender Muster
nach [**Zhang** S.7]

4.3.2. Grundlagen für einen Rückabgleich

Als Gegenstück zur Vorstrukturierung der Lerndaten (4.1.2.) gibt es auch Verfahren zur Nachbearbeitung der Simulationsergebnisse (womit nicht Nachtraining bei unbefriedigender Konvergenz gemeint ist). Während der Fusionsgedanke von 2.3. auf den gemeinsamen Bereich RBF $\cap$ SVM abzielt, liegt das Augenmerk bei den Nachbessungsverfahren gerade im Bereich der spezifischen Eigenheiten RBF $\cup$ SVM \ (RBF $\cap$ SVM).

Unter dieser Perspektive könnten spezifische Vorzüge beider Netz-Typen dadurch genutzt werden, daß man nach der Trainingsphase die Testergebnisse des eines Netzes zum Nachtrainieren des jeweils anderen verwendet (*crossvalidation*), d.h. sozusagen Pseudo-Vorwissen nutzbar macht (*parallel second view*), ohne dafür eine symbolisch codierte Wissenskomponente zu benötigen, oder indem man zwei unterschiedliche Netze so organisiert, daß ein *master*-Netz als Vorwissen die Einschätzung eines vorgeschalteten Netzes kennt und berückichtigt (*serial second view*), z.B. ein RBF-Netz dient zur Vorbereitung der Gewinnung virtueller Supportvektoren. Man würde also Verbesserungen suchen durch gleich trainierte unterschiedliche Netze oder unterschiedlich trainierte gleiche Netze.

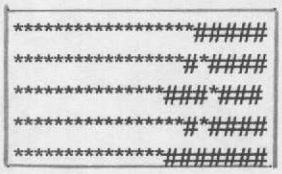
HERMES u.a. konstruierten einen BAYESschen Schätzer (vgl. 4.1.1.), mit dem er das Ergebnis einer SVM-Klassifikation konfrontiert [Herme] S.2 . Die Verschränkung beider Verfahren besteht dabei darin, daß der Algorithmus nicht die geschätzten Klassenwahrscheinlichkeiten - bezogen auf die Grunddatenmenge - verwendet, sondern die SVM-ermittelten kumulativen Wahrscheinlichkeiten aus der Klassifikation (Risikoreduktion).

Ziel ist es, die durch SVM gewonnene Klassifikation so nachzujustieren (bereinigen), daß offensichtliche Fehler (Ausreißer) korrigiert werden. Einen solchen Ansatz zu erwägen und zu konstruieren, setzt aber, was bei **HERMES** u.a. nicht weiter gewürdigt wird, kontextuelles Vorwissen voraus, hier etwa zu beschreiben als: in einem abgrenzbaren Flurbereich (Feld, Straße, Grundstück vs. Wald, Fluß) gibt es üblicherweise keine insularen Bereiche, die nicht dem jeweiligen Cluster angehören. Als Entscheidungskriterium wird lediglich eine 'Kostenfunktion' vorgeschlagen, die als Schranke dafür gilt, in welchen Fällen das Ergebnis des Bayesschen Schätzers als Korrektiv herangezogen werden soll und in welchen nicht.

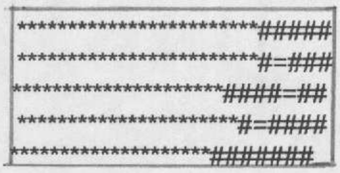
Der Forschngsbericht zeigt aber, daß die filigranste statistische Aufbereitung von Netzausgaben dennoch in die Irre führen kann, wenn man auf die Einbringung von Kontextwissen zur der Interpretation der Ergebnisse verzichtet.

Klassifikationsaufgabe

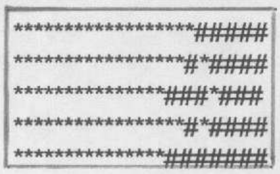
z.B.
eingescannte Textur mit $* \varepsilon A$, $\# \varepsilon A^c$
= Nachbearbeitung im Sinne von HERMES



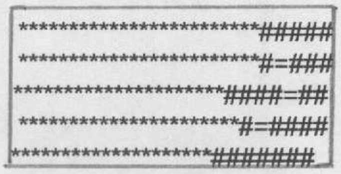
Fall (1 - vor)



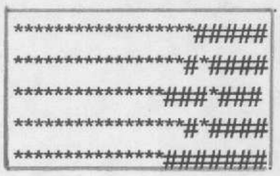
Fall (1 - nach)
Interpretation plausible
Nachbarschaftsbeziehung:
> restaurieren



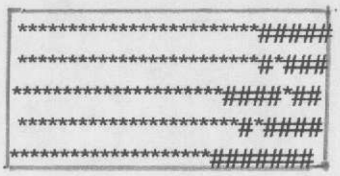
Fall (2a - vor)



Fall (2a - nach)
Interpretation Knickstelle:
> restaurieren



Fall (2b - vor)



Fall (2b - nach)
Interpretation Knickstelle:
> original belassen

Abb.4.3.2. Beispiel Kartierung

Anhand seiner geographischen Kartierung kann man sich vergegenwärtigen, daß der Nachbarschaftsabgleich bzgl. eines Gebietes, das für eine Straße steht, plausibel sein kann (größere Gesteinsbrocken oder singuläre Bäume befinden sich i.a. nicht auf Straßen), für einen Fluß aber u.U. keinen Sinn macht, da er durchaus kleinere oder größere Inseln, sogar gegründete Bauwerke enthalten kann - ein anschauliches Beispiel für das Kriterium der Modell-Schlüssigkeit (3.3.3.).

4.3.3. Vererbung von Eigenschaften

Da die verdeckten Knoten eines RBF-Netzes nicht nur alle in ein und derselben Schicht liegen, sondern auch dynamisch sukzessive konstruiert werden, gibt es, abweichend von der allgemeinen Eigenheit neuronaler Netze, den Weg von der Eingabe zur Ausgabe nicht auf symbolischer Ebene nachvollziehen zu können, bei RBF-Netzen günstigere Möglichkeiten, eine ***Explikationskomponente*** zu konstruieren, indem man einzelne Knoten mit Bedeutung attribuiert (da jede Stützstelle ein vollständiges Muster repräsentiert) und interpretiert (was auf der Grundlage von 2.3. folglich auch zumindest für bestimmte SVM-Netze möglich wäre).

Das darf zwar einerseits nicht dazu verleiten, das sog. "Großmutter-Neuron" synthetisieren zu wollen, (wie dieser Irrweg inzwischen ironisch kolportiert wird), andererseits ist aber genau dies ein zentrales Merkmal menschlichen Denkens, daß aus neuronalen Prozessen irgendwie auf komplexe Weise symbolisch kodierte Information gewonnen und weiterverarbeitet wird.

Eine solche Gewinnung und Weiterverarbeitung symbolisch kodierter Information könnte auf zweierlei Wegen erfolgen:

(W1) Planung eines erkenntnisgeleiteten Nachtrainings anhand der Ergebnisse der Trainingsphase für Bedarfsfälle (was natürlich für RBFs ganz anders ablaufen würde als für SVMs);

(W2) Einbringen von Kontext-Informationen, ohne allerdings ein symbolisch kodiertes Modell von der Domäne einzuführen

Es sollte also nicht darum gehen, das gerade typische subysmbolische Arbeiten eines NNs durch Überstülpen eines Modells von der Anwender-Domäne zu verwässern, d.h. z.B. eine algorithmische Schablone zu entfernen, die beschreibt: Was sind die Merkmale eines Gesichtes? Jede explizite Vorannahme in diese Richtung würde auch die Möglichkeiten der Generalisierbarkeit (z.B. eine Erweiterung der Anwendung auf Tiergesichter) unnötig einschränken.

Sehr wohl könnte es aber der Modell-Angemessenheit ebenso nutzen wie der Modell-Komplexität, wenn man z.B. bestimmte, erwartbare Symmetrie-Eigenschaften in einem Gesicht (wie etwa ein Abgleich zwischen paarweise vorhandenen Partien) als algorithmische Verzweigung einarbeiten würde. Dies wäre der Nutzung von Invarianten zur Generierung virtueller Support-Vektoren vergleichbar.
Der Nutzen symbolisch codierter Information kann also weniger in der Verschränkung beider Ansätze ausgeschöpft werden als in der Zusammenfügung getrennt gewonnener Teilergebnisse (Konkatenation).

Aus der Grammatiktheorie kennt man den Ansatz, kontextuelle Information derart über Pfade zu transportieren, daß man Knoten synthetisch attribuiert und diese Attribute nach festen Regeln über die Knoten vererbt. Dieses Verfahren kennt man von den Syntaxbäumen im Compilerbau. In ähnlicher Weise könnte in den Pfaden von Netzen eine algorithmisch gesteuerte Vererbung organisiert werden, um Externwissen in einer späteren Netzgeneration in Internwissen (s. 3.1.3.) zu überführen (vgl. induktives Erlernen von Bedeutung; assoziatives Erfassen).

Beispiele für solche Knotenattribute könnten etwa sein:
(1) Nebenbedingungen (*constraints*) über Nachbarschaftsbeziehungen zwischen Mustern; wenn z.B. aus einer Filmsequenz oder einem Fotosatz bekannt ist, daß zwei Gesichter sehr unterschiedlichen Ausdrucks oder aus unterschiedlichen Zeiten mit Sicherheit dieselbe Person zeigen.
(2) notwendige Bedingungen (*preconditions*) innerhalb von Mustern wie Augenfarbe oder andere sog. 'unveränderliche Kennzeichen' (d.h. Identität von Matrixwerten in ganz bestimmten Positionen/Feldern), d.h. Merkmale, bei denen u.U. jegliches Rauschen ausgeschlossen werden kann.

Wie in einem biologischen Bauplan könnten auf diese Weise sogar über einen Metaalgorithmus Modifikationen an den Objekt-Algorithmen vorgenommen werden. Auf welche Weise man etwa semantische Regeln für die Vererbung konstruieren müßte, kann allerdings im Rahmen der vorliegenden Arbeit nicht weiter vertieft werden.

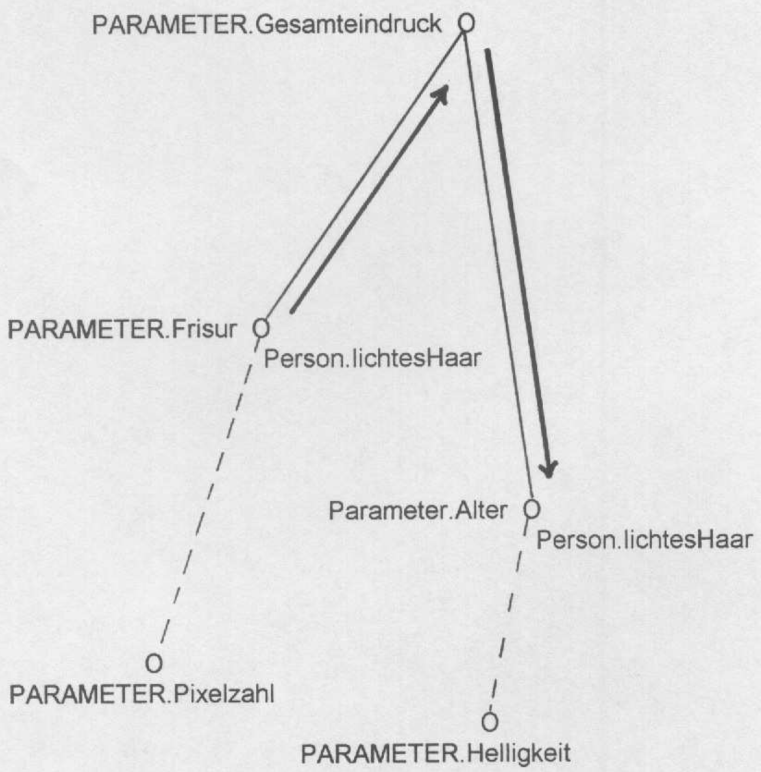


Abb.4.3.3. Vererbung im Strukturbaum
für einen Informationsverarbeitungs-Algorithmus

5. Diskussion der Ergebnisse und Vergleich

5.1. Kontrolle der Hypothesen

Der Basis-Datensatz wurde aufgrund folgender Überlegungen auf (06/30) festgelegt:

(1) Der Datensatz sollte die Zahl von 40 Mustern nicht wesentlich unterschreiten, da diese Größenordnung bei der Gesichtererkennung als erfolgversprechend für eine fehlerfreie Anpaßbarkeit gilt (s. 3.2.1.).
(2) Der Datensatz sollte die Zahl von 40 Mustern nicht wesentlich überschreiten, da die Entwicklung bei der Anpassung leichter erkennbar ist, wenn die Simulationen (zunächst) noch oberhalb der 0-Fehler-Grenze bleiben. Bei Simulationen, die mehr oder weniger weit unterhalb der 0-Fehler-Grenze liegen, müßte rekonstruiert werden, ob unter Umständen schon mehr Lernddaten eingesetzt worden sind, als erforderlich gewesen wäre.
(3) Es sollte bei card(A)=10 möglich bleiben, Elemente aus A in den Testdaten-Satz einzubringen, die im Trainings-Satz noch nicht verwendet worden waren.
(4) Es sollte versucht werden, das erwähnte strukturelle Problem für die Fragestellung von H.I, card(A^c) >> card (A), möglichst gering zu halten. (Ansonsten bieten die Fragestellungen von H.IIa/H.IIb die Möglichkeit, auch größere Lerndatensätze mit card (A) = card(A^c) zu erstellen.)

H.I Gesichtererkennung mit dem Basis-Datensatz (1) (s. ANHANG)

Die Anzahl der Iterationen bis zum Abschluß der Optimierung bei SSVM(RBF_1) bzw. SVM_{CG}(RBF_1) lag bei 630 bzw. 2*630 und bis zum asymptotischen Erreichen der Steigung 0 für die Fehlerkurve bei SNNS(RBF_1) bei 180, was aber, wie bereits diskutiert (s. 3.3.3.), nicht unmittelbar verglichen werden kann. Beide Zyklen-Anzahlen reichten nicht aus, um einen fehlerfreien Traingsstand zu erreichen. Die Fehlerrate lag bei 11% für beide SSVM bzw. 14% für SNNS. Dies dürfte auf die Konstellation Anzahl der Merkmale= m >> n =Anzahl der Lerndaten (Muster) zurückzuführen sein.

Betrachtung von H.I
unter den einzelnen Aspekten anhand der Kriterien (in der Numerierung von Abschnitt 4.*.*):

H.I - K4.1.1. "Generalisierbarkeit"

Egal, ob man Trainings- und Testdatensatz von Datensatz (1) tauscht, zum Testen einen weiteren Datensatz gleichen Zuschnitts einführt (Datensatz (9)) oder mit einem alternativen Trainings- und Testdatensatz arbeitet (Datensatz (0)): Die Ergebnisse verschlechtern sich für keinen der drei Simulatoren erkennbar, d.h. es gibt keinen Anhaltspunkt dafür, die Generalisierungsfähigkeit unterschiedlich einzuschätzen.

H.I - K4.1.2. "Bereichsweise Verteilung"

Die Ergebnisse mit dem künstlich erzeugten *bias* (Datensatz (3)) sind erkennbar schlechter als beim Standarddatensatz, d.h. bei SNNS(RBF_1) fällt die Fehlerkurve deutlich flacher ab, bei SSVM(RBF_1) und bei $SSVM_{CG}$(RBF_1) bleibt nach der Optimierung die Fehlerrate höher als beim Standarddatensatz.

H.I - K4.1.3. "Erfassung Grundgesamtheit"

Das Training mit dem besonders kleinen Datensatz (2) führte für alle Simulatoren zu Ergebnissen, die denen mit dem Standarddatensatz (1) gleichwertig waren, sowohl, wenn man gegen einen ebenso kleinen Datensatz testete, als auch, wenn man zum Testen den größeren Standarddatensatz selbst heranzog. Diese Robustheit konnte im Vergleich zwischen SVM(RBF_1) und SVM_{CG}(RBF_1) anhand der Theorie erwartet werden, für SNNS(RBF_1) war sie eher überraschend, korrespondiert aber mit der allgemeinen Beobachtung, daß die Fehlerkurve bei Datensätzen mit weniger als etwa 100 Mustern im Bereich der ersten etwa 50 Iterationszyklen gewöhnlich sehr steil abfällt, daß danach aber die Fehlerrate pro Zyklus nur noch geringfügig gesenkt werden kann.

Das Training mit dem großen Datensatz (4) führt zu einer erheblich erhöhten Fehlerrate bei SVM(RBF_1) und auf kaum niedrigerem Niveau für SVM_{CG}(RBF_1) bzw. zu einer sich bereichsweise plateauartig entwickelnden Fehlerkurve bei SNNS(RBF_1), knapp oberhalb des Niveaus des Standarddatensatzes. Verfolgt man die Reihenfolge der Eingaben, so zeigt sich, daß hier offensichtlich das bereits erwähnte Problem card (A^c) >> card(A) Wirkung zeitigt, d.h. die Gewichte können nicht mehr bzgl. der Knoten von A und A^c in gleicher Weise nachtariert werden.

Der *landmark*-Datensatz (5) reichte für alle drei Simulatoren aus, um ein Ergebnis mit einer Fehlerrate knapp unterhalb des Basis-Datensatzes zu erzielen, wobei der Gewinn im Selektions-Koeffizienten bei SSVM größer ist als bei SNNS. Bzgl. der Fehlerrate hat sich also das Herausnehmen nicht-relevanter Merkmale günstig ausgewirkt. Die Komplexität änderte sich nicht, d.h. sie hing (hier) offenbar nur von der Anzahl der Lerndaten, nicht von der Anzahl der Merkmale ab.

Mit dem *lawnmower*- Datensatz (6) konnte keines der Netze erfolgreich trainiert werden (Fehlerquote über 50%, also über der Würfelquote bei SVM bzw. *overflow*-Abbruch bei SNNS), was im Nachhinein auf einen nicht mehr rekonstruierbaren Irrtum bei der automatischen Matrix-Zeilen-Selektion im Lerndatensatz zurückgeführt wird.

H.I - K4.2.1. "Minimale Beschreibungslänge"

Der angestrebte Vergleich mit den spezialisierten CoVar-RBF_2 -Netzen führte zu keinen verwertbaren Ergebnissen (d.h. es traten gegenüber der Standardsimulation zu H.I keine erkennbaren Unterschiede auf), weil offensichtlich zu oft nicht-reguläre CoVar-Matrizen auftraten (eine Printmarke liefert hierfür den Indikator), so daß die entsprechende Alternativ-Normierung nicht greifen konnte.

Das Ergebnis erscheint insofern als kontra-intuitiv, als gerade bei derart geringen Differenzen in den Merkmals-Valenzen eine besondere Wirkung der Umnormierung hätte erwartet werden können.

H.I - K4.2.2. "Filterverfahren"

Für kleine N_D mit $N_S = N_D$, insbesondere dann, wenn $m >> N_D$, kann PCA ihre spezifische Wirkung nicht entfalten. Der Versuch bleibt insofern für alle drei Simulatoren im vorgegegeben Rahmen von Datensatz (1) ohne interpretierbares Ergebnis, da die erzeugten Datensätze in der Größe unterhalb der Verwertbarkeitsgrenze liegen.

H.I - K4.2.3. "bias / variance -Dilemma"

Der Versuch mit dem lokalisiert-globalen Vapnik-Kern führt mit Datensatz (1) bzgl. Fehlerrate und Iterationsanzahl zu denselben Ergebnissen wie für SVM(RBF_1) und SVM_{CG}(RBF_1), lediglich die numerischen Werte der Lagrange-Koeffizienten weichen ab..

H.IIa - K4.3.1. ”Trennschärfe”

Der Versuch mit dem Datensatz (7) zur Unterscheidung männlicher und weiblicher Gesichter erbrachte eine Fehlerrate (um 15%) für SVM(RBF_1) und SVM_{CG}(RBF_1), wobei SVM_{CG}(RBF_1) 2 Supportvektoren weniger benötigte. Die Fehlerrate von SNNS(RBF_1) stabilisierte sich knapp *darunter*. Das Ergebnis liegt im Vergleich zu H.IIb erstaunlich niedrig (im Bereich von H.I) - wahrscheinlich, weil der Merkmals-Bereich der Frisurpixel bei den vorliegenden Bildern (ohne Fälle von Verkleidung) überdurchschnittlich markant ausfällt und gute Anhaltspunkte bietet.

Besonders augenfällig war, daß in dem Versuch mit den unmarkierten Aufnahmen des ambivalenten Satzes s17 einzelne *items* s17_* nach Training der Klasse A, andere der Klasse A^c zugeordnet wurden.

H.IIb - K4.3.2. ”Rückabgleich”

Zum Versuch mit dem Datensatz (9) zur Klassifizierung einer Altersskala: Die Ergebnisse fallen hier wie erwartet kritischer als bei H.I mit Datensatz (1) aus (Fehlerrate über 30%), da die Unterschiede in den Merkmalsausprägungen zwischen den Mustern verschiedener Klassen bei diesem Ansatz noch geringer ausfallen.
Die Ambivalenzen, die auch einen menschlichen Entscheider treffen würden (sie gehen ja in die Markierung des Datensatzes ein), schlagen durch. Es treten überdies deutlich mehr α-Fehler als ß-Fehler auf, was darauf hindeutet, daß die Unterschiede zwischen den Gesichtern verschiedener Personen nicht unbedingt als größer eingestuft werden als die Unterschiede zwischen verschiedenen Gesichtsausdrücken einer Person. Das wiederum dürfte daran liegen, daß die Anzahl Merkmale $m >> n$, die Anzahl der Muster, so daß das Netz nicht ausreichend zwischen zufälligen und diskriminierenden Abweichungen unterscheiden kann.

H.III - 4.3.3. “Explikationskomponente / Vererbung ”

Die verfremdeten Bilder von Datensatz (9), bei denen die Originale mittels Bildbearbeitungs-Algorithmen z.T. verschattet, z.T. verschliert und z.T. entprägt wurden, führten zwischen SVM_{CG}(RBF_1), SVM(RBF_1) und SNNS(RBF_1) zu genau proportional besseren Ergebnissen als bei H.I mit Datensatz (1), offenbar weil die beschriebenen Verfremdungen die

Unterschiede zu jedem Originalgesicht deutlich größer werden ließen als zwischen zwei Originalgesichtern.

Diese Vermutung korrespondiert allerdings nicht mit dem Ergebnis, daß in diesem Falle bei SVM(RBF_1) und bei SVM_{CG}(RBF_1) eine Tendenz zu ß-Fehlern bestand, d.h. die gedachte Trennlinie wurde tendenziell zwischen Varianten eines Gesichtes angesiedelt, statt zwischen Gesicht und Nicht-Gesicht.

5.2. Folgerungen aus den Ergebnissen

Die Vorläufigkeit der Ergebnisse von 5.1. muß vor dem Hintergrund der vorhandenen Datensätze gesehen werden, die noch in mehrfacher Hinsicht weiter ausgewertet werden könnten (z.B. in Hinblick auf *cross validation*), was aber in dem hier vorgegebenen Rahmen nicht mehr möglich ist.

Es hat sich gezeigt, daß die Simulation mit komplexen Modulen wie SNNS und SSVM, wenn sie über die reine Demonstration mit sog. *toy examples* hinausgeht und darüber hinaus partielle Weiterentwicklungen anstrebt, ohne einen direkten und kontinuierlichen Kontakt zur Entwicklergruppe der ursprünglichen Programme kaum zu bewerkstelligen ist. Erschwerend war, daß diese beiden Simulatoren keinem einheitlichen Programmierkonzept verpflichtet sind (was sich besonders drastisch in den E/A-Strukturen sowie im Zuschnitt der Manuale zeigt).

Während schon die theoretischen Unterschiede zwischen den beiden zu vergleichenden Verfahren einen differenzierten Versuchsaufbau erforderten, steigerte sich der Bedarf an Fallunterscheidungen und Einschränkungen durch diese ingenieurtechnisch bedingten Unterschiede zwischen beiden Simulatoren in einem Ausmaß, das üblicherweise nur noch von kleinen Forschungsgruppen zu bewältigen ist.

Darüber hinaus muß erwogen werden, ob zunächst Vortests mit einfachen Texturen im Sinne eines *benchmarking* erforderlich wären, um die entwickelte $SVM_{CG}(RBF_1)$ gegenüber den beiden anderen Simulatoren genauer zu justieren. Daran anschließend könnte die gewählte realitätsnahe und anwendungsrelevante, aber damit auch sehr komplexe Domäne zu Ergebnissen führen, die wie gewünscht in engem Vergleich zu menschlichen Entscheidungs-Prozessen in ihrer Vielschichtigkeit zu interpretieren sind.

Die bisherigen Ergebnisse zeigen, daß die geringe Anzahl variierbarer Netzparameter zurecht als vorteilhaft eingestuft (2.1.2.) wurde, weil auf diese Weise nicht komplexe Syndrome von Wirkungszusammenhängen untersucht werden mußten. Es hat sich allerdings bestätigt, was bereits sporadisch in der Forschungsliteratur referiert worden war, daß die Wirkung der zentralen Netzparameter σ_i (respektive γ) nur sehr begrenzt ist und i.a. weder die Konvergenzmöglichkeiten noch die Konvergenzgeschwindigkeit wesentlich verbessern kann.

Entscheidend ist offenbar, wieviel Support-Vektoren bzw. Zentroide erzeugt und verwendet werden. Dies ist nach Beginn der eigentlichen Simulationsphase bei RBFs nur noch bedingt (über eine Konvergenzschranke) steuerbar, bei SVMs prinzipiell nicht mehr. Die Anzahl Iterationen bezieht sich bei diesen Netzen lediglich auf die Qualität der numerischen Annäherung an den exakten Optimierungsprozeß. Es bleibt festzuhalten, daß eine möglichst niedrige Anzahl SVs bzw. ZTs zwar notwendige Bedingung für die günstigste Modellkomplexität ist, daß aber die Modellpaßform u.U. von Fall zu Fall das Arbeiten mit einer nicht nicht-minimalen Anzahl nahelegt. Es ist jedoch nicht so (s. 4.2.3.), daß eine fortlaufende Vergrößerung der Anzahl SVs bzw. ZTs fortlaufend zu einer Verbesserung der Modellpaßform führen würde.

Hingegen hat sich gezeigt, daß die diversen Verfahren, die Einfluß auf die Auswahl, Reihenfolge oder Aggregation der Lern-Daten nehmen (s. 4.1.* und 4.2.*), offensichtlich entscheidende Veränderungen im Konvergenzverhalten herbeiführen können.

> (Z1) Fusion

Insgesamt hat die Modellvariante $SVM_{CG}(RBF_1)$ (s. 2.3.2. und 2.3.3.) mindestens so gut abgeschnitten wie $SVM(RBF_1)$, und diese bis auf einen Fall besser als $SNNS(RBF_1)$. Die Unterschiede fielen allerdings in den meisten Fällen so gering aus, daß die gewünschte Signifikanz bei größeren Datensätzen nicht absehbar ist.

Die $SVM_{CG}(RBF_1)$ konnte zwar im untersuchten N_D-Bereich die Fehlerrate nicht durchgängig senken, kam aber mit O(2n) im Vergleich zu $SVM(RBF_1)$ mit weniger Stützstellen μ_i aus, als jene Supportvektoren ξ_i benötigte. Hält dieser Effekt für größere $N_{D,}$ so wäre dies von erheblicher Bedeutung und könnte dann auch in einer Sekundärwirkung u.U. die Fehlerrate senken.

>(Z2) Integration

Es wurden mehrere Gütekriterien formuliert oder zusammengeführt, dabei eine Risikofunktion, die aus dem BAYES-Theorem motiviert ist. Die Gütekriterien können natürlich anhand der vorliegenden Arbeit nicht abschließend beurteilt, aber an den vorausgehenden Interpretationsmöglichkeiten (5.1.) gemessen werden.

Viele theoretisch bedeutsame Maße wie die VC-Dimension, aber auch die übliche Bestimmung von Komplexitätsklassen liefern kaum Maße, die für praktische Anwendungen aussagekräftig wären. Insbesondere, wenn man an Realzeit-Anwendungen denkt, nützt es wenig, die Anzahl benötigter Iterationen zu kennen, (zumal wenn diese von Verfahren zu Verfahren etwas anderes bedeuten, d.h. in unterschiedlicher Weise von $n=N_D$ und m=dim abhängen). Der Anwender wird vielmehr gerade bei lern-orientierten Algorithmen etwas über die tatsächlich benötigte *cpu*-Zeit (in einer bestimmten *hardware*-Umgebung) wissen wollen. Diese wiederum ist aber nur vergleichbar zu ermitteln, wenn die in Frage stehenden Verfahren in dieselbe Programmierumgebung implementiert sind.

Insbesondere wurde begründet, warum für den hier in Frage stehenden Anwendungsbereich eine Erweiterung der üblichen Kategorien von Modell-

Paßform und Modell-Komplexität benötigt wird, die domänen-sensitive Einschätzungen ermöglicht (Modell-Schlüssigkeit) (s. 3.3.3.).

>(Z3) Expansion

An den entsprechenden Stellen wurde vermerkt, wo und wie Verfahren zur Nachbesserung die erzielten Ergebnisse verfeinern könnten, was jedoch im Rahmen dieser Arbeit nicht überprüft werden konnte. Das unter 4.3.3. skizzierte Konzept zur Vererbung könnte jedoch (ähnlich wie das BAYES-Theorem als Interpretationsparadigma) eine Grundlage für die Zusammenführung und einheitliche modulare Strukturierung von Nachbesserungs- bzw. Rückabgleichsverfahren liefern.

Als Resümee kann damit ein Raster angeboten werden, anhand dessen gezielte, Signifikanz anstrebende Einzeluntersuchungen zu allen unter 4. aufgeworfenen Fragen konzipiert werden können. Es enthält zeilenweise drei unterschiedliche Verfahren unter drei spaltenweise aufgeführten Kriterien, von denen je eines den Bereichen Modell-Paßform, Modell-Komplexität und Modell-Schlüssigkeit angehört:

	Fehler-Rate F%	cpu-Zeit IT	Selektions-Koeffizient N_s / N_d
SNNS.RBF$_1$			
SSVM.SVM(RBF$_1$)			
SSVM.SVM$_{CG}$(RBF$_1$)			
Signifikanz-Test			

Auch wenn die Hypothesen aus 3.2.2. aufgrund der empirischen Ergebnisse modifiziert werden müssen, liefert die vorliegende Studie dennoch hinreichend Anlaß, an dem vorgelegten Konzept (2.3.*;3.3.*;4.3.*) festzuhalten.

5.3. Perspektiven

Selbst der begrenzte Rahmen der vorliegenden Untersuchung hat erkennen lassen, daß es sich bei dem Konzept der SVM um ein sehr umfassendes, mächtiges Instrument zur Lösung von Klassifikationsaufgaben handelt. Seit Schölkopfs Dissertation aus dem Jahre 1997, in der das Verfahren erstmals umfassend dargestellt worden ist, erschien zunächst eine Reihe von Artikeln und Forschungsberichten, die darauf ausgerichtet waren, das neue Instrument zu explorieren. Die Ergebnisse dieser Phase bildeten die Grundlage für Abschnitt 2.2. der vorliegenden Arbeit.

Es zeigten sich die ersten Schwächen des Verfahrens, wie sie typisch sind für universelle, vielseitige Ansätze, daß sie im Detail bisweilen Nachbesserungen bedürfen, weil sie zu teilweise unbefriedigenden Ergebnissen führen. Mehrere Forschungsergebnisse aus dieser Phase 1999-2000 haben ihren Niederschlag in den Überlegungen von 4.3. gefunden.

Seit dem vergangenen Jahr wird nunmehr verstärkt versucht, SVMs in verschiedenen mathematischen (statistischen) Paradigmen zu verankern, da die Grundidee der trennenden Hyperebene eigentlich eher symbolisch existiert und der tatsächliche Umgang mit der Entscheidungsoberfläche anderen Regularien folgt. Diese Grundlagenliteratur aus 2000-2001 ist zur Basis von 4.1. geworden.

Man kann also erwarten, daß nachfolgend vermehrt Algorithmen entwickelt oder wiederentdeckt werden, die eine Verschmelzung des grundlegenden Verfahrens SVM mit von Fall zu Fall benötigten Verfeinerungen ermöglichen. Eine Fusion von SVM-Netzen mit klassischen RBFs kann die Palette probater Werkzeuge für eine Vielzahl sich abzeichnender Anwendungsmöglichkeiten (vgl. die Diskussion um neue Identitätskontrollmerkmale in Dokumenten) zweifellos bereichern.

Literaturverzeichnis

Amari, S. und Wu, S. Improving Support Vector Machine Classifiers by Modifying Kernel Functions. Neural Networks 1999 S. 783-789 (AmaWu)

Anthony, M. Probabilistic Analysis of Learning in Neural Networks. The PAC Model and ist Variants. In: Neural Computing Surveys 1.1997 S. 1-47 (Antho)

Barlow, R.B. jr. Auge und Gehirn: Zwei-Wege-Kommunikation. In: Spektrum der Wissenschaft 6.90 S. 68-74 (Barlo)

Belhumeur, P.N. u.a. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE 19 7.1997 S. 711-720 (Belhu)

Bennett, K.P. und Demiriz, A. Semi-Supervised Support Vector Machines. In: Kearns, M.S. u.a. Hrg. Advances in Neural Information Processing Systems. MIT 1998 (BenDe)

Bergmann, L. und Schaefer, C. Lehrbuch der Experimentalphysik Bd. 1. Berlin 10.1990 (BerSc)

Bishop, C.M. Radial Basis Functions. In: Ders. Neural Networks for Pattern Recognition. Oxford 1995 S. 164-191 (Bisho)

Brause, R. Neuronale Netze. Stuttgart 1991 (Braus)

Brown, M.P.S. u.a. Knowledge-based Analysis of Microarray Gene Expression Data Using Support Vector Machines. Proc. of National Academy of Science 1997 S. 262-267 (Brown)

Burges, C.J.C. A Tutorial on Support Vector Machines for Pattern Recognition. Boston 1998 S. 1-43 (Burge)

Burges, C.J.C. und Schölkopf, B. Improving the Accuracy and Speed od Support Vector Machines. In: Neural Information Processing Systems 9.7 1997 (BurSc)

Campbell, C. Algorithmic Approaches to Training Support Vector Machines. A Survey. In: Proc. Of ESANN 2000 S. 1-8 (Campb)

Christianini, N. und Shawe-Taylor, J. An Introduction to Support Vector Machines. Cambridge 2000. (ChrSh)

Christianini, N. u.a. Dynamically Adapting Kernels in Support Vector Machines. NEUROCOLT TR 17.1998 (Chris)

Cortes, C. und Vapnik, V. Support-Vector Networks.
Boston 1995 S.275-289 (CorVa)

Craw, I. u.a. How Should We Represent Faces for Automatic Recognition? IEEE 21 8.1999 S. 725-736 (Crawl)

Crick, F. und Koch, C. Das Problem des Bewußtseins. In: Spektrum der Wissenschaft 11.92 S. 144-152 (CriKo)

Demiriz, A. und Bennett, K.P. Optimization Approaches to Semi-supervised Learning. In: Ferris, M.C. u.a. (Hrg.) Complementarity: Applications, Algorithms and Extensions. Boston 2001 S. 121-141 (DemBe)

Evgeniou, T. u.a. Image Representations for Object Detection using Kernel Classifiers. In: ACCV 2000 (Evge1)

Evgeniou, T. u.a. Regularization Networks und Support Vector Machines. In: ACM 13.2000 S. 1-50 (Evge2)

Gerthsen, C. und Vogel. H. Physik. Berlin 17.1993 (GerVo)

Girolami, M. Mercer Kernel Based Clustering in Feature Space. In: IEEE Transactions on Neural Networks 2001 (Girol)

Guttmann, C. Lehrbuch der Neuropsychologie. Stuttgart 1992 (Guttm)

Hermes, L. u.a. Support Vector Machines for Land Usage Classification in Landsat TM Imagery. IEEE 1 1999 S. 348-350 (Herme)

Hinton. G.E. Wie neuronale Netze Erfahrung lernen. In: Spektrum der Wissenschaft 11.92 S. 134-143 (Hinto)

Joachims, T. Estimating the Generalization Performance of a SVM Efficientily. In: Proc. Of Int. Conf. Machine Learning 2000 (Joach)

Kecman, V. Learning and Soft Computing, Support Vector Machines, Neural Networks und Fuzzy Logic Models. (Kecma)

Kimura, D. Weibliches und männliches Gehirn. In: Spektrum der Wissenschaft 11.92 S. 104-113 (Kimur)

Lapedes, A. und Farber, R. How Neural Nets Work. In: Anderson, D.Z. (Hrg.) Neural Information Processing Systems - Natural and Synthetic, New York 1988 S. 442-456 (LapFa)

Lee, Y.J. und Mangasarian, O.L. SSVM: A Smooth Support Vector Machine for Classification. Madison 3.1999 (LeeMa)

Lemm, J.C. u.a. A Bayesian Approach to Inverse Quantum Statistics. In: Phys. Rev. Letters 84.2000 S. 2068-2071 (Lemm J)

Lenze, B. Mathematische Aspekte neuronaler Netze. Kursskript Hagen 1996 (Lenze)

Leonhardt, H. u.a. (Hrg.) Nervensystem Sinnesorgane. Stuttgart 1987 (Leonh)

Lin, Y. u.a. Support Vector Machines for Classification in Nonstandard Situations. S. (LinYi)

Mangasarian, O.L. und Musicant, D. R. Data Discrimination via Nonlinear Generalized Support Vector Machines. Math. Programming Technical Report Madison 3.1999 (ManM1)

Mangasarian, O.L. und Musicant, D. R. Lagrangian Support Vector Machines. In: J. of Machine Learning Research 1.2001 S. 161-177 (ManM2)

Musavi, M.T. u.a. On the Training of Radial Basis Function Classifiers. Neural Networks Bd.5 1992; S. 595-603 (Musav)

Osuna, E. u.a. Improved Training Algorithm for Support Vector Machines. NNSP 1997 (Osuna)

Poggio,T. und Girosi, F. A Theory of Networks for Approximation and Learning. A.I.Memo Nr. 1140 MIT 1989 (PogG1)

Poggio, T. und Girosi, F. Networks for Approximation and Learning. Proc. of IEEE 78 9.1990 S. 1481-1497 (PogG2)

Pontil, M. und Verri, A. Support Vector Machines for 3D Object Recognition. IEEE 20 6.1998 S. 637-646 (PonVe)

Rätsch, G. LIBSVM. Ein Simulator für Support-Vektor-Maschinen. (Rätsc)

Rissanen, J. A Universal Prior for Integers and Estimation by Minimum Description Length. In: Annals of Statistics 2.1983 S. 416-431 (Rissa)

Rojas, R. Theorie der neuronalen Netze. Berlin 1993 (Rojas)

Roobaert, D. Improving the Generalisation of Linear Support Vector Machines: an Application to 3D Object Recognition with Cluttered Background. Proc. SVM Workshop at LJCAI 1999 Stockholm (Rooba)

Roobaert, D. und Van Hulle, M.M. View-based 3D Object Recognition with Support Vector Machines. Proc. IEEE 1999 (RooVa)

Rosenblatt, F. Principles of Neurodynamics. New York 1962 (Rosen)

Rüping, S. MYSVM. Ein Simulator für Support-Vektor-Maschinen. Dortmund 2001 (Rüpin)

Rumelhart, D.E. und McClelland, J.L. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Cambridge Mass. 1986 (RumMc)

Samaria, F.S. und Harter, A.C. Parametrisation of a Stochastic Model for Human Face Identification. Proc. 2. IEEE Appl. of Computer Vision 12.1994 S. 1-5 (Samar)

Saunders, C. u.a. Support Vector Machine - Reference Manual London 1998 (Saund)

Schalkoff, R. Pattern Recognition. Statistical, Structural and Neural Approaches. New York 1992 (Schal)

Schölkopf, B. Support Vector Learning. Berlin 1997 (Schö1)

Schölkopf, B. The Kernel Trick for Distances. In: TR MSR 51.2000 S. (Schö2)

Schölkopf, B. u.a. Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers. MIT 1996 (Schö3)

Schölkopf, B. u.a. Estimating the Support of a High diemsnional Distribution. TR 87.1999 (Schö4)

Schölkopf, B. u.a. Generalization Bounds via Eigenvalues of the Gram Matrix. Neuro COLT Technical Report TR London 1999 (Schö5)

Schölkopf, B. u.a. A Generalized Representer Theorem. In: NEUROCOLT 2000 Tech. Report 81 (Schö6)

Schölkopf, B. u.a. Input Space vs. Feature Space in Kernel-based Methods. IEEE Transactions on Neural Networks 10; S. 1000-1017 (Schö7)

Schölkopf, B. u.a. Kernel-dependent Support Vector Error Bounds. ICANN 1999; S. 103-108 (Schö8)

Schölkopf, B. u.a. Kernel PCA. ICANN 1997 S. 1-6 (Schö9)

Schölkopf, B. u.a. Support Vector Methods in Learning and Feature Extraction. Berlin 1997 (Schö0)

Smola, A. u.a. The Entropy Regularization Information Criterion. In: Adv. In Neural Inf. Proc. Systems 2000 (Smola)

Sollich, P. Probabilistic Interpretation and Bayesian Methods for Support Vector Machines. In: Proc. of ICANN 1999 S. 91-96 (Solli)

Steinwart, I. On the Influence of the Kernel on the Generalization Ability of Support Vector Machines. 1.2001 (Stein)

Street, W.N. und Mangasarian, O.L. Improved Generalization via Tolerant Training. J. of Optimization Theory and Applications Bd. 96 1998 S. 259-279 (StrMa)

Suykens, J.A.K. u.a. Sparse Least Squares Support Vector Machine Classifiers. In: ESANN 2000 S. 37-42 (Suyke)

Thompson, R.F. Das Gehirn. Von der Nervenzelle zur Verhaltenssteuerung. Heidelberg 1985 (Thomp)

Tipler, P.A. Physik. Heidelberg 1994 (Tiple)

Tipping, M.E. The Relevance Vector Machine. Cambridge MR 2001 S. 1-7 (Tippi)

Tong, S. und Koller, D. Restricted Bayes Optimal Classifiers. In: Proc. of AAAI 2000 (TonKo)

Tresp, V. The Generalized Bayesian Committee Machine. Proc. of 6. ACM SIGKDD 2000 S. 130-139 (Tresp)

Vanderbei, R.J. LOQO - An Interior Point Code for Quadratic Programming. *Institutspaper* (Vande)

Vidyasagar, M. A Theory of Learning and Generalization. Berlin 1997 (Vidya)

Vogt, M. Combination of Radial Basis Function Neural Networks with Optimized Learning Vector Quantization. ICNN 1993 San Francisco Bd. 3 S. 1841-1846 (VogtM)

Weston, J. und Herbrich, R. Adaptive Margin Support Vector Machines. In: Smola, A. u.a. (Hrg.) Advances in Large Margin Classifiers, Cambridge Mass. 2000 S. 281-296 (WesHe)

Williams, C.K.I. und Seeger, M. The Effect of the Input Density Distibution on Kernel-based Classifiers. In: Proc. of 17. ICML 2000 (WilSe)

Zeki, S.M. Das geistige Abbild der Welt. In: Spektrum der Wissenschaft 11.92 S. 54-63 (ZekiS)

Zell, A. Simulation Neuronaler Netze. Bonn ^{2} 1997 (Zell1)

Zell, A. u.a. SNNS Stuttgart Neural Network Simulator. User Manual 4.0. Stuttgart 1995 (Zell2)

Zhang, X. Using Class-Center Vectors to Build Support Vector Machines. NNSP 1999 (Zhang)

A N H A N G

Legende Sonderzeichen und Akronyme

CoVar	Kovarianz-Matrix
IH	Hilbertraum
K(,)	Kern-Operator
LIBSVM	SVM-Simulator von Rätsch
MDL	minimale Beschreibungslänge *minimal description length*
MYSVM	SVM-Simulator von Rüping
NN	neuronales Netz
N_D	Anzahl (Lern-) Daten
N_S	Anzahl Support-Vektoren
N_Z	Anzahl Zentroide (Stützstellen)
PCA	Hauptachsentransformation (für Eigenvektoren) *principal component analysis*
$P(x \mid a)$	bedingte Wahrscheinlichkeit
IR	Menge der reellen Zahlen
IR x ***IR***	kartesisches Produkt
$\boldsymbol{IR}^n$	n-dimensionaler euklidischer Raum
RBF	Radiale Basis-Funktion /-Funktions-Netz
SNNS	Stuttgarter Neuronale Netze Simulator
SSVM	Schölkopf-Version von SVM
SVM	Support-Vektor-Maschine
SVM(GBF)	SVM mit einem Gaußschen (RBF)-Kern
SVM_{CG} (RBF_1)	Support-Vektor-Maschine mit RBF-Variante 1, basiert auf cent_grav()-Algorithmus
dim	Dimension
ev	Eigenvektor
exp	Exponential zur Basis e
ident	Identitätsabbildung
ln	natürlicher Logarithmus
sign	Signum-Funktion
sqrt	Quadratwurzel

φ ()	Dichtefunktion
ϕ_i ()	RBF-Funktion
$\varnothing$ ()	Transformationsfunktion: $\varnothing = (\phi_1, \phi_2, \ldots \phi_k)$
X_i	i-ter Vektor
$\lvert X - X_i \rvert$	Betrag; euklidische Norm
$\lVert x \rVert$	L2-Norm
W	Normalenvektor
{a,b,c}	Menge von Elementen
$\sum_i$	Summe über i
i,,j,k=1..n	Laufindizes
$<\delta>$	Kronecker-Symbol
$[a_{ij}]$	Matrix in Komponentenschreibweise
$[a_{ij}]^T$	Matrix-Transponierte
$[a_{ij}]^{-1}$	Matrix-Inverse
w_i	Gewichtsparameter
y_i	Marke für die Klassenzugehörigkeit: $y_i \varepsilon \{-1,0,1\}$
α_i	Lagrange-Koeffizient
γ	Kern-Parameter
σ_i	Glättungsparameter
λ_i	Eigenwert
μ_i	Zentroid, RBF-Stützstelle
ξ_i	Support-Vektor
$\theta+$	(obere) Aktivationsschwelle
ψ	Eigenfunktion
AS******	Trainingsdatensatz
VO******	Testungsdatensatz
*.dat	Lerndaten-Datei aus der Eingabe von SSVM
*.cfg	Status-Datei aus der Ausgabe von SNNS
*.log	Informations-Datei aus der Ausgabe von SNNS
*.net	Struktur-Datei aus der Eingabe von SNNS
*.par	Parameter-Datei aus der Eingabe von SSVM
*.pat	Lerndaten-Datei aus der Eingabe von SNNS
[*kursiv*]	festsehender Fachbegriff oder fortlaufendes Zitat

Datensatzliste

Olivetti-Datenbank:

Person	Geschl.	Alter	Frisur	Besonderheiten
s01	m	30	licht	-
s02	m	35	reduziert	Brille
s03	m	40	licht	-
s04	m	35	reduziert	-
s05	m	25	licht	Ohr
s06	m	25	voll	Brille
s07	m	30	reduziert	Brille
s08	f	25	voll	-
s09	m	40	licht	-
s10	f	30	voll	-
s11	m	40	licht	Bart
s12	m/f [1]	25	voll	-
s13	m	40	sehr licht	Brille
s14	m	35	licht	Brille, Bart
s15	m	40	reduziert	-
s16	m	35	licht	Bart
s17	m	25	reduziert	Brille [2], Bart
s18	m	25	reduziert	-
s19	m	30	licht	Brille
s20	m	30	voll	Brille
s21	m	30	voll	-
s22	m	30	sehr licht	Farbiger
s23	m/(f)	25	voll	-
s24	m	40	reduziert	-
s25	m	40	licht	Bart
s26	m	35	reduziert	Bart
s27	m	40	sehr licht	Brille
s28	m	40	sehr licht	Brille, Bart
s29	m/(f)	40	voll	-
s30	m	25	voll	-
s31	m	35	licht	Brille, breit
s32	f/(m)	40	reduziert	-
s33	m	30	voll	-
s34	m	50	sehr licht	Brille
s35	f	35	voll	-
s36	m	30	reduziert	Brille
s37	m	40	sehr licht	Brille, Bart
s38	m	35	voll	-
s39	m/(f)	35	voll	-
s40	m/(f)	25	voll	-

[1] Es gibt mehrere Gesichter, die keine eindeutige Zuordnung m/f erlauben, auch wenn nicht in jedem Falle beide Möglichkeiten gleich wahrscheinlich wirken

[2] Es gibt Varianten mit und ohne Brille, möglicherweise weil Person Brille nicht immer trägt

Lern-Datensätze

Datensatz (0)

```
H.I Basis-Datensatz:  AS(06/30)b Training   items 0-4/5
                      VO(06/30)b Testung    items 4/5-9

Basiskonstellation:

s21 (m;30;voll;-)     vs.  s33  gleichartiges Muster
                           s10  weiblich
                           s30  jünger
                           s01  mit weniger Haaren
                           s20  mit Brille
                           s22  Farbiger
```

Datensatz (1)

```
H.I Alternativ-Datensatz: AS(06/30)a Training items 0-4/5
                          VO(06/30)a Testung items 4/5-9

Konstellation:

s19                   vs.  s11
                           s14
                           s16
                           s25
                           s26
                           s34
```

Zufalls-Satz von Personen, die in keinem anderen Datensatz Verwendung finden

Datensatz (2)

```
H.I  Klein-Datensatz: AS(06/12)  Training   items 0-5
                      VO(06/12)  Testung    items 4-9

s21                   vs.  s33
                           s30
```

Datensatz (3)

```
H.I  Klein-Datensatz: AS(07/13)  Training   items 0-5
                      VO(07/13)  Testung    items 4-9

s21                   vs.  s33
                           s30
     mit Doppelungen s21_3, s33_3  bzw. s21_7, s30_7.
```

Datensatz (4)

H.I Groß-Datensatz: AS(08/50) Training *items* 0-4
VO(08/50) Testung *items* 5-9

Erweiterung von AS(06/30) bzw. VO(06/30) um Personen, die gegenüber s21 in **zwei** Kenndaten abweichen
s04
s06
s07
s18

ferner von s21 AS(08/50) mit *items* 0-7
VO(08/50) mit *items* 2-9

Datensatz (5)

H.I reduzierter Datensatz: AS(06/30) Training *items* 0-5
VO(06/30) Testung *items* 4-9

Basiskonstellation:

s21 (m;30;voll;-)	vs.	s33	gleichartiges Muster
		s10	weiblich
		s30	jünger
		s01	mit weniger Haaren
		s20	mit Brille
		s22	Farbiger

"landmark"-Reduktion: (s. 4.1.3.) s**s**.pgm

Datensatz (6)

H.I reduzierter Datensatz: AS(06/30) Training *items* 0-5
VO(06/30) Testung *items* 4-9

Basiskonstellation:

s21 (m;30;voll;-)	vs.	s33	gleichartiges Muster
		s10	weiblich
		s30	jünger
		s01	mit weniger Haaren
		s20	mit Brille
		s22	Farbiger

"lawnmower"-Reduktion (s.4.1.3.) s**z**.pgm

Datensatz (7)

H.IIa Datensatz: AS(18/18) Training *items* 0-5
VO(18/18) Testung *items* 4-9

Basiskonstellation: m/f

s08	35	vs.	s18	25
s10	30		s30	25
s32	40		s23	25

Sonderfälle für Testung ohne y_i-*label*: s12,s29,s39,s40

Datensatz (8)

```
H.IIb   Datensatz:  AS(24/24)    Training   items  0-2
                    VO(24/24)    Testung    items  7-9

Basiskonstellation:  <=30/>30
    s05               vs.        s03
    s06                          s09
    s08                          s13
    s12                          s15
    s18                          s24
    s23                          s27
    s30                          s29
    s40                          s32
          (ohne Bart)
```

Sonderfälle für Testung ohne y_i-*label*: s17,s34

Datensatz (9)

```
H.III   Datensatz  AS(20/20)     Training   items  0-3
                   VO(20/20)     Testung    items  6-9
Originale  vs. s01 emboss-Verfremdung
               s10 emboss-Verfremdung
               s20 emboss-Verfremdung
                   [s21 spread-Verfremdung]
               s22 spread-Verfremdung
               s30 spread-Verfremdung
```

Konstellation: die unterschiedlichen Fälle aus Datensatz (1)

Datensatz (0)

```
H.I     Datensatz  AS(20/20)     Training   items  0-4
                   VO(20/20)     Testung    items  5-9
Originale      s21
               s05
               s17
               s31
Konstellation: Extremfälle der Datenbank

Verfremdungen  s21
               s05
               s17
               s31
(geometrische Verschattungen)
```

Die *items* aller Datensätze wurden in der Reihenfolge der obigen Personenliste nach + und – Etikettierung proportional wechselnd (wegen des *posh chunking*) eingegeben, also zum Beispiel für den Klein-Datensatz: s21-0,s33-0,s30-0, s21-1,s33-1,s30-1 usw.

```
jeweils:  SNNS-Format   *.dat
          SSVM-Format   *.asc
```

Simulationsprotokolle

*.cfg

```
filenames:
Layers :
Learn parameters: 5
param 1:   0.10000
param 2:   0.50000
param 3:   1.00000
param 4:   1.00000
param 5:   1.00000

Update parameters: 5
param 1:   0.00000
param 2:   0.00000
param 3:   0.00000
param 4:   0.00000
param 5:   0.00000

Initialisation parameters: 5
param 1:   0.10000
param 2:   0.10000
param 3:   0.10000
param 4:   0.10000
param 5:   0.10000

Displays: 1

display no              : 1
layers                     : 1
update type            : 2
setup show title     : 1

Printer-Configuration
```

*.log

```
current logfile:
#input  units:     10304
#output units:     1
#patterns    :       2
#subpatterns :
#links       :

learning function  : RBF-DDA
update function    : Synchonous_Order
init function      : RBF_Weights
network file       : sim1.net
learn pattern file : AS0630.pat
test  pattern file : VO0630.pat
```

*.txt

```
Copyright (c) 1998  RHBNC London - All rights reserved
written by C.Saunders, M.O. Stitson, J. Weston,
           L. Bottou, B. Schoelkopf, A. Smola
Version  SIM1

START >--------------------------------
```

```
1.Phase>----Adaption ------

BOTTOU.extern.itz=  1
BOTTOU.intern.itz=  630

Positive SVs: 1 8 15 22 29 36
Negative SVs: 2 3 4 5 6 7 9 10 11 12 13 14 16 17 18 19 20 21 23 24 25
26 27 28 30 31 32 33 34 35

Max alpha size: 1.64778
margin from hyperplane: b0.empirical_risk() is 0.647783
Zielfunktion f.objective= 9.8867
----------------------------------------
STABILISIERIERUNGS-Set:

Fehler-Anteil Nerr/Nd   F= 0 %

im einzelnen:
Total number of samples Nd=          36
Positive samples:     6
of which errors (Irrtum; ALPHA-Fehler):      0
Negative samples:     30
of which errors (Risiko; BETA-Fehler) :      0

----------------------------------------
OPTIMIERUNGS-Set:

No. 1  of SV+ is incorrectly classified.
No. 8  of SV+ is incorrectly classified.
No. 15  of SV+ is incorrectly classified.
No. 22  of SV+ is incorrectly classified.
Fehler-Anteil Nerr/Nd   F= 11 %

im einzelnen:
Total number of samples Nd=          36
Positive samples:     6
of which errors (Irrtum; ALPHA-Fehler):      4
Negative samples:     30
of which errors (Risiko; BETA-Fehler) :      0

There is 1 lagrangian multiplier per support vector.
No.     alpha(0)       Deviation
    1      1.64778     -4.66699e-07
    2     0.352217     -2.17604e-14
    3     0.327462     1.42296e-06
    4     0.352217
    5      0.32531     -6.35104e-06
    6     0.352217
    7     0.352217
    8      1.64778     -4.66699e-07
    9     0.352217
   10     0.279684     -2.0177e-06
   11     0.352217
   12     0.335803     -7.69315e-06
   13     0.352217
   14     0.352217
   15      1.64778     -4.66699e-07
   16     0.352217     -2.17604e-14
   17     0.214961     -6.79742e-07
   18     0.352217
   19     0.287853     3.94137e-07
   20     0.352217
   21     0.352217
   22      1.64778     -4.66699e-07
   23     0.352217
   24     0.230568     1.97887e-06
   25     0.352217
```

```
   26    0.287328     -3.11599e-07
   27    0.352217
   28    0.352217
   29     1.64778     -4.66699e-07
   30    0.352217
   31    0.275517     -2.65875e-06
   32    0.352217
   33    0.277876     1.91505e-06
   34    0.352217
   35    0.352217
   36     1.64778     -4.66699e-07
Finished checking support vector accuracy.

>>> save SVS.ATT
>>> >>> save AKT_SVS.ASC
There are Ns= 36 SVs
(6 positive and  30 negative)
**********
Selektions-Koeffizient Ns/Nd=  1
**********
Total   deviation(margin) is 2.82232e-05
Average deviation(margin) is 7.83978e-07
Minimum alpha     is 0.214961
Maximum alpha     is 1.64778

Alphas greater than or equal 0 is 36
Alphas greater than or equal 0 is 0
```

Details zu erwähnten Algorithmen

(1) Definition Neuronales Netz

Grundlage der Definition ist ein gerichteter Graph
$G:= (X,H, \gamma)$ mit der Inzidenzabbildung γ und mit $\delta^+(v) + \delta^-(v) > 0$ für alle Ecken $v \in X$, d.h. es soll keine Ecken geben, die nicht an mindestens eine Kante gebunden wären. Außerdem soll G schlicht sein, d.h. endlich ohne Mehrfachkanten und ohne Schlingen.

Auf dieser Grundlage werden nunmehr eine Teilmenge von Ecken $X^* \subseteq X$ und eine Teilmenge von Kanten $H^* \subseteq H$ mit folgenden Eigenschaften gebildet:

$X^* = X \setminus \{v \in X / \delta^+(v) \cdot \delta^-(v) = 0\}$
$H^* = H \setminus \{ h \in H / \gamma(h) \in (X \setminus X^*) x (X \setminus X^*)\}$
X* ist demnach die Menge aller Ecken, die nicht nur Anfangs- oder nur Endecke sind; H* ist die Menge aller zugehöriger Kanten .
Diese Mengendefinitionen ermöglichen es, die drei Standardschichten eines Netzes zu beschreiben: Eingabe-, verdeckte und Ausgabeschicht.
Bezeichnet man nunmehr die Ecken eines solchen Graphen als Knoten, so erhält man mit dem Tupel (X,X^*,H,H^*,γ) die übliche formale Definition für ein neuronales Netz NN.

Enthält das NN keine geschlossenen Pfade, so spricht man von einem *feedforward*-Netz.
nach [**Lenze**] S. NN-G 3f

Diese formale Definition sagt allein wenig über die Funktionalität eines NN aus, ermöglicht es aber, Maße für seine strukturellen Eigenschaften zu definieren, z.B. die (minimale) Beschreibungslänge.

(2) Kovarianz-Matrizen

Die Kovarianz als statistisches Streuungsmaß bildet in der Perspektive ein Gegenstück zu einem Lagemaß wie dem Konzept von Stützstellen und Support-Vektoren, mittels derer die Lerndatenmenge strukturiert und "schwerpunktmäßig" beschrieben wird. Bekanntermaßen sagen aber die Lagemaße allein i.a. nicht genug über über die Charakteristik einer Datenmenge (Musterverteilung) aus. Varianz und Kovarianz liefern dazu eine entscheidende Ergänzung.

Varianz: $\Sigma_i (X-\underline{X})^2/n$ $\quad X=(x_1,x_2, \ldots x_m)$
Kovarianz: $\Sigma_i (X-\underline{X})(Y-\underline{Y})^2/n$

```
/* Normierung der e-Funktion auf Kovarianzen */
```

```
double rad_non_sym(x1,x2,m)
{
  for(j=1;j<=m;j++)
   diff[j][1] = x1[j]-x2[j];

transpose(diff,diffT,m,1);
for (i=1;i<=m;i++)
  {
   sig[i][1] = x1[i];
   sig[i][2] = x2[i];
   }
CoVT(sig,m,2,cov,covar);
   inv_mat(covar,covI,m);

mul_mat(diffT,covI,tmp,1,m,m);

mul_mat(tmp,diff,erg,1,m,1);

result2 = erg[1][1];
return(result2);
}
```

(3) Mercer-Kerne

Das Theorem von Mercer gehört in den Kontext der Funktionalanalysis und besagt, daß ein symmetrischer, stetiger Kern k eines Integraloperators K mit

(1) $(K\, f)(y) = \int_T k(x,y)\, f(x)\, dx$

(2) $\int_{TxT} k(x,y)\, f(x)\, f(y)\, dx\, dy >= 0$

T eine kompakte Untermenge des $I\!R^n$, $f \in L^2(T)$
sich darstellen läßt als eine uniform konvergente Reihe auf T x T aus Termen von Eigenwerten λ_i und Eigenfunktionen $\phi_i(x)$ durch $k(x,y) = \sum_i \lambda_i\, \phi_i(x)\, \phi_i(y)$, $i=1..N <= \infty$.

Dieser Zusammenhang wird in der Theorie der SVMs zur Vereinfachung von Rechnungen genutzt.

Man nimmt sich Kerne vor, von denen man aus der Theorie der Integralgleichungen weiß, daß sie den Bedingungen des Mercer-Theorems genügen, und prüft, ob sie im Zusammenhang mit der Approximation von Verteilungen interpretierbar sind oder die Dichte von Verteilungen repräsentieren könnten.

Genau solche Kerne verwendet man heuristisch und explorativ, da die entsprechende Ersetzung nach dem Mercer-Theorem statthaft ist, ohne daß man der Frage nachginge, zu welcher Integralgleichungslösung K gehört. Folglich ist auch für die hier in Frage stehende Substitution nicht explizit bekannt, welches K zu welchem $\varnothing$ gehören würde. Insbesondere bleibt die Frage offen, welche Dimension der Merkmalsraum besitzt, bzgl. dessen K substituiert wird. Mittels K wird auf den Argumenten des Eingaberaumes operiert.

(4) Quadratische Optimierung und duales Problem

Man betrachte eine Funktion f, so daß
$y = f(X), \quad X = (x_1, x_2, \ldots x_m)$

Sei X_0 Stelle eines gesuchten Extremums;

unter den Nebenbedingungen

(1) $g_i(X) = 0$ für i=1...m; m<n

(2) Funktionaldeterminante $|g_{ik}(X)| = |\delta g_i(X) / \delta x_k|$
nicht sämtlich =0

Die weitere Behandlung zielt auf die bekannte notwendige Bedingung:

$(\delta / \delta x_k) \quad (f(X) + \Sigma_i \alpha_i g_i(X)) = 0 \quad k=1...n$

Daraus ergeben sich insgesamt (m+n) Gleichungen, aus denen sich m Lagrangesche Multiplikatoren α_i gewinnen lassen.

Die Bezeichnung "duales Problem" für das hier in Frage stehende Optimierungsproblem ergibt sich aus der Einsicht, daß bei einem Wechsel der Perspektive gilt: max $f(X;\alpha_i)$ = min $f(X;b,W)$ = $f(X_0)$.

(5) Schwerpunkt-Algorithmus Cent_Grav()

```
NS=n;
Nueber3=0;
for(i=1;i<=NS;i++)
   {
    for(j=i+1;j<=NS;j++)
       {
        for(k=j+1;k<=NS;k++)
          Nueber3=Nueber3+1;
        }
    }
temp=copy_mat(ksi,m,NS);

test=2;

while(test>0)
       {
        mue=matrix(1,m,1,Nueber3);

        /* Zentroide berechnen und zuweisen */
        Nueb=0;
        for(i=1;i<=Nueber3;i++)
           {
            for(j=i+1;j<=Nueber3;j++)
               {
                for(k=j+1;k<=Nueber3;k++)
                  {
```

```
                /* Bildung der Nueber3 Zentroide; l=m Merkmale */
                Nueb=Nueb+1;
                for(l=1;l<=m;l++)
                  {
                   if (test=2)
                     {
                      mue[l][Nueb]=(ksi[l][k]-(ksi[l][i]-
ksi[l][j])/2)/3;
                      test=1;
                      }
                   if (test=1)
                     {
                      mue[l][Nueb]=(temp[l][k]-(temp[l][i]-
temp[l][j])/2)/3;
                      }

                   if (l=m)
                     {
                      if (mue[l][Nueb]<0)
                        {
                         mue[l][Nueb]=-1;
                         }
                        else
                          {
                           mue[l][Nueb]=1;
                           }
                      }

                   }
                }
             }
          }

       /* Test auf Konvergenz (Abstand der mue-Vektoren=0) */
       test=0;

/* jeder mit jedem */
       for(j=1;j<=Nueber3;j++)
         {
          for(k=j+1;k<=Nueber3;k++)
            {
/* ueber alle Merkmale */
             for(l=1;l<m;l++)
               {
/* Test darf sich nicht auf Labels l=m beziehen */
                if          ((mue[l][j]-mue[l][k]>zero_t)||(mue[l][k]-
mue[l][j]))
                  test=1;
                }
             }
          }

       /* Aktualisierung der Dimension der Zentroid-Matrix */
       lf=0;
       for(i=1;i<=Nueber3;i++)
         {
          for(j=i+1;j<=Nueber3;j++)
            {
             for(k=j+1;k<=Nueber3;k++)
             lf=lf+1;
             }
          }
       Nueber3=lf;

       /* Aktualisierung Zwischenspeicher */

       temp=copy_mat(mue,m,Nueber3);
```

```
    }
return(mue);
}
```

(1) Der Algorithmus berechnet zu NS Support-Vektoren $\binom{NS}{3}$ Schwerpunkte s_l aller Dreiecke, die kombinatorisch aus den NS Eckpunkten zu bilden sind. Die s_l liegen räumlich orthogonal zur Dreiecksfläche.
(2) Der Algorithmus wird mit den s_l als neuen Support-Vektoren in einer weiteren Schleife wieder aufgenommen .
(3) Obwohl die Anzahl der s_l für $N_S > 4$ von Schleife zu Schleife (erheblich) steigt, konvergiert der Algorithmus, da der Schwerpunkt eines Dreiecks (im Gegensatz zum Höhenschnittpunkt oder Umkreismittelpunkt) stets innerhalb des betreffenden Dreiecks liegt, das Polygon mit den Eckpunkten s_l also von Schleife zu Schleife weiter implodiert.
(4) Das Verfahren liefert im physikalischen Sinne durch partielle Polygonzüge regionale, Schwerpunkte (Zentroide), da für jedes Dreieck sukzessive der Schwerpunkt bestimmt wird. Diese fallen mit ihrem jeweiligen Masseschwerpunkt zusammen, weil sich alle Orthogonalen in dem regionalen Schwerpunkt schneiden.

Bildet der Polygonzug infinitesimal umfassend den Rand des gesamten Clusters ab, so wird durch dieses Verfahren der zentrale Schwerpunkt des gesamten Clusters ermittelt.

Dem Spezialfall N_s=4 kommt in der Praxis insofern besondere Bedeutung zu, als wegen $\binom{N_s}{3}$= N_s während der Iterationen zur Konvergenz der Speicherplatzbedarf nicht steigt, was bei $N_s >> 4$ zu erheblichen technischen Problemen führen kann, die u.U. nur durch eine Lockerung der Konvergenzschranke überwunden werden können.

(6) Das BAYES-Theorem

Allgemein gilt:
$P(\boldsymbol{X}i|x)= [P(x|\boldsymbol{X}i)*P(\boldsymbol{X}i)] /[\Sigma_k P(x|\boldsymbol{X}k)*P(\boldsymbol{X}k)]$
mit der totalen Wahrscheinlichkeit $P(x)= \Sigma_k P(x|\boldsymbol{X}k)*P(\boldsymbol{X}k)$.

Verteilung - Dichtefunktion

$P(x \mid \boldsymbol{X}i) = \int p_i(x)\, dx$ Dichtefunktionbedingte Wahrscheinlichkeit für geg. x bzgl. ***X*i**
Wie groß ist die Wahrscheinlichkeit, x zu bekommen, wenn x zu ***X***i gehört ?

Als Spezialfall des BAYES-Theorems für eine Zerlegung des Ergebnisraumes Ω in genau zwei Ereignisse $\Omega = \{ \boldsymbol{X}1, \boldsymbol{X}2\}$ - also für 2-Gruppen-Klassifikationen - ergibt sich:
Ereignis $\boldsymbol{X}1$: Muster gehört zu Klasse A
Ereignis $\boldsymbol{X}2$: Muster gehört zu Klasse A^c

Ereignis x: Eingabevektor von Merkmalen (Muster)

apriori-Wahrscheinlichkeiten:
$P(\boldsymbol{X}1)$ = GesamtWahrscheinlichkeit für A
$P(\boldsymbol{X}2)$ = GesamtWahrscheinlichkeit für A^c

bedingte apriori- (Schluß-) Wahrscheinlichkeiten:
$P(x \mid \boldsymbol{X}1)$ = bedingte Wahrscheinlichkeit dafür, daß ein Muster x zu A gehört
$P(X \mid \boldsymbol{X}2)$ = bedingte Wahrscheinlichkeit dafür, daß ein Muster x zu A^c gehört

aposteriori-(Rückschluß) Wahrscheinlichkeiten
$P(\boldsymbol{X}1 \mid x)$ = Wahrscheinlichkeit für A,$\boldsymbol{X}1$ unter der Bedingung, daß x gegeben ist
$P(\boldsymbol{X}2 \mid x)$ = Wahrscheinlichkeit für A^c ,$\boldsymbol{X}2$ unter der Bedingung, daß x gegeben ist

BAYES: $P(X1|x) = [P(X1) \,.\, P(x|X1]) / [P(X1) \,.\, P(x|X1) + P(X2)\, P(x|X2)]$

für ein Ereignis x, so daß gilt: X1 trifft zu => x liegt zugrunde.

Die Zuordnung durch einen BAYESschen Schätzer erfolgt zur wahrscheinlichsten Klasse.

Daraus abeleitet kann man die spezifische, bedingte Kostenfunktion für jede Klasse $\boldsymbol{X}i$ definieren.

Der Erwartungswert für das bedingte Risiko $R(e_i \mid x)$ bei einer Entscheidung e_i zu einem Ereignis x wird danach gesetzt durch

$$R(e_i \mid x)= \Sigma_i\, P(x|\boldsymbol{X}i)*P(\boldsymbol{X}i \mid x) .$$

(7) Eigenwert, Eigenfunktion, PCA

Die (K)PCA ist ein Verfahren, mit Hilfe einer Basistransformation diejenigen Vektoren, die die maximale Varianz tragen, im Feld nach vorn zu rücken. Es handelt sich also (im Rahmen von überwachtem Lernen) um eine spezifizierte Varianzanalyse.

Erinnert sei in diesem Zusammenhang an die prägende Eigenschaft der Eigenvektoren, unter jeder linearen Abbildung ihre Richtung beizubehalten; d.h. solche Abbildungen entsprechen dann der komponentenweisen Multiplikation mit einem Skalar.

Im Falle einer hohen Anzahl m von Komponenten benötigt man in der Praxis eine Problemtransformation für das in 4.2.2. rekapitulierte Basisverfahren.

Grundlage:

Matrix mit n Trainingsdaten $D:= (\boldsymbol{v}_1, \boldsymbol{v}_2, ... \boldsymbol{v}_n)$;
D besitzt m*n Positionen
$\boldsymbol{v}_i$ besitzt m= 10304 Komponenten; n << m

Komponenten-Mittelwerte über die Trainingsdaten $\boldsymbol{m}_V := 1/n \ \Sigma_i \ \boldsymbol{v}_i$
1 <= i <= n
$\boldsymbol{m}_V$ besitzt m = 10304 Komponenten

Abweichungs-Vektor: $d_1 := \boldsymbol{v}_i - \boldsymbol{m}_i \quad 1 <= i <= n$
d_1 besitzt m= 10304 Komponenten

$\boldsymbol{D}_V := (d_1, d_2, ...d_N)$ Matrix der N Abweichungsvektoren
$\boldsymbol{D}_V$ besitzt m*n Positionen

Dann gilt nach Lemma: $CoVar_V = 1/n \ \boldsymbol{D}_V \boldsymbol{D}_V^T$
$CoVar_V$ besitzt aus dem Skalarprodukt m*m Positionen

Daraus ergibt sich, daß $Co_V := 1/n \ \boldsymbol{D}_V^T \boldsymbol{D}_V$
aus dem Skalarprodukt gerade n*n Positionen besitzt

Vorgehensweise:
(0) Eigene Prozedur zur Bestimmung der d_i.

(1) Bestimme mittels *ALGEBRA*-Routine **matmult** Co_V mit der Dimension n*n.

(2) Bestimme mittels *ALGEBRA*-Routine **eigen** Eigenwerte λ_j und Eigenvektoren $\boldsymbol{l}_j$, so daß in der üblichen Weise gilt: $\lambda_j \ \boldsymbol{l}_j = Co_V \ \boldsymbol{l}_j$.
Davon gibt es bis zu n Exemplare mit je n Komponenten.

(3) Dann gilt durch Links-Multiplikation mit $\boldsymbol{D}_V$ unter Kommutativität der Skalare:

$$\lambda_j \boldsymbol{D}_V \boldsymbol{l}_j = \boldsymbol{D}_V Co_V \ \boldsymbol{l}_j = 1/n \ \boldsymbol{D}_V \boldsymbol{D}_V^T \boldsymbol{D}_V \ \boldsymbol{l}_j$$

bzw. in neuer Lesart mit Assoziativgesetz :

$$\lambda_j (\boldsymbol{D}_V \boldsymbol{l}_j) = \qquad = [1/n \ \boldsymbol{D}_V \boldsymbol{D}_V^T] (\boldsymbol{D}_V \boldsymbol{l}_j)$$

Folglich sind in dieser Interpretation die $(\boldsymbol{D}_V \ \boldsymbol{l}_j)$ Eigenvektoren von $[1/n \ \boldsymbol{D}_V \boldsymbol{D}_V^T] = CoVar_V$ mit den zugehörigen Eigenwerten λ_j.

d.h. benötigt wird noch $\boldsymbol{D}_V \boldsymbol{l}_j$, erhältlich mit **matmult.**
Davon gibt es folglich ebenfalls bis zu n Exemplare, nunmehr mit m Komponenten. λ_j bleiben erhalten.