Signalraumdetektion und ihre Anwendungen

Ingo Dahm

Institut für Roboterforschung
Abteilung Informationssysteme
Universität Dortmund
Otto-Hahn-Strasse 4
44221 Dortmund

15.10.2005

Für meine Familie.

Inhaltsverzeichnis

Kapitel 1

Klassifizierung

1.1 Grundlagen

Als Klassifizierung oder Klassifikation bezeichnet man einen Vorgang oder eine Methode zur Einteilung von Objekten in so genannte Klassen. Dabei kann eine Klasse aus keinem, einem oder mehreren Elementen (den Objekten) bestehen. Objekte können beliebige Gegenstände oder Sachverhalte sein, die sich nach irgendeiner Art unterscheiden lassen, z.B. von biologischen oder physikalischen Werten [VR05]. Dabei ist die Klassenbildung nicht nur seit langem eine grundlegende wissenschaftliche Methodik: Lebewesen klassifizieren ständig, damit sie ihre Umwelt besser interpretieren und rascher auf sie reagieren können. Vielfach bedeutet Klassifikation auch besseres Verstehen und Erkennen von Zusammenhängen, indem relevante von irrelevanter Information getrennt wird.

Darüber hinaus ist die Klassenbildung in vielen Lebensbereichen Grundlage des menschlichen Zusammenlebens. Wahrgenommene Töne klassifizieren wir beim Hören zu Worten, gelesene Symbole ordnen wir Buchstaben zu, die wir wiederum zu Worten formen; auch die Zuordnung von Worten zu ihrer Bedeutung ist ein Klassifikationsprozess. Viele diese Verfahren laufen eher unbewusst ab, in gewisser Weise „automatisch“.

Die automatische Klassifikation steht im Gegensatz zur manuellen Klassifizierung, bei der die zu klassifizierenden Objekte von Menschen manuell den entsprechenden Klassen zugeordnet werden. In technischen Prozessen soll durch eine maschinelle automatische Klassifikation eine Einteilung von Objekten in Klassen ohne menschliche Hilfe erreicht werden. Prinzipiell kann hier die automatische Klassifizierung genauer und schneller als die manuelle Klassifizierung erfolgen. Somit ist die automatische Klassifizierung ein fundamentales Verfahren für eine Reihe von technischen Systemen und Anwendungen. Ein Beispiel ist das Sortieren von Werkstücken durch Maschinen in einer Produktionsanlage: Anhand visueller Merkmale, die durch Menschen nur schlecht zu erkennen sind, sortieren

Roboter und Automaten die produzierten Werkstücke und ordnen sie bestimmten Qualitätsklassen zu.

Typische von Menschen nur schwer unterscheidbare Objekte sind Türschlüssel (siehe Abb. 1.1). Diese sehen sich oft sehr ähnlich, obwohl sie bei genauerer Betrachtung eindeutig aufgrund ihrer Form voneinander zu unterscheiden sind. Somit kann nicht nur die Klassifizierung, sondern bereits die Extraktion eindeutiger Merkmale problematisch sein. Die Klassifizierung hat folglich zwei Problemkreise: Zum einen müssen relevante Merkmale bestimmt, zum anderen muss aus diesen Merkmalen eine Klasse abgeleitet werden. In der Praxis sind diese Herausforderungen häufig noch mit der Fragestellung verkoppelt, wie man die Unterscheidungsmerkmale besonders schnell, sicher und gefahrlos feststellen bzw. messen kann. So behilft man sich häufig mit eindeutigen (farbigen) Markierungen, z.B. bei Schlüsseln in Form von Anhängern, wie in Abb. 1.1 links oben illustriert.

Abbildung 1.1: Gruppe von unterschiedlicher Schlüssel.

Liegen *alle* Informationen über ein Objekt vor, so kann es immer korrekt klassifiziert werden. Wie am Beispiel der Schlüssel jedoch deutlich wird, kann es mühsam sein, alle relevanten Eigenschaften eines Objektes zu bestimmen. Somit muss die Klassifizierung häufig mit unvollständigen oder auch nicht korrekten Aussagen über ein Objekt funktionieren.

In einigen Fällen wird die Klassifizierung dafür benutzt, um auf unbekannte Eigenschaften zu schließen. Typische Fragestellungen aus der Natur sind zum Beispiel: „Ist die Pflanze giftig?“ oder „Ist dieser Vierbeiner ein Raubtier?“. Hier kann eine fehlerhafte Klassifizierung gefährliche, ja sogar tödliche Folgen haben.

Wenn für die Klassifizierung weitere Einschränkungen gelten, z.B. ein Zeitlimit, so ist es unter Umständen gar nicht möglich, alle prinzipiell relevanten Eigenschaften zu bestimmen. Dem Problem der beschränkten Zeit zur Merkmalsbestimmung und Klassifizierung kann man auf verschiedene Weise begegnen. Eine Variante ist es, sich auf wenig einfach bestimmbare Eigenschaften des Objektes zu beschränken („Ist eine Tomate rot, dann ist sie reif.“ – siehe Abb. 1.2 links). Ein anderes Verfahren ist es, Informationen zusammen zu fassen und eine Klassifizierung durchzuführen, die auf diesen abstrakteren Hilfsinformationen basiert. Beispielsweise ist die Einteilung in interessante und uninteressante Bücher schwierig, ohne diese komplett zu lesen. Eine Entscheidung lässt sich jedoch durchaus treffen, wenn der Inhalt zu abstrakteren Eigenschaften des Buches, wie Thematik, Name des Autors, Inhaltsangabe, Gliederung usw., zusammengefasst wird. Basierend auf Vorwissen um ähnliche Bücher kann dann eine Klassifizierung vorgenommen werden (z.B. „Thriller von Stephen King sind interessant.“).

Durch den Verzicht auf potentiell relevante Informationen und durch Messfehler bei der Bestimmung der Eigenschaften eines Objektes kann es bei der Klassifizierung zu Fehlern kommen. Daher ist die Fehlerrate der Klassifikation, d.i. das Verhältnis der Anzahl von fehlerhaft klassifizierten Objekten zu allen klassifizierten Objekten, ein wesentliches Gütemaß des eingesetzten Verfahrens.

Abbildung 1.2: Grüne, unreife Tomaten und rote, reife Tomate (links) sowie Tomate mit unklarem Reifegrad (rechts).

Potentielle Fehler können in jedem Fall vor der mit der Klassifizierung verbundenen Komplexitätsreduktion besser erkannt werden als hinterher. Insbesondere führt eine binäre Ja-Nein-Entscheidung („X ist ein Y“ bzw. „X ist *kein* Y“) zu einem Informationsverlust, der keinen Rückschluss auf die Korrektheit der

Entscheidung zulässt. Daher geben moderne Klassifizierer zusätzlich zur Aussage über die Klasse einen Wert aus, der die Zuverlässigkeit der getroffenen Entscheidung beschreiben soll. Dieses Maß wird gemeinhin Zuverlässigkeitsinformation genannt. Eine große rote Tomate würde als „reif“ mit hoher Zuverlässigkeit klassifiziert werden (Abb. 1.2, links), eine mittelgroße rote Tomate mit einigen grünen Stellen ebenfalls als „reif“, jedoch mit niedrigerer Zuverlässigkeit (Abb. 1.2, rechts). Die Angabe der Zuverlässigkeit einer Entscheidung bietet Vorteile bei der auf die Klassifizierung folgenden (Daten-)Verarbeitung: Ein unsicher als „essbar“ erkannter Pilz wird nicht gegessen, ein sicher als „essbar“ erkannter hingegen schon.

Die Extraktion von sinnvoll verwertbarer Zuverlässigkeitsinformation ist neben der Korrektheit der durch den Klassifizierer getroffenen Entscheidung bzw. einer niedrigen Fehlerrate eine besonders wichtige Eigenschaft. Weitere Randbedingungen sind die bereits erwähnte Geschwindigkeit des Verfahrens, algorithmische Komplexität, die Kosten des Klassifizierers und dessen Fähigkeit zum Umgang mit sehr komplexen, hochdimensionalen Eingangsdaten.

Eine Rahmenbedingung, die die automatische Klassifizierung besonders erschwert, ist dann gegeben, wenn sich Klassen, Eigenschaften oder Entscheidungsgrenzen[1] mit der Zeit verändern. Während beispielsweise ein Hochdruckgebiet im Sommer „schönes Wetter“ beschert, ist es im Winter das Tiefdruckgebiet, welches Schnee und somit „schönes Wetter“ mit sich bringt. Derartigen wechselnden Randbedingungen wird durch den Einsatz von lernfähigen Klassifikationsverfahren Rechnung getragen.

1.2 Klassifizierungsverfahren

Je nach Aufgabenstellung sind die einzelnen beschriebenen Randbedingungen mehr oder weniger bedeutsam. Infolgedessen gibt es eine Vielzahl von spezialisierten Klassifikationsverfahren für unterschiedliche Anwendungsfälle. Im Folgenden werden exemplarisch einige Verfahren vorgestellt. Dabei wird zwischen primitiven und komplexen Klassifizierern unterschieden, wobei letztere aus primitiven Komponenten aufgebaut sind.

1.2.1 Primitive Klassifizierer

Die hier vorgestellten Klassifizierer sind alle nach dem gleichen Schema aufgebaut, das in Abbildung 1.3 dargestellt ist: Ein Objekt s^i, das durch M Merkmale gegeben ist, wird durch den Merkmalsvektor $\vec{s}^i$ charakterisiert. Dieser liegt am Eingang des Klassifizierers an. Der Klassifizierer ist gekennzeichnet durch eine

[1] Eine Entscheidungsgrenze gibt an, wie stark eine Eigenschaft ausgeprägt sein muss, damit sie die Entscheidung beeinflusst

Reihe interner Parameter w_j, die als Parametervektor $\vec{w}$ zusammengefasst werden.

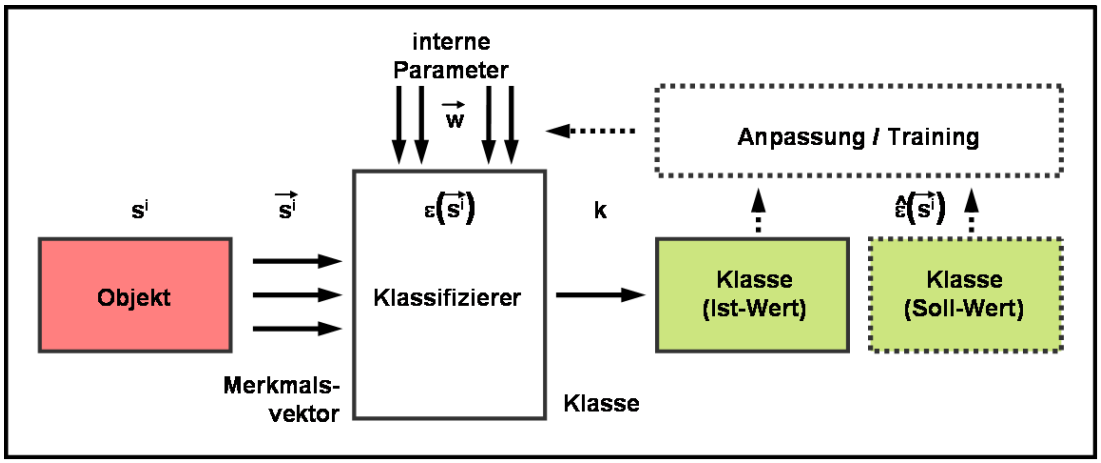

Abbildung 1.3: Blockschaltbild eines Klassifizierers. Punktierte Bereiche sind optional.

Auf verschiedene Weise bestimmen die Klassifizierer aus $\vec{s}^i$ und $\vec{w}$ das Ergebnis $\varepsilon(\vec{s}^i)$ der Klassifizierung: die Klasse k. Dieses Ergebnis wird auch Ist-Wert genannt. Der Ist-Wert steht im Gegensatz zum Soll-Wert, welcher als Orakel das korrekte Ergebnis der Klassifikation $\hat{\varepsilon}(\vec{s}^i)$ liefert.

Basierend auf Soll- und Ist-Wert, können durch ein geeignetes Lernverfahren die internen Parameter w_j des Klassifizierers zur Verringerung des Klassifikationsfehlers angepasst werden. Diese Adaption ist typischerweise zeitlich begrenzt und wird nach Erreichen eines festgelegten Kriteriums beendet, d.h. der punktiert dargestellte Zweig in Abbildung 1.3 wird dann abgeschaltet.

1.2.1.1 Das Adaptive Lineare Element

Das so genannte Adaptive Lineare Element (Adaline) ist ein einfacher Klassifizierer der vollständig aus linearen Komponenten aufgebaut ist [Wid62]. Die Eingangsgrößen s_0 bis $s_{(M-1)}$ werden mit Konstantenmultiplizierern w_0 bis $w_{(M-1)}$ gewichtet. Die Produkte werden schließlich zuzüglich einer weiteren Konstante w_M aufsummiert. Diese Summe wird als ν bezeichnet. Ziel ist es, dass für eine bestimmte Eingangssignal-Konstellation am Ausgang des Adaline ein Soll-Wert $\hat{\nu}$ anliegt.

$$\nu = \begin{pmatrix} s_0 \\ \vdots \\ s_{(M-1)} \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ \vdots \\ w_{(M-1)} \end{pmatrix}^T + w_M \tag{1.1}$$

Für Übereinstimmung von Soll- und Istwert ist die Wahl geeigneter Koeffizienten w_0 bis w_M essentiell. Diese Koeffizienten werden auch Gewichte genannt. Die Anpassung der zunächst zufällig gewählten Gewichte kann sowohl im kontinuierlichen als auch diskreten Zeit- und Wertebereich erfolgen. Zur Vereinfachung sei im Folgenden eine zeitdiskrete Adaption angenommen. ξ beschreibe hierzu eine Quantisierungsfunktion, die den Werte-kontinuierlichen Bereich auf einen Werte-diskreten Bereich abbildet. Für Wertekontinuität gelte $\xi : x \rightarrow x$. Für alle weiteren Ausführungen gelte: Ein Parameter x habe zum Zeitpunkt t den Wert $\langle x \rangle_t$. Der auf t folgende Zeitpunkt sei $t + \Delta t$. Als verkürzte Schreibweise für $t + x\Delta t$ gelte $t + x$.

Für das Adaline-Neuron gilt nun die in Gleichung 1.2 beschriebene Regel zum Anpassen der Gewichte. Diese Vorschrift wird auch Lernregel genannt [Wid62].

$$\langle w_i \rangle_{(t+1)} = \xi \left(\langle w_i \rangle_t + \langle \Lambda \rangle_t \cdot (\hat{\nu}(\langle \vec{s} \rangle_t) - \langle \nu(\langle \vec{s} \rangle_t) \rangle_t) \cdot \langle s_i \rangle_t \right) \tag{1.2}$$

Hierbei ist $\langle \Lambda \rangle_t$ ein Korrekturfaktor mit $0 < \langle \Lambda \rangle_t \leq 1$. Falls die Differenz zwischen Soll- und Istwert ungleich Null ist, werden die Koeffizienten w_i derart angepasst, dass der Fehlerwert $(\hat{\nu}(\vec{s}) - \langle \nu(\vec{s}) \rangle_t)$ verringert wird.

Wesentlich für die Lernfunktion ist die Wahl von $\langle \Lambda \rangle_t$. Während die Wahl einer Konstante Vorteile in Form einer einfachen Implementierung birgt, bewirkt eine Wahl gemäß Gleichung 1.3 einen besonders schnell konvergierenden Lernprozess [Wid62].

$$\langle \Lambda \rangle_t = \left(\sum_{j=1}^{t} \sqrt{\sum_{i=0}^{M-1} (\langle s_i \rangle_j)^2} \right) \tag{1.3}$$

Adaline-Neuronen weisen aufgrund ihrer Linearität Vorteile bei der Implementierung auf [Cal99]. Andererseits stehen diesem Vorteil die nicht beschränkten Werte- und Definitionsbereiche der Koeffizienten gegenüber, was die Implementierung erschweren kann. Diese Problematik wird durch das so genannte künstliche Perzeptron gelöst.

1.2.1.2 Das Perzeptron

Ein Perzeptron ist ebenfalls ein sehr einfacher Klassifizierer, der auf dem Prinzip der Nervenzelle, dem Neuron, basiert. Definitionsgemäß ist ein künstliches Perzeptron in drei Schichten aufgebaut, die auch Layer[2] genannt werden [And95]: **S**ensor-Schicht, **A**daptions-Schicht und **R**esultat-Schicht. Die S-Schicht beinhaltet die Werte der Eingangsdaten s_0 bis $s_{(M-1)}$. Im A-Layer werden die Sensor-Daten gewichtet, und in der dritten Schicht, dem R-Layer, wird das Ergebnis des Perzeptrons ausgegeben. Der strukturelle Aufbau eines derartigen Systems ist in Abbildung 1.4 wiedergegeben:

[2]Layer: engl. „Schicht“

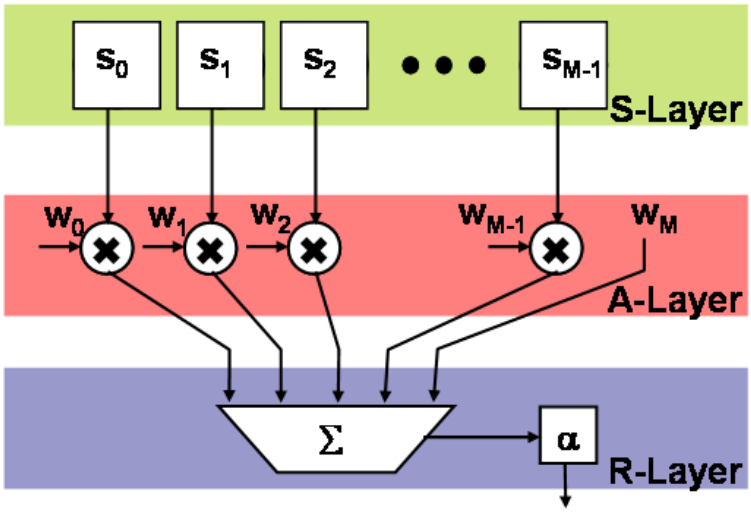

Abbildung 1.4: Blockschaltbild eines Perzeptrons mit S-, A- und R-Layer.

Der grundlegende Unterschied zwischen Adaline-Neuron und Perzeptron besteht darin, dass die Ausgabe des Perzeptrons auf den Wertebereich $[0, 1]$ beschränkt wird [Ros61]. Somit ist das Perzeptron prinzipiell als Erweiterung des Adaline-Neurons zu betrachten, wobei dessen Ausgang ν mit einer so genannten Aktivierungsfunktion α gemäß Gleichungen 1.4-1.5 zum Ausgang π konvertiert wird[3].

$$\pi = \alpha(\nu) = \alpha\left(\sum_{i=0}^{M-1}(s_i \cdot w_i) + w_M\right) \tag{1.4}$$

$$\pi = \alpha\left(\begin{pmatrix} s_0 \\ \vdots \\ s_{(M-1)} \\ 1 \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ \vdots \\ w_{(M-1)} \\ w_M \end{pmatrix}^T\right) \tag{1.5}$$

Durch die Beschränkung des Wertebereichs ergeben sich eine Reihe von Vorteilen bei der Software- und Schaltungsimplementierung im Vergleich zum Adaline-Neuron [Wid62, Zel94]. Andererseits ist die von *Frank Rosenblatt* ursprünglich geforderte Diskretisierung des Wertebereichs auf $\{0, 1\}$ gelegentlich nachteilig, was durch eine entsprechende Anpassung der Aktivierungsfunktion verändert werden kann [Ros61]. Häufig wird zu diesem Zweck eine sigmoide Aktivierungsfunktion gemäß Gleichung 1.6 verwendet, die sowohl die Forderungen nach Beschränkung als auch Kontinuität des Wertebereiches erfüllt und darüber hinaus umkehrbar eindeutig ist [Hay94].

[3]In der Literatur wird α häufig auch mit A bezeichnet

$$\alpha : x \rightarrow \frac{1}{1 + e^{\Psi_0(\Psi_1 - x)}} \tag{1.6}$$

Die Adaption eines künstlichen Perzeptrons erfolgt in Analogie zur Adaline-Lernregel gemäß Gleichung 1.2, wobei der zeitvariable Korrekturfaktor durch eine Konstante, die Lernrate Λ, ersetzt wird:

$$\langle w_i \rangle_{(t+1)} = \xi \left(\langle w_i \rangle_t + \Lambda \cdot (\alpha(\hat{\pi}(\langle \vec{s} \rangle_t) - \langle \pi(\langle \vec{s} \rangle_t) \rangle_t) \cdot \langle s_i \rangle_t \right) \tag{1.7}$$

Diese Lernregel ist in der Literatur als „Λ-Reinforcement-Rule“ und auch als „Perzeptronen-Lernregel“ bekannt.

1.2.1.3 Das Kernel-Perzeptron

Eine gängige Erweiterung des Perzeptrons stellt das so genannte Kernel-Perzeptron dar, bei dem die Eingangsdaten einer Transformationsvorschrift ϕ gemäß Gl. 1.8 unterzogen werden, die auch Kernel-Funktion genannt wird [CST04]. Ziel dieser Transformation ist die Vereinfachung der zu klassifizierenden Merkmalsvektoren. Dies kann zum Beispiel dadurch geschehen, dass das Vorwissen über deren Lage im Merkmalsraum in die Transformationsvorschrift integriert wird.

$$\phi(\vec{s}) = \left(\phi_0(\vec{s}), \phi_1(\vec{s}), ...\phi_{(M-1)}(\vec{s}) \right) \tag{1.8}$$

Ein Beispiel für den sinnvollen Einsatz eines Kernel-Perzeptrons ist in Abbildung 1.5 illustriert. Aufgrund ihrer Eigenschaften s_0^j und s_1^j sind die Objekte s^j in „Rot“ und „Grün“ zu klassifizieren (linker Teil der Abbildung). Durch Anwendung der Transformationsvorschriften 1.9 und 1.10 entstehen aus den alten Merkmalen neue abstrakte Eigenschaften, die die Klassifizierung vereinfachen (rechter Teil von Abb. 1.5).

$$\phi_0(\vec{s}) = \sqrt{(s_0)^2 + (s_1)^2} \tag{1.9}$$

$$\phi_1(\vec{s}) = tan^{-1} \left(\frac{s_0}{s_1} \right) \tag{1.10}$$

Der transformierte Eingangsvektor $\phi(\vec{s})$ muss dabei nicht zwingend die gleiche Dimension haben wie der ursprüngliche Merkmalsvektor $\vec{s}$. Gegebenenfalls könnte im obigen Beispiel der transformierte Raum auch eindimensional sein, indem auf die für die Klassifizierung redundante abstrakte Eigenschaft $\phi_1(\vec{s})$ verzichtet wird.

Somit ist der Ausgang eines allgemeinen Kernel-Perzeptrons, wie in Gleichung 1.11 angegeben, bestimmt.

$$\kappa = \alpha(\nu) = \alpha \left(\sum_{i=0}^{M-1} (\phi_i(\vec{s}) \cdot w_i) + w_M \right) \tag{1.11}$$

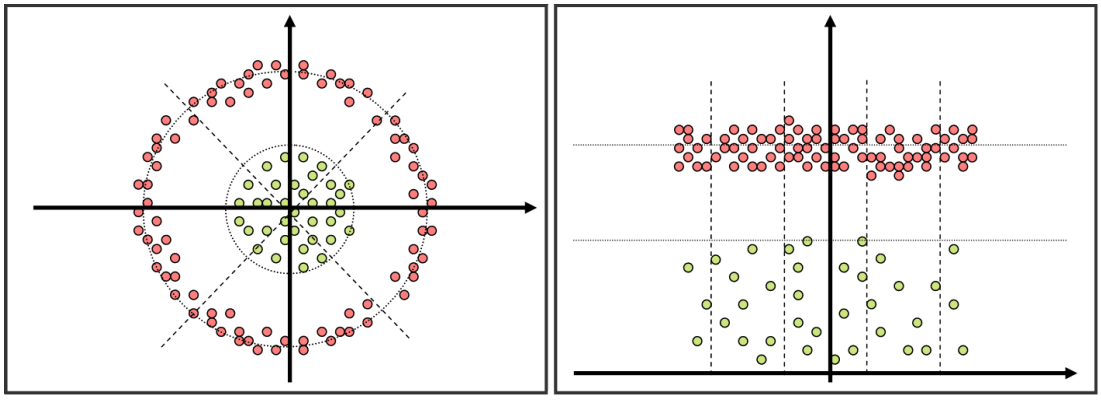

Abbildung 1.5: Beispiel für eine sinnvolle Transformation mittels Kernel-Perzeptron. Links: Ursprünglicher Merkmalsraum. Rechts: Transformierter Raum.

Sowohl Adaline-Neuronen als auch Perzeptronen haben eine besondere Bedeutung, wenn man sie nicht allein betrachtet, sondern im Verbund, dem so genannten neuronalen Netz.

1.2.2 Neuronale Netze

In einem neuronalen Netz werden die Ausgänge eines oder mehrerer adaptiver Primitive auf den Eingang eines oder mehrerer weiterer Primitive gelegt. Hierdurch ergibt sich eine hierarchische, netzartige Struktur, die als Hilfmittel zur Lösung komplexer Probleme der Klassifizierung und Mustererkennung geeignet ist [Zel94, Hay94].

Stellvertretend für die Vielzahl von Architekturen künstlicher neuronaler Netze ist in Abbildung 1.6 ein so genanntes Multi-Layer-Perzeptronen-Netzwerk dargestellt. Das Netzwerk wird durch die Eingangsschicht (oben) mit einem dreidimensionalen Eingangsvektor gespeist.

Das Eingangssignal klassifizieren fünf Perzeptronen in der Eingangsschicht, ein Perzeptron in der Ausgabeschicht und sieben Perzeptronen in einer „versteckten“ Schicht, dem „hidden layer“.

Während einschichtige Netze, resp. Perzeptronen und Adaline-Neuronen, ausschließlich linear trennbare Strukturen [NAMM93] klassifizieren können, ist es durch geeignete Wahl der Gewichte bzw. eine geeignete Lernregel in neuronalen Netzen darüber hinaus möglich, auch nichtlineare Funktionen zu approximieren [Hay94].

1.2.2.1 Selbstorganisierende Karten

Selbstorganisierende Karten setzen sich aus Eingabe- und Ausgabeneuronen zusammen, die jeweils eine Schicht bilden. Die Ausgabeneuronen werden auch als

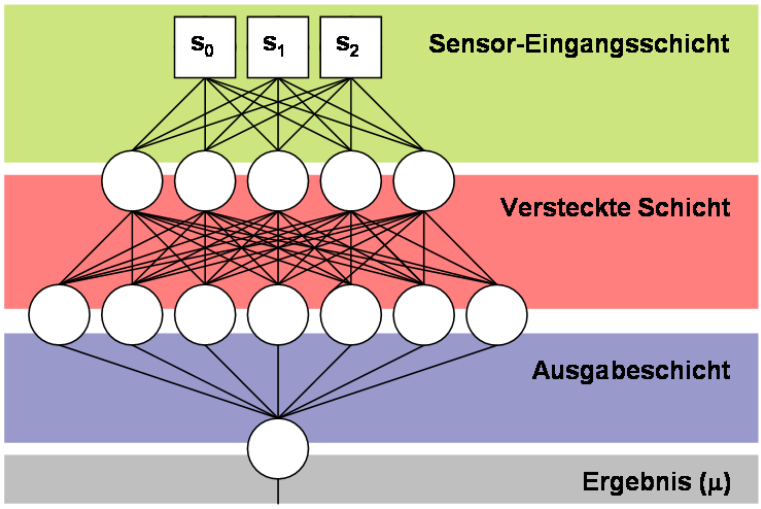

Abbildung 1.6: Prinzipskizze eines dreilagigen Multi-Layer-Perzeptronen-Netzwerkes mit einer zusätzlichen Sensor-Eingangsschicht (oben).

Kohonen-Neuronen bezeichnet. Die Eingabeneuronen sind über Gewichtsvektoren mit den Kohonen-Neuronen verbunden. Die Eingabeschicht besteht aus der Menge der Eingabevektoren $\vec{s} = (s_0, s_1, ...s_{M-1})^T$, wobei M wiederum die Anzahl der Eingabevektoren ist.

Die Kohonen-Neuronen werden als Gitter dargestellt, wobei die Verbindungen zwischen den Ausgabeneuronen als „Nachbarschaftsbeziehung“ zu verstehen sind. Wenn man den Wert eines Neurons verändert, beeinflusst das alle umliegenden (benachbarten) Knoten. Je nach Nachbarschaftsbeziehung ist diese Veränderung mehr oder weniger stark.

Dimension und Form der Ausgabeschicht werden einmalig festgelegt und später nicht mehr verändert. Beide Schichten sind durch Gewichtsvektoren $\vec{w}^j = (w_0^j, w_1^j, ..., w_{(M-1)}^j)$ verbunden, wobei j der Knoten aus der Kohonenschicht ist, mit dem der betreffende Gewichtsvektor verbunden ist.

Selbstorganisierende Karten lernen dadurch, dass bei einer Eingabe ein so genanntes Siegerneuron $\hat{j}$ bestimmt wird. Dies geschieht mittels Abstandsbestimmung von Merkmals- und Gewichtsvektor: das Neuron mit dem geringsten Abstand zum Eingabevektor gewinnt! Als Abstandsmaß wird die euklidische Norm nach Gleichung 1.12 verwendet.

$$||\vec{s} - \vec{w}^j|| = \sqrt{\sum_{i=0}^{(M-1)} (s_i - w_i^j)^2} \tag{1.12}$$

Besitzen zwei oder mehr Neuronen die gleiche euklidische Norm, so wird unter ihnen zufällig ein Sieger bestimmt. Schließlich wird das Gewicht des Sieger-

Neurons verändert. Zusätzlich werden ebenfalls die Gewichte benachbarter Neuronen angepasst. Dabei kommt es nicht darauf an, dass die Neuronen in der Nachbarschaft der Eingabe ähnlich sind. Es zählt ausschließlich die „geographische“ Entfernung zu dem als ähnlich befundenen Neuron. Die Stärke der Veränderung der Gewichte wird über die folgende Trainingsfunktion bestimmt:

$$\langle w^j \rangle_{(t+1)} = \langle w^j \rangle_t + \Lambda(t) \cdot \langle \theta(\hat{j}, j) \rangle_t \cdot \sqrt{\sum_{i=0}^{M-1} (s_i - \langle w_i^j \rangle_t)^2} \tag{1.13}$$

Die Trainingsfunktion richtet sich somit zum einen nach der Nachbarschaftsfunktion $\theta(i, j)$, bezogen auf das Siegerneuron $\hat{j}$, und zum anderen nach einer zeitabhängigen Funktion $\Lambda(t)$, die die bereits im Abschnitt 1.2.1 erläuterte Lernrate reguliert.

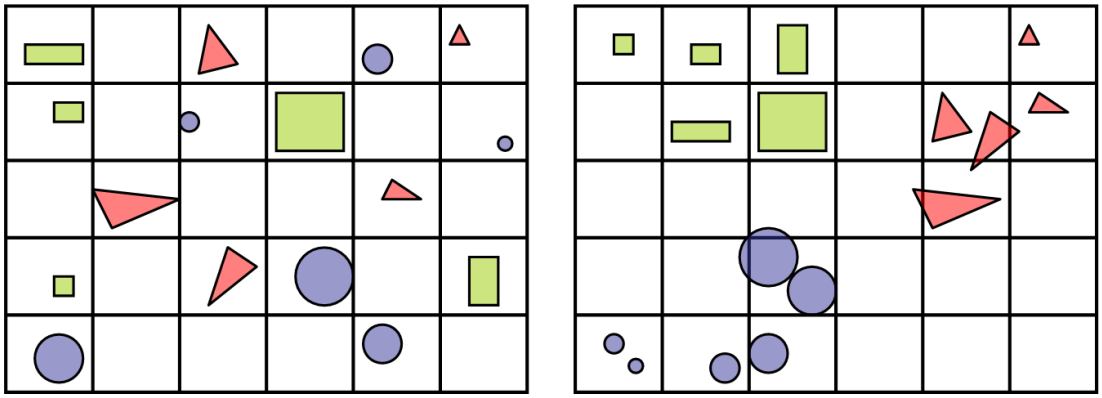

Abbildung 1.7: Struktur einer selbstorganisierenden Karte vor (links) und nach (rechts) Abschluss der Trainingsphase [GASRG98].

Das wichtigste Merkmal der selbstorganisierenden Karte ist die Fähigkeit des so genannten unüberwachten Lernens. Diese Eigenschaft wird in Abbildung 1.7 verdeutlicht [GASRG98]. Dort sind die Positionen der Siegerneuronen bei Eingabe von 16 verschiedenen geometrischen Figuren vor und nach der Trainingsphase dargestellt. Es ist ersichtlich, dass nach abgeschlossener Trainigsphase ähnliche Objekte dicht beeinander liegen.

Somit gestattet die Struktur selbstorganisierender Karten ohne manuelle Eingriffe Ähnlichkeiten zwischen Objekten zu extrahieren und durch ihre Nachbarschaft zu visualisieren. Sind jedoch die wichtigen Merkmale der zu klassifizierenden Objekte bereits vor der Trainingsphase bekannt, so ist es nicht möglich, dieses Wissen in die selbstorganisierende Karte zu integrieren.

1.2.2.2 Mehrschichtige Perzeptronen-Netze

Im Gegensatz zur selbstorganisierenden Karte, ist die typische Anwendung eines mehrschichtigen Perzeptronennetzwerkes (MLP) die Approximation einer unbekannten Abbildungsvorschrift, von der einige Argumente $\vec{s}^i$ und die zugehörigen Funktionswerte $\hat{\mu}(\vec{s}^i)$ bekannt sind. Ziel ist es, dass das MLP die Abbildungsvorschrift anhand der exemplarischen Werte erlernt und dadurch in der Lage ist, den Funktionsverlauf zu generalisieren, so dass auch ein nicht trainiertes $\vec{s}^j$ auf ein sinnvolles $\mu(\vec{s}^j)$ abgebildet wird.

Alternativ kann das MLP auch zur Klassifizierung eingesetzt werden. In diesem Fall ist die zu erlernende Abbildung eine Klassifizierungsregel, um $\vec{s}^i$ auf eine vorgegebene Klasse $\mu : \vec{s} \rightarrow k$ abzubilden.

Das MLP besteht aus miteinander vernetzten Perzeptronen oder Adaline-Neuronen. Zur Vereinfachung sei im Folgenden Zyklenfreiheit angenommen: Kein Ausgang eines Perzeptrons darf direkt oder indirekt mit sich selbst verbunden sein. Die Gewichte aller Neuronen werden zunächst mit Zufallswerten besetzt. Wird ein Merkmalsvektor an der Eingangsschicht des MLP angelegt, so wird, gemäß der Arbeitsweise der einzelnen Neuronen (siehe Abschnitte 1.2.1.1 und 1.2.1.2) und deren Verkettung untereinander, ein Ausgabe-Vektor $\mu(\vec{s})$ ermittelt. Dieser weicht in der Regel vom vorgegebenen Sollwert $\hat{\mu}(\vec{s})$ ab.

Nun sind alle beteiligten Perzeptronen des MLP zu trainieren. Eingesetzt werden hierbei der so genannte Backpropagation-Algorithmus oder dessen Derivate [RHW86, Cal99]. Grundlage des Lernverfahrens ist die Bestimmung des Fehlerwertes, d.i. die Differenz zwischen Soll- und Ist-Wert der Ausgabeschicht $(\hat{\mu}(\vec{s}) - \mu(\vec{s}))$. Dieser Fehler wird berechnet und danach von der Ausgangsschicht in Richtung Eingangsschicht zurück propagiert. Dabei werden in den Zwischenschichten jeweils die Fehler bezüglich der Zwischenergebnisse berechnet und nach vorne weitergegeben. Mit Hilfe dieser Fehler wird eine Korrektur der Netzgewichte und Schwellwerte vorgenommen [RHW86].

Bei der Art der Fehlerminimierung handelt es sich um ein Gradientenabstiegsverfahren, das den mittleren quadratischen Fehler minimiert. Die Geschwindigkeit des Lernverfahrens kann durch einen Parameter eingestellt werden, der – in Analogie zum Perzeptron – als Lernrate Λ bezeichnet wird.

1.2.2.3 Geometrische Interpretation

Betrachtet man die durch neuronale Netze klassifizierbaren Objekte, so sind sie durch ihre M messbaren Eigenschaften (z.B. Farbe, Form und Masse) charakterisiert. Prinzipiell kann jedes Objekt durch einen abstrakten Vektor beschrieben werden, der das Maß der Erfüllung jeder einzelnen Eigenschaft beinhaltet. Dieser Vektor zeigt auf die Position des zugehörigen Objektes im so genannten Merkmalsraum. Somit können die zu klassifizierenden Objekte als Punkte in Merkmalsräumen aufgefasst werden.

Zweckmäßigerweise sollten Objekte der gleichen Klasse möglichst dicht beieinander liegen. Damit würde sich der kausale Zusammenhang in der Lage der Objekte im Raum bzw. in der Nachbarschaft ähnlicher Objekte widerspiegeln. Eine derartige Nachbarschaftsbeziehung wird in Abbildung 1.8 illustriert: Während im linken Bild kein Zusammenhang zwischen Klasse (rot vs. grün) und Lage im Merkmalsraum erkennbar ist, entspricht die Lage der Objekte im rechten Bild dem geforderten Nachbarschaftskriterium.

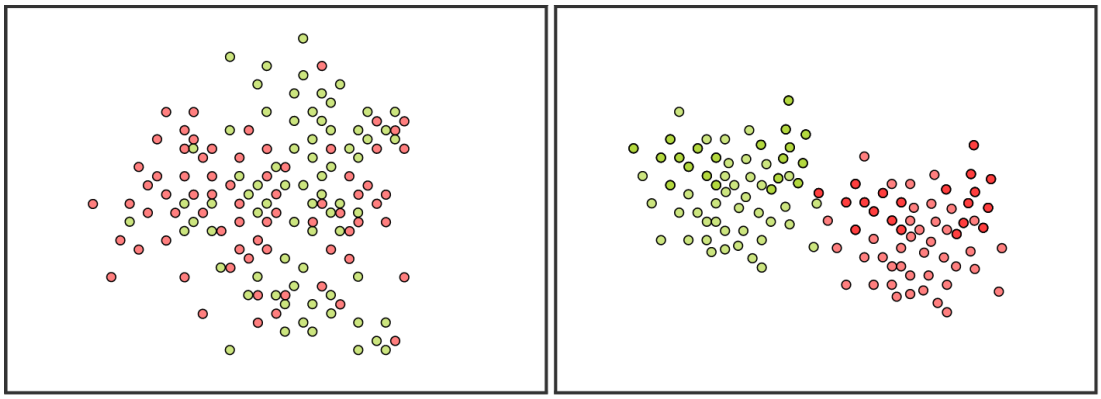

Abbildung 1.8: Zusammenhang zwischen Klasse (rot vs. grün) und Position im Raum.

Unter Maßgabe dieser Vereinfachung liegt die Problematik der Klassifizierung darin, eine mathematische Formulierung für diejenigen Hyperflächen zu finden, die die jeweils zusammengehörigen Objekte im Merkmalsraum voneinander trennen. Diese Trennung kann durch eine so genannte Diskriminanzfunktion $\delta_n(\vec{s}^j)$ erfolgen. Wenn ein zu klassifizierendes Objekt $\vec{s}^j$ der Klasse k^a angehört, so muss unter der Voraussetzung, dass zwischen K Klassen zu unterscheiden ist, für die Diskriminanzfunktion gelten:

$$\delta_a(\vec{s}^j) > \delta_{(i\neq a)}(\vec{s}^j) \,\forall\, i \tag{1.14}$$

$$\delta_a(\vec{s}^j) = \max\left[\delta_i(\vec{s}^j)\right]_{i=0}^{(K-1)} \tag{1.15}$$

Die trennende Hyperfläche zwischen den Punkten der Klasse k^a und denen der Klasse k^b ist dann gegeben durch die Gleichung

$$\delta_a(\vec{s}^j) - \delta_b(\vec{s}^j) = 0. \tag{1.16}$$

Die Klassifizierung mittels Hyperflächen ist besonders einfach, wenn sich diese Flächen durch eine oder mittels Kombination mehrerer Hyperebenen approximieren lassen. Im allereinfachsten Beispiel würde jede Hyperfläche durch genau eine Ebene approximiert werden, wobei diese den Merkmalsraum in Voronoi-Regionen

partitionieren würden [PS93]. Jede Ebene ließe sich in der so genannten Normalenform durch ihren Normalenvektor und den Abstand zum Ursprung angeben. Eine Entscheidungsebene, die zwischen a und b trennt, ist somit darstellbar als

$$E^{(a,b)} : \begin{pmatrix} w_0^{(a,b)} \\ w_1^{(a,b)} \\ \vdots \\ w_{(M-1)}^{(a,b)} \end{pmatrix} \cdot \vec{s}^T + w_M^{(a,b)} = 0 \tag{1.17}$$

Die Diskriminanzfunktionen $\delta_a(\vec{s}^j)$ und $\delta_b(\vec{s}^j)$ sind hier gegeben durch den Abstand des Punktes $\vec{s}^j$ von der Ebene $E^{(a,b)}$, der dadurch berechnet werden kann, dass $\vec{s}^j$ in die Ebenengleichung eingesetzt wird. Liegt nämlich die Ebenengleichung in der Hesse'schen Normalenform vor, d.h. der Normalenvektor wurde auf $||\vec{w}^{(a,b)}|| = 1$ normiert, so wird – durch Einsetzen des Merkmalsvektors $\vec{s}^j$ in die Gleichung – dessen Abstand zur Ebene $E^{(a,b)}$ bestimmt. Das Vorzeichen des Normalenvektors ist hierbei so zu wählen, dass die Gleichungen 1.14 und 1.15 erfüllt werden.

Somit lässt sich Gleichung 1.17 in die Perzeptronenformel 1.4 überführen. Dementsprechend kann jedes Perzeptron auch als Ebene im Merkmalsraum betrachtet werden. Gemäß seiner Dimension M soll der Merkmalsraum im Folgenden als $\underline{\underline{R}}^M$ bezeichnet werden. Zu klassifizierende Objekte s^j werden nun auch als Punkte $\vec{s}^j$ in $\underline{\underline{R}}^M$ betrachtet. Die Interpretation eines Perzeptrons als Hyperebene in diesem Raum führt zu den so genannten Support-Vektor-Maschinen.

1.2.2.4 Support-Vektor-Maschinen

Support-Vektor-Maschinen (SVM) sind spezielle Klassifizierer, die den Merkmalsraum mit Hilfe von Hyperebenen partitionieren, welche durch Perzeptronen implementiert werden [CST04, SS02].

Aufgrund ihrer Lage kann es schwierig sein, Klassen von Objekten im Merkmalsraum mit Hilfe einer Hyperebene zu trennen (vgl. Abbildung 1.5, links und Abbildung 1.8, links). Das Prinzip der Klassifikation mit SVM beruht daher auf dem Finden einer optimal trennenden Hyperebene in einem Hilfsraum, dessen Dimension wesentlich höher ist als die des ursprünglichen Merkmalsraumes. Dabei besteht die zugrunde liegende Idee darin, dass der Merkmalsraum mitsamt den zu klassifizierenden Objekten durch eine geeignete Abbildung ϕ so transformiert wird, dass eine lineare Trennbarkeit der Objekte möglich wird [VC74]. Diese Vorgehensweise, die bereits beim Kernel-Perzeptron Anwendung findet (vgl. Abschnitt 1.2.1.3), soll am folgenden Beispiel verdeutlicht werden:

Eine Anzahl S an M-dimensionalen Merkmalsvektoren sei aufgrund ihrer Anordnung in $\underline{\underline{R}}^M$ nicht linear trennbar. Die Transformationsvorschrift $\phi(s_1, s_2, \cdots s_M)$ sei gegeben als quadratisches Polynom:

$$\phi(\vec{s}) = \sum_{i=1}^{M-1} (\Phi_i \cdot s_i) + \sum_{i,j=1}^{M-1} \left(\Phi_{(i,j)} \cdot s_i \cdot s_j\right) + \Phi_M \tag{1.18}$$

Durch den Übergang zu quadratischen Polynomen entstehen „neue“ Eigenschaften, nämlich die Quadrate und die gemischten Produkte der „alten“ Merkmale. Für $M = 2$ lassen sich beispielsweise folgende Substitutionen durchführen: $\widetilde{s}_1 = s_1, \widetilde{s}_2 = s_2, \widetilde{s}_3 = s_1 \cdot s_2, \widetilde{s}_4 = (s_1)^2$ und $\widetilde{s}_5 = (s_2)^2$.

Hiermit ist anschaulich klar, dass ein spezieller, in $\underline{\underline{R}}^2$ nicht linear trennbarer Fall mit der zugehörigen Hyper*fläche* $E^{(a,b)}$ in einen linear trennbaren Fall in $\underline{\underline{R}}^5$ mit der Hyper*ebene* $\phi(E^{(a,b)})$ gemäß Gleichung 1.19 überführbar ist.

$$\phi(E^{(a,b)}) : \vec{\widetilde{w}} \cdot (\vec{\widetilde{s}})^T + \widetilde{w}_M = 0 \tag{1.19}$$

Hierzu müssen noch die Koeffizienten $\widetilde{w}_i$ bestimmt werden. Diese werden im Fall der SVM dadurch festgelegt, dass zunächst diejenigen Merkmalsvektoren gewählt werden, die voraussichtlich am dichtesten an den Entscheidungsebenen liegen. Diese speziellen Merkmalsvektoren werden auch Support-Vektoren genannt. Nun sind die Koeffizienten so zu berechnen, dass die Summe der Abstände *aller* Support-Vektoren von den Entscheidungsebenen maximal groß wird.

Die Transformation ϕ kann dabei sehr komplex sein. Die Dimension des Hilfsraumes ist prinzipiell nicht beschränkt. Dennoch ist die Berechnung einer Hyperebene mit vergleichsweise geringem Aufwand möglich, da die SVM so formuliert werden kann, dass die Trainingsdaten nur in Skalarprodukten auftreten [VC74, Vap98].

Nach einer Trainigsphase ist die Support-Vektor-Maschine gegebenenfalls in der Lage, den Testdatensatz korrekt zu klassifizieren. Auf neuen Beispielen wird die Klassifikationsregel in manchen Fällen Fehler machen. Diese Fehler vorher sicher zu erkennen, ist natürlich nicht möglich - sonst könnte man sie ja vermeiden. Allerdings wäre es hilfreich, wenn man für jedes klassifizierte Objekt angeben könnte, wie zuverlässig die Klassifikationsregel ist. Hierzu wurden für SVM in der Vergangenheit zwei Verfahren vorgeschlagen, die auf Modellierung bedingter Wahrscheinlichkeit basieren [Vap98, Pla99]. Somit sind SVM prinzipiell in der Lage, Zuverlässigkeitsinformationen zu extrahieren und auszugeben.

Gleichzeitig können mit ihrer Hilfe Klassifizierungen mit einer sehr niedrigen Fehlerrate durchgeführt werden. Grundlegendes Vorwissen kann durch die Wahl geeigneter Kernel-Funktionen und Testdatensätze integriert werden. Demgegenüber stehen jedoch eine vergleichsweise lange Laufzeit, die geringe Transparenz sowie die Notwendigkeit der Trainingsphase und des zugehörigen manuell zu klassifizierenden Testdatensatzes.

Die Nische der schnell durchzuführenden Klassifizierungen mit Hilfe einer Partitionierung des Merkmalsraumes besetzt der so genannte Signalraumdetektor. Dieses Klassifizierungsverfahren wird im Folgenden vorgestellt und im weiteren

Verlauf der Arbeit hinsichtlich der in Abschnitt 1.1 dargestellten Entwurfsziele untersucht und modifiziert.

1.2.3 Die Signalraumdetektion

Voraussetzung für die Signalraumdetektion ist es, dass alle zu klassifizierenden Objekte über eine endliche Anzahl M von reell quantifizierbaren Eigenschaften bestimmbar sind. Dabei sei im Folgenden von $M \geq 1$ ausgegangen. Für den Trivialfall $M = 0$ gilt: Die Objekte haben keine unterschiedlichen Eigenschaften und gehören somit alle derselben Klasse an.

1.2.3.1 Modell der Signalraumdetektion

Für die Signalraumdetektion seien die zu klassifizierenden Objekte als s^j gegeben. Jede Eigenschaft e_i mit $i \in \{0, 1, \cdots (M-1)\}$ wird als Dimension des Merkmalsraumes $\underline{\underline{R}}^M$ aufgefasst, der durch die Einheitsvektoren $\vec{e}_0$ bis $\vec{e}_{(M-1)}$ aufgespannt wird.

Somit gibt s_i^j den Wert der Eigenschaft e_i des zu klassifizierenden Objektes s^j an. Dieser Wert ist bestimmbar durch skalare Multiplikation des Merkmalsvektors $\vec{s}^j$ mit dem Einheitsvektor $\vec{e}_i$, der die Dimension e_i im Signalraum repräsentiert:

$$s_i^j = \vec{s}^j \cdot \vec{e}_i^T \tag{1.20}$$

Die Anzahl der zu klassifizierenden Objekte sei im Folgenden als S angenommen. Weiterhin sei mit K die Anzahl der möglichen Klassen gegeben, denen ein Objekt s^j zugeordnet werden kann. Für die folgenden Betrachtungen wird von $K \geq 2$ ausgegangen. Die Fälle, dass s^j keiner ($K = 0$) oder genau einer Klasse ($K = 1$) zugeordnet werden kann, werden aufgrund ihrer Trivialität ausgeklammert.

Eine weitere zwingende Voraussetzung für die Signalraumdetektion ist das bereits in Abschnitt 1.2.2.3 diskutierte Nachbarschaftskriterium, dass diejenigen Objekte, die derselben Klasse angehören, durch ihre Lage in $\underline{\underline{R}}^M$ von denjenigen Objekten, die zu anderen Klassen gehören, unterscheidbar sein müssen. Demnach ist eine Objektkonstellation, wie im linken Teil der Abbildung 1.8, für die Signalraumdetektion ungeeignet.

Die Nachbarschaftsbeziehung muss zu einer linearen Trennbarkeit der Objekte nach ihrer Klassenzugehörigkeit führen: Die partitionierenden Hyperflächen müssen sich durch einen Satz von Hyperebenen gemäß Abb. 1.9 approximieren lassen. Durch die Partitionierung müssen $U \geq K$ Unterräume u^i entstehen, wobei sich dann jedem Unterraum mittels surjektiver Abbildung $\varepsilon_1 : u^i \rightarrow k^i$ jeweils genau eine Klasse k^j zuordnen lässt.

Anschließend lässt sich die Klassifizierung von s^j durch eine Lagebestimmung von $\vec{s}^j$ durchführen, bei der der umschließende Unterraum mit $\varepsilon_0 : \vec{s}^j \rightarrow u^i$ ermittelt wird. Ist der Unterraum bestimmt, so lässt ε_1 den Rückschluss auf

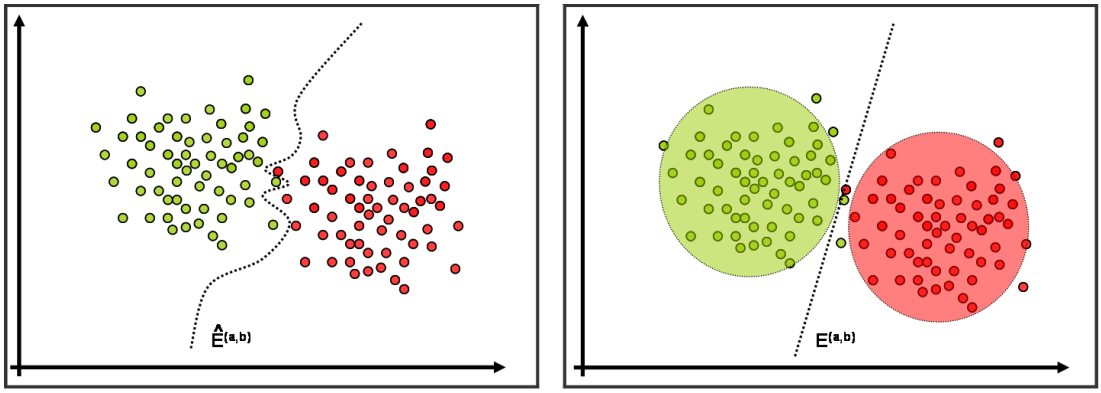

Abbildung 1.9: Approximation einer Hyperfläche. Links: Exakte Partitionierung des Signalraumes. Rechts: Approximation der Hyperfläche durch Hyperebene.

die zuzuordnende Klasse zu. Dieser Klassifizierungsvorgang soll im Folgenden als $\varepsilon : \vec{s}^j \rightarrow k^i$ beschrieben werden. Der Soll-Wert der Klassifizierung, also die korrekt zuzuordnende Klasse, wird dementsprechend als $\hat{k} = \hat{\varepsilon}(\vec{s}^j)$ bezeichnet.

Somit entstehen bei der Signalraumdetektion zwei Problemgruppen:

1. Geeignete Partitionierung des Merkmalsraumes,

2. Bestimmung der Lage eines zu klassifizierenden Objektes.

Die Partitionierung des Merkmalsraumes kann auf Basis eines vorhandenen (statistischen) Modells der zu klassifizierenden Objekte erfolgen [JM97, MJ98]. Eine weitere Methode ist es, die Entscheidungsebenen anhand so genannter Referenzobjekte auszurichten. Hierzu wird ein Satz von R Objekten r^i bzw. deren zugehörigen Merkmalsvektoren $\vec{r}^i$ verwendet, wobei die jeweils zuzuordnenden Klassen $\hat{\varepsilon}(\vec{r}^i)$ a priori bekannt sein müssen. Basierend auf diesen Referenzobjekten ist es – analog zu den Support-Vektoren der SVM bzw. der Trainingsmenge des MLP – möglich, eine sinnvolle Ebenenkonstellation zu bestimmen. Darauf aufbauend kann dann die Partitionierung des Merkmalsraumes, gemäß den bei der Klassifizierung herrschenden Randbedingungen, angepasst werden.

Angenommen, der Merkmalsraum wäre bereits mittels Hyperflächen in Unterräume aufgeteilt, so besteht eine effiziente Methode zur Lagebestimmung des Merkmalsvektors $\vec{s}^j$ darin, den Abstand des Punktes im Raum zu allen umhüllenden Hyperflächen zu bestimmen [AMN+98]. Die Lagebestimmung relativ zu einer Grenzfläche ist jedoch schwierig, wenn diese Hyperfläche komplizierte Formen annimmt. Zur Komplexitätsreduktion wird daher bei der Signalraumdetektion jede Hyperfläche durch eine endliche Anzahl an Hyperebenen approximiert. Mit dieser Näherung ist es möglich, einfach zu beschreibende Unterräume zu verwenden und somit den Aufwand bei der Lagebestimmung ε_0 zu reduzieren.

Ist eine eindeutige räumliche Trennung von Objekten, die unterschiedliche Klassen repräsentieren, nicht durchführbar oder nicht im Rahmen der erlaubten algorithmischen Komplexität zur Lagebestimmung möglich, so kann eine Partitionierung von $\underline{\underline{R}}^M$ zumindest dazu dienen, Bereiche dominanter Wahrscheinlichkeit von Bereichen geringerer Aufenthaltswahrscheinlichkeit zu trennen. Hierzu wird die Aufenthaltswahrscheinlichkeit $\omega(s^j)$ eines Objekts s^j in einem der Klasse k^i zugeordnetem Unterraum als „dominant“ bezeichnet, wenn sie gemäß Gleichung 1.21 größer ist als die Wahrscheinlichkeit der Zugehörigkeit zu jeder anderen Klasse in Verbindung mit dem gleichzeitigen Auftauchen eines Ereignisses $\vec{\eta}$ (beispielsweise eines Messfehlers), das die Position von s^j derart verschiebt, dass das so entstandene Objekt den umhüllenden Unterraum fälschlicherweise nach $u \rightarrow k^i$ gewechselt hat.

$$\omega(s^j \in k^i) \geq \omega(s^j \notin k^i | \vec{\eta}) \tag{1.21}$$

Hier zeigt sich, dass die Anzahl der Klassen, in die ein Signalraumdetektor die Eingangsdaten klassifizieren kann, systembedingt begrenzt ist. Für mindestens eine Klasse k^i ist die Wahrscheinlichkeit mit $\omega(s^j \in k^i) \leq \frac{1}{K}$ begrenzt. Für hinreichend große K kann diese Wahrscheinlichkeit kleiner als jeder Mess- und Quantisierungsfehler werden. Daher wird im Folgenden von einer „genügend kleinen Anzahl“ Klassen ausgegangen.

Somit können mindestens K Unterräume von $\underline{\underline{R}}^M$ gefunden werden, in denen die zu einer Klasse k^i zuzuordnenden Merkmalsvektoren $\vec{s^j}$ eine dominante Aufenthaltswahrscheinlichkeit aufweisen. Statistisch gesehen stellen diejenigen Hyperflächen, die alle Punkte verbinden, für die $\omega(s^j \in k^i) \geq 0.50$ gilt, die beste Wahl zur Partitionierung des Merkmalsraumes dar.

Die Bestimmung dieser Hyperflächen kann jedoch sehr aufwändig sein. Je nach Beschaffenheit der Hyperflächen ist die Lagebestimmung des Merkmalsvektors bezüglich der Hyperfläche $\varepsilon_0(\vec{s})$ ebenfalls mit hohem Aufwand verbunden. Aus diesem Grund ist auch hier die Approximation der Hyperflächen durch Hyperebenen eine sinnvolle Alternative. Durch die Partitionierung entstehen Unterräume, die unter Anwendung von ε_1 bei vergleichsweise geringem algebraischem Aufwand einer Klasse zugeordnet werden können.

Allerdings entstehen auch Bereiche, die nicht eindeutig einer Klasse zuzuordnen sind. Das sind mindestens all jene Punkte, die exakt auf der Hyperfläche liegen. Zusätzlich bewirkt die Approximation einen Fehler bei der Partitionierung von $\underline{\underline{R}}^M$, der sich in der Genauigkeit der Klassifizierung niederschlägt. Fehlerrate und algorithmische Komplexität stehen sich also bei der Signalraumdetektion als komplementäre Entwurfsziele gegenüber. Demnach ist es eine wissenschaftliche Herausforderung, Verfahren zu finden, die ein unter bestimmten Randbedingungen optimiertes Verhältnis von algorithmischer Komplexität und Fehlerrate bieten. Hierzu und um sie für verschiedene Verfahren vergleichen zu können, müssen zunächst Komplexität und Fehlerrate definiert werden.

1.2.3.2 Fehlerbestimmung und Modellierung

Durch Approximierung der Hyperflächen mittels Hyperebenen entsteht bei der Signalraumdetektion im Fall binärer Entscheidungen grundsätzlich immer dann ein systematischer Fehler, wenn die Linien, die die Bereiche gleicher Wahrscheinlichkeit $\omega(s^j \in k^i)$ verbinden – die so genannten Equi-Probability-Linien (EQL) – nicht identisch mit den partionierenden Hyperebenen sind [CS89].

Zusätzlich zu diesem systematischen Fehler kommen Ungenauigkeiten von Messwerten und Sensoren, Quantisierungsfehler sowie Abweichungen vom theoretischen Modell aufgrund von Beschränkungen bei der Merkmalserfassung bzw. bei der Implementierung des Klassifizierers hinzu. Alle Fehler zusammen können sich als nicht korrekte Zuordnung eines Objektes zu seiner zugehörigen Klasse niederschlagen.

Zur Definition des Fehlers sei zunächst die *korrekte* Zuordnung eines Objektes s^j zu einer Klasse, der Soll-Wert, gegeben durch $\hat{\varepsilon}(\vec{s}^j)$. Das Ergebnis der Klassifizierung mittels Signalraumdetektion, der Ist-Wert, sei als $\varepsilon(\vec{s}^j)$ bezeichnet.

Immer dann, wenn sich Soll- und Ist-Wert unterscheiden ($\hat{\varepsilon}(\vec{s}^j) \neq \varepsilon(\vec{s}^j)$), liegt ein Klassifizierungsfehler vor. Hierzu sei eine Fehlerfunktion $\varphi(x, y)$ genau so definiert, dass

$$\varphi(x,y) = \begin{cases} 1 & \text{wenn } x \neq y, \\ 0 & \text{wenn } x = y. \end{cases} \tag{1.22}$$

Die Fehlerrate[4] ρ^{ER} der Klassifizierung sei gegeben durch:

$$\rho^{ER} = \frac{1}{S} \sum_{j=0}^{S-1} \varphi\left(\hat{\varepsilon}(\vec{s}^j), \varepsilon(\vec{s}^j)\right) \tag{1.23}$$

Für den Fall einer binären Entscheidung wird die Fehlerrate auch Bitfehlerrate[5] genannt, und $\varphi(x, y)$ aus Gleichung 1.23 entspricht der Booleschen XOR-Verknüpfung:

$$\rho^{BER} = \frac{1}{S} \sum_{j=0}^{S-1} \left(\hat{\varepsilon}(\vec{s}^j) \oplus \varepsilon(\vec{s}^j)\right) \tag{1.24}$$

Soll eine fehlerfreie Zuordnung aufgrund der messbaren Eigenschaften s_i^j des Objektes s^j möglich sein, so muss es mindestens $R \geq 1$ Referenzobjekte geben, für die gilt:

$$\hat{\varepsilon}(\vec{s}^j) = \varepsilon(\vec{s}^j = \vec{r}^j) \; \forall \; j \in [0, (R-1)] \tag{1.25}$$

[4] engl.: Error-Rate (ER)
[5] engl.: Bit-Error Rate (BER)

Diese Referenzobjekte werden als r^j mit Merkmalsvektor $\vec{r}^j$ und Eigenschaften r_i^j bezeichnet. Der Abstand des Merkmalsvektors eines beliebigen zu klassifizierenden Objektes $\vec{s}^j$ lässt sich über $||\vec{r}^i - \vec{s}^j||$ relativ zum Referenzobjekt angeben. Er lässt sich interpretieren als der Einfluss eines additiven Fehlerevents, über das $\vec{r}^i$ mittels Rauschvektor $\vec{\eta}$ nach $\vec{r}^i + \vec{\eta} = \vec{s}^j$ verschoben wird. Für jeden Unterraum u^i sei nun genau jenes Objekt r^i als Referenzobjekt bezeichnet, das im korrekten Unterraum $u^j = \varepsilon_0(\vec{r}^i)$ mit $(\varepsilon_1(u^j) = \hat{\varepsilon}(\vec{r}^i)$ liegt, so dass der Term 1.26 minimal für das gewählte i wird.

$$\sum_{j=0}^{S-1} \varphi\left(\varepsilon(r^i), \hat{\varepsilon}(s^j)\right) \cdot ||\vec{r}^i - \vec{s}^j|| \tag{1.26}$$

Die Merkmalsvektoren $\vec{r}^i$ können im Merkmalsraum als Punkte in ausgezeichneter Lage interpretiert werden: Idealerweise bilden alle $\vec{s}^j$ um $\vec{r}^i$ Punktwolken, so genannte Cluster c^i. Die Konstellation der Punkte im Raum lässt sich durch die Form dieser Cluster charakterisieren. Hierzu betrachtet man die EQL bezüglich der einzelnen $\vec{r}^i$. Formen die EQL Kreise ($M = 2$), Kugeln ($M = 3$) bzw. Hyperkugeln ($M > 3$), so lässt sich das so genannte AWGN-Modell ansetzen [DR87]: Alle $\vec{s}^j$ sind modellierbar als Summe aus einem Aufpunkt $\vec{r}^i$ und einem Rauschvektor $\vec{\eta}$, der durch weißes Rauschen[6] charakterisiert wird. In diesem Spezialfall lassen sich die Unterräume u^i durch Bildung der so genannten Voronoi-Regionen bestimmen [PS93]. Dieses Verfahren ist in Abbildung 1.9 illustriert.

Neben dem links in Abbildung 1.8 skizzierten Fall, dass Objekte gar keine Cluster formen und daher die Signalraumdetektion nicht angewendet werden kann, wird im Folgenden zwischen vier besonderen Formen der EQL unterschieden. Diese Varianten der Cluster-Anordnung werden durch Abbildung 1.10 illustriert:

1. Die erste Variante besteht darin, dass der Zusammenhang zwischen $\vec{r}^i$ und c^i durch Addition von AWGN modelliert werden kann [DR87]. Im einfachsten Fall sei $\vec{\eta}$ ein Vektor, dessen Komponenten η_i zufällig erzeugt wurden und dessen Länge der Gaußverteilung folgt. In diesem Fall liegen alle gleich wahrscheinlich auftretenden Punkte des Clusters auf der Oberfläche einer Hyperkugel, deren Zentrum durch $\vec{r}^i$ gegeben ist.

2. Ein weiterer Fall ist dadurch gegeben, dass die Elemente η_j des Rauschvektors durch gaußförmiges Rauschen mit einer jeweils verschiedenen Varianz gebildet werden. Mathematisch gesehen ist dies ein Spezialfall einer lineare Transformation von auftretendem AWGN, der durch eine rotationsfreie Rücktransformation in den Fall (1) überführt werden kann (Abbildung 1.10 rechts oben).

3. Ergibt sich die Anordnung durch allgemeine lineare Transformation τ aus dem AWGN-Fall, so lässt sich durch eine geeignete Rücktransformation

[6] engl.: additive white gaussian noise (AWGN)

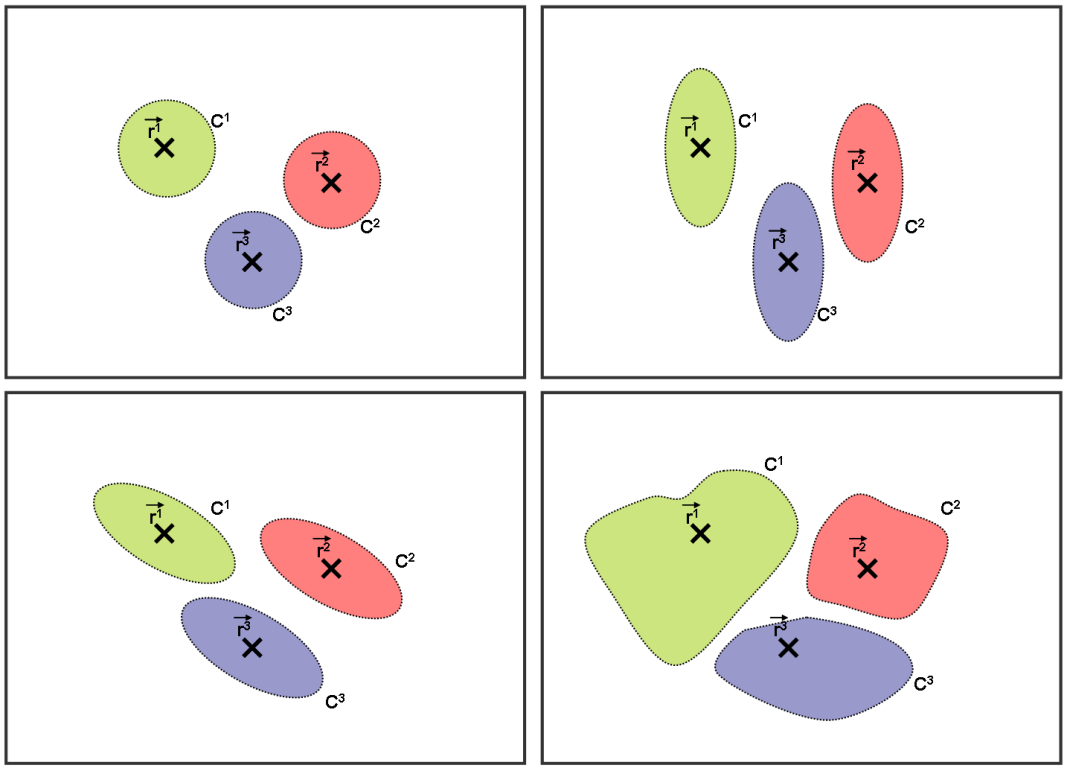

Abbildung 1.10: Cluster-Form für: AWGN, spezielle Lineartransformation, allgemeine Lineartransformation und nichtlineare Transformation des AWGN-Falles (v.l.o.n.r.u.).

τ^{-1} dieser Fall ebenfalls nach (1) überführen [Str00] (Abbildung 1.10 links unten).

4. Schließlich gibt es noch eine Reihe von weiteren Beziehungen zwischen Clustern und Referenzpunkten. Diese sollen für die folgenden Betrachtungen als „komplexe“ (typischerweise nichtlineare) Zusammenhänge bezeichnet werden (Abbildung 1.10 rechts unten).

Für die Fälle (1) bis (3) lassen sich die Entscheidungsebenen $E^{(a,b)}$ besonders einfach bestimmen. Wie in den folgenden Abschnitten gezeigt wird, ist die Signalraumdetektion in derartig zu modellierenden Fällen besonders effizient einsetzbar. Hierzu werden zwei spezialisierte Detektoren verwendett: die Basisversion des Signalraumdetektors (SSD) und der Signalraumdetektor mit „Noise Whitening“ (WSSD).

1.2.3.3 Der Signalraumdetektor

Der Signalraumdetektor basiert auf dem Prinzip der Klassifizierung durch Partitionierung des Merkmalsraumes (vgl. Abschnitt 1.2.2.3). Dabei wird der Sig-

nalraum durch eine Vielzahl von Entscheidungsebenen so aufgeteilt, dass jeder der Cluster c^0 bis $c^{(R-1)}$ durch mindestens eine Entscheidungsebene von jedem Cluster, der eine andere Klasse repräsentiert, getrennt wird.

Bei der Originalversion des SSD werden Entscheidungsebenen verwendet, die durch die Grenzen der Voronoi-Regionen gegeben sind. Die Diskriminanzfunktionen sind allein durch das Vorzeichen der eingesetzten Ebenengleichung sowie eine nachgeschaltete Boolesche Logik β gegeben. Dabei bildet β die eindeutige Lage eines beobachteten Signalpunktes relativ zu allen Entscheidungsebenen surjektiv auf die Zielmenge der zuzuordnenden Klassen ab, was durch die Diskriminanzfunktion 1.27. verdeutlicht wird.

$$\delta_a : \beta \left[sign(\begin{pmatrix} w_0^{(a,b)} \\ w_1^{(a,b)} \\ \vdots \\ w_{(M-1)}^{(a,b)} \end{pmatrix} \cdot (\vec{s}^j)^T + w_M^{(a,b)}) \right]_{a=0,b=1}^{a<b \forall a,b} \tag{1.27}$$

Für eine hohe Anzahl an Referenzpunkten sind entsprechend viele Entscheidungsebenen zu implementieren und somit Multiplikationen durchzuführen, denn für jedes Punktepaar $(\vec{r}^i, \vec{r}^j)$ ist mindestens eine Entscheidungsebene erforderlich. Diese ist durch die M Elemente des Normalenvektors gegeben. Die Anzahl der zu implementierenden Multiplikationen ergibt sich aus Gl. 1.28.

$$M \cdot \begin{pmatrix} R \\ 2 \end{pmatrix} = M \cdot \frac{R^2 - R}{2} \tag{1.28}$$

Der Aufwand der Klassifizierung steigt also quadratisch mit der Anzahl der Referenzpunkte. Somit lässt sich ein Signalraumdetektor überall dort gut einsetzen, wo die Dimension des Merkmalsraumes und die Anzahl der Referenzpunkte klein sind. Aus diesem Grund wird er bislang in der Praxis „nur“ für niedrig komplexe Signalräume eingesetzt, beispielsweise als Detektor von Signalfolgen im Festplattenlesekanal mit $M = 3$ und $R = 8$.

Auch für vergleichweise niedrig komplexe Klassifizierungen erscheint es sinnvoll, die Multiplikationen zu parallelisieren, um die Geschwindigkeit des Klassifizierers nicht künstlich zu drosseln. Dies ist auch praktisch durchführbar, weil zwischen den einzelnen Multiplizierern, die die Komponenten der Normalenvektoren der Entscheidungsebenen repräsentieren, keine Abhängigkeit besteht. In Abbildung 1.11 ist beispielhaft die Implementierung eines Signalraumdetektors bei Parallelisierung der Multiplikationsvorgänge dargestellt: Ein dreidimensionaler Eingangsvektor wird mit Hilfe von zwei Entscheidungsebenen klassifiziert. Die zwei zugehörigen Normalenvektoren werden durch insgesamt sechs Konstantenmultiplizierer implementiert. Der Ebenenabstand zum Ursprung $w_M^{(a,b)}$ wird mittels Schwellwertschalter abgebildet, über dessen Ausgang das Vorzeichen der Ebenengleichung in die Detektorlogik β eingespeist wird.

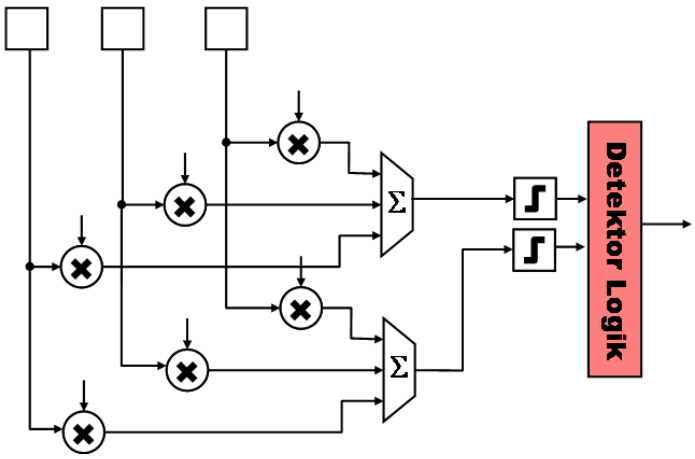

Abbildung 1.11: Blockschaltbild eines Signalraumdetektors mit parallel ausgeführter Skalarmultiplikation für zwei Entscheidungsebenen und $M = 3$.

Folglich ist, um die Signalraumdetektion auch für komplexere Klassifizierungsprobleme erfolgreich einzusetzen, eine Komplexitätsreduktion des Verfahrens erforderlich. Dabei liegt der Fokus auf einer Reduktion der Anzahl der benötigten Multiplizierer. Dies kann durch folgende Vorgehensweisen erzielt werden:

1. Mehrere Cluster, die der gleichen Klasse zugeordnet sind, werden zu einem gemeinsamen Unterraum zusammengefasst [JM98].

2. Die Parallelität von Entscheidungsebenen wird für algebraische Vereinfachungen verwendet [SHS00].

3. Mittels Signalraumtransformation wird eine Konstellation erreicht, in der die Klassifizierung mit achsenparalleler Trennung durchgeführt werden kann.

4. Durch eine geeignete Transformation werden Komponenten der Normalenvektoren auf $w_i \in \{-1; +1\}$ abgebildet.

Die beschriebenen Verfahren 2, 3 und 4 sind in Abbildung 1.12 illustriert. Der obere Teil der Darstellung zeigt, wie der Signalraum $\underline{\underline{R}}^3$ durch drei Ebenen partitioniert wird. Die mittlere Zeichnung verdeutlicht, wie die Komplexität reduziert wird, wenn Entscheidungsebenen parallel zueinander liegen: Dadurch, dass die Normalenvektoren identisch sind, entfallen bei P parallelen Ebenen $M \cdot (P - 1)$ Multiplikationen.

In der unteren Grafik wird schließlich verdeutlicht, wie die Anzahl der Multiplizierer durch „strength reduction“ bei den Faktoren 0 (3. Fall) und 1 (4. Fall) verringert werden kann.

Einschränkend auf die Einsatzfähigkeit des Signalraumdetektors wirkt sich neben der erforderlichen Komplexitätsreduktion auch das zugrunde liegende Modell aus. Voraussetzung für den Einsatz des originalen Signalraumdetektors ist die Modellierbarkeit der Cluster als AWGN um die Referenzpunkte. In der Praxis stößt ein derartiges Modell schnell an seine Grenzen. Der so genannte WSSD-Detektor erweitert das Signalraumkonzept für den Fall des farbigen Rauschens.

1.2.3.4 Der WSSD-Detektor

Für das Konzept des WSSD-Detektors (Signalraumdetektor mit Noise Whitening) wird angenommen, der existierende Merkmalsraum wäre durch eine lineare Transformation τ aus dem AWGN-Fall erzeugt worden. Der Signalraum $\tau(\underline{\underline{R}}^M)$ wird erzeugt, indem jeder Punkt $\vec{s}^j$ gemäß Gleichung 1.29 transformiert wird.

$$\tau : \vec{s}^j \rightarrow \underline{\Phi} \cdot \vec{s}^j + \vec{\Phi} \tag{1.29}$$

Ziel ist es zunächst, diese Transformationsvorschrift τ zu finden, um sie in τ^{-1} zu invertieren. Durch Anwendung von τ^{-1} kann der bestehende Merkmalsraum wiederum in den AWGN-Fall transformiert werden. Dort erfolgt dann die Bestimmung der optimalen Entscheidungsebenen durch Bildung der Voronoi-Regionen.

Werden nun die Entscheidungsebenen durch erneute Anwendung der Transformationsvorschrift in $\tau(E^{(a,b)})$ umgewandelt, so lässt sich zeigen, dass – bis auf den bereits in Abschnitt 1.2.3.2 beschriebenen systematischen Fehler – die entstehende Ebenenkonstellation optimal für diesen Fall des farbigen Rauschens ist [SHS00].

Daher weist der WSSD-Detektor unter Einfluß von farbigem Rauschen eine deutlich niedrigere Fehlerrate als die Basisversion des SSD auf [Str00]. Weitere Konsequenz ist, dass eine Verringerung der Fehlerrate nicht über eine Veränderung der partitionierenden Ebenenkonstellation möglich ist.

Betrachtet man die Diskriminanzfunktion 1.27 bzw. das Implementierungsbeispiel in Abb. 1.11, so stellt man fest, dass ein erheblicher Teil der berechneten Zwischenwerte nicht weiter verwendet wird. Allein das ermittelte Vorzeichen bezüglich der Lage relativ den Entscheidungsebenen $E^{(a,b)}$ geht in die Boolesche Algebra β ein, die letztlich über die Klasse entscheidet.

Betrachtet man neben dem Vorzeichen des Ebenenabstandes auch dessen Betrag, so lassen sich zusätzliche Informationen während des Detektionsprozesses gewinnen. Für den konventionellen Signalraumdetektor ist diese Zusatzinformation, mit deren Hilfe sich die Zuverlässigkeit der Entscheidung abschätzen lässt, zunächst nicht nutzbar. Prinzipiell existiert jedoch eine Reihe von Verfahren, die es ermöglichen, aus der Zuverlässigkeit der Entscheidung einen Gewinn bei der Datendecodierung und Fehlerkorrektur zu ziehen [Hee98, HOP96].

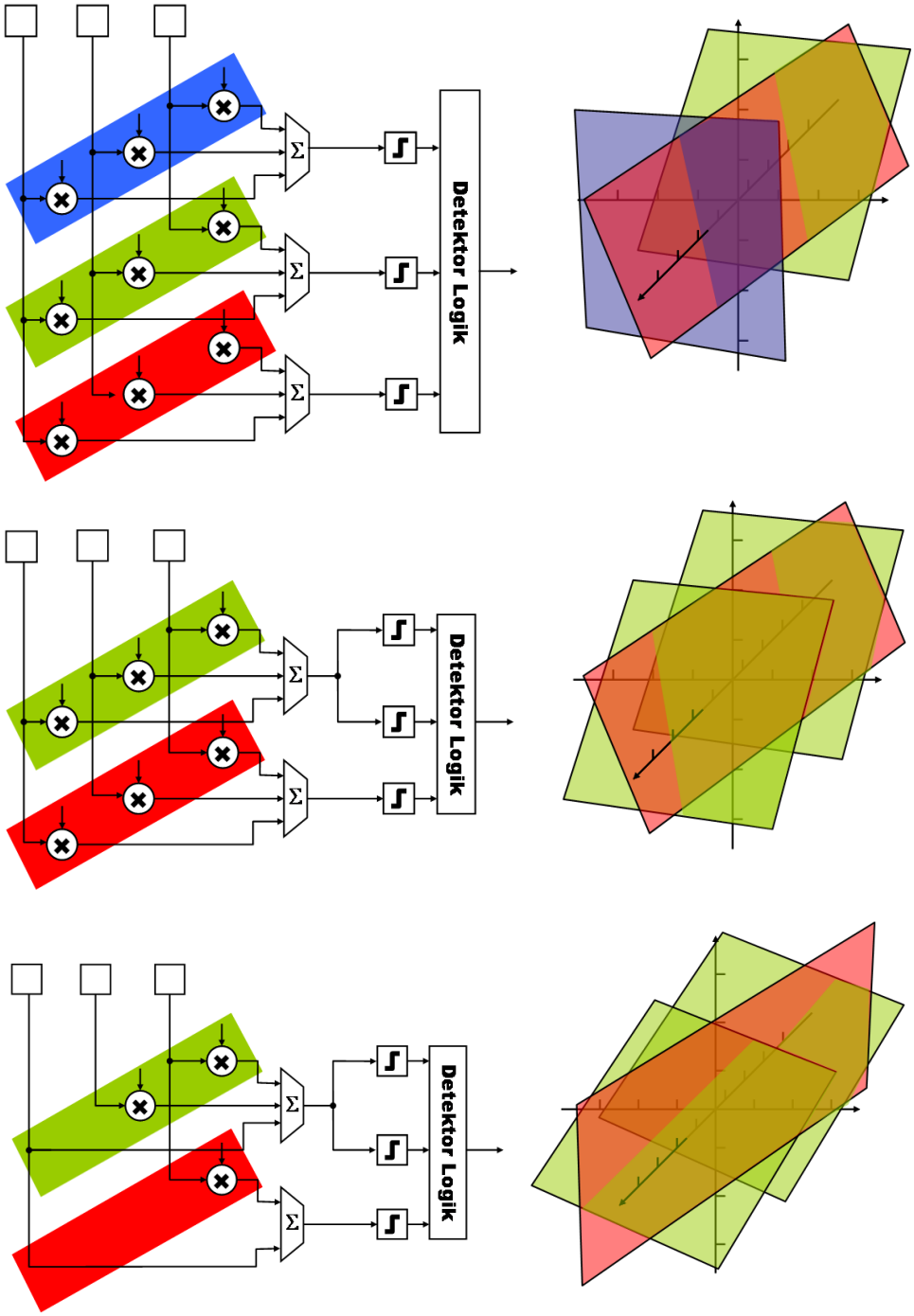

Abbildung 1.12: Komplexitätsreduktion des Signalraumdetektors. Oben: Ursprüngliche Signalraumkonstellation. Mitte: Parallele Entscheidungsebenen. Unten: Spezielle Signalraumkonstellation mit Null- und Eins-Faktoren

Diese Betrachtungen führen zum S3D, dem so genannten Soft-Output-Signalraumdetektor (einer Erweiterung des Basiskonzeptes), der zusätzlich Zuverlässigkeitsinformationen zur getroffenen Entscheidung ausgibt.

1.2.3.5 Der S3D-Detektor

Geometrisch geben die Ebenengleichungen des Signalraumdetektors den Abstand von $\vec{s}^{j}$zur Entscheidungsebene $E^{(a,b)}$ an. Ist dieser Abstand Null, so liegt $\vec{s}^{j}$ genau auf der Entscheidungsebene, und eine eindeutige Entscheidung ist nicht möglich. Je größer der Abstand von dieser „unsicheren“ Entscheidungsschwelle ist, desto höher ist die Wahrscheinlichkeit, dass die Entscheidung korrekt ist [SD02].

Dieses anschauliche Verfahren der Betrachtung des Ebenenabstandes als Maß für die Zuverlässigkeit der Entscheidung wurde bereits in mehreren Publikationen vorgestellt und dahingehend untersucht, inwiefern die Verwendung der Zuverlässigkeitsinformation zur Verbesserung der Fehlerrate verwendet werden kann [SSHS03, SMO01, Ste02].

Für die Implementierung des S3D-Detektors muss vor der Vorzeichenbetrachtung mittels Schwellwertbildung im ursprünglichen Blockschaltbild des SSD (siehe Abbildung 1.11) ein weiterer Zweig zur Extraktion von Zuverlässigkeitsinformationen eingefügt werden [Sch04]. Dieser Ansatz wird in Abbildung 1.13 illustriert.

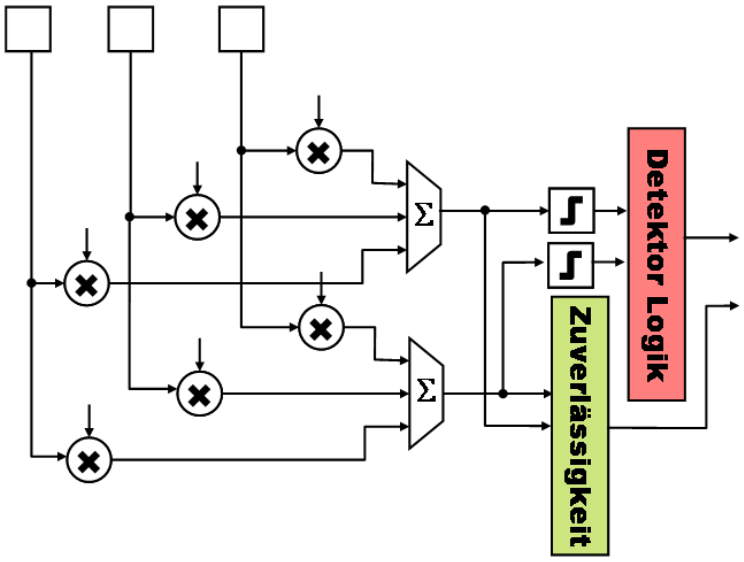

Abbildung 1.13: Blockschaltbild eines Signalraumdetektors mit Einheit zur Extraktion von Zuverlässigkeitsinformation.

1.3 Herausforderungen bei der Signalraumdetektion

Die vorgestellte Signalraumdetektion ist ein Verfahren, um Objekte anhand ihrer Eigenschaften zu klassifizieren. Sind die Objekte als Summe aus Referenzobjekt und Rauschvektor darstellbar, wobei der Rauschvektor als AWGN modelliert werden kann, so sind die Ebenengleichungen des Signalraumdetektors durch die Grenzen der so genannten Voronoi-Regionen gegeben.

Für Signalräume, die sich durch lineare Transformation, wie dargelegt, bestimmen lassen, ist mit dem WSSD-Detektor eine für diese Fälle optimale Ebenenkonstellation gegeben [SHS00]. Die Bestimmung optimierter Ebenengleichungen, die sich nicht durch lineare Transformation, wie zuvor, ergeben, ist ein bedeutsames Thema, um weitere Anwendungsgebiete für die Signalraumdetektion zu erschließen.

Wesentlich für die Ausdehnung der Signalraumdetektion auf weitere Anwendungsgebiete ist ihre Ergänzung um ein adaptives Konzept, das geeignet ist, Parameterschwankungen und Bauteilvariationen auszugleichen, ohne ein komplettes Re-Design der Schaltung durchführen zu müssen.

Schließlich ist besonderes Augenmerk der Klassifizierungsgeschwindigkeit zu widmen. Aus diesem Grund ist zu untersuchen, ob und wie die Geschwindigkeit der Signalraumdetektion durch eine Komplexitätsreduktion und algorithmische Optimierung nachhaltig gesteigert werden kann.

Im weiteren Verlauf dieser Arbeit werden diese Problemstellungen adressiert und Lösungsmöglichkeiten aufgezeigt. Es wird auch dargestellt, wie die Kombination der entwickelten Verfahren den Weg öffnet für den Einsatz der Signalraumdetektion bei bislang nicht mittels SSD handhabbaren Klassifizierungsproblemen.

Zu diesem Zweck wird die Arbeit in vier Schwerpunkte gegliedert:

1. Signalraumdetektion in komplexen Merkmalsräumen.

2. Adaptive Klassifizierungsverfahren.

3. Klassifizierung von mehreren Symbolen gleichzeitig.

4. Kombination der vorgestellten Verfahren.

Alle Schwerpunkte werden jeweils mit praktischen Beispielen unterlegt. Sie entstammen sowohl der Signalverarbeitung für magnetische Kanäle, der Bildverarbeitung als auch der Klassifizierung statistischer Daten. Diese Vielfalt soll die Relevanz der bearbeiteten Thematik für eine Vielzahl an Anwendungen der Signalverarbeitung demonstrieren und exemplarisch unter Beweis stellen. Des weiteren sind die Inhalte der vier Schwerpunkte und die Untersuchungsschritte folgendermaßen charakterisiert:

1. Die Komplexität der Detektionsalgorithmen hängt in hohem Maß von der Heuristik der zu analysierenden Signalfolgen bzw. des Signalraumes ab. Besonders geartete Signalräume und Signalfolgen können zu besonders einfachen Konstallationen von Entscheidungsebenen und somit zu Detektorvarianten geringer Komplexität führen [SHS00]. In Kapitel 2 wird daher untersucht, inwiefern die Transformation des Signalraumes in einen für die Klassifizierung günstigeren Signalraum eine Komplexitätsreduktion des Gesamtsystems ermöglicht.

2. Die Genauigkeit der Klassifizierung hängt insbesondere von der Signalqualität ab. Da praktisch alle zu klassifizierenden Signale durch Umwelteinflüsse und Messungenauigkeiten gestört sind, hängt die Optimierung der Entscheidungsebenen ebenfalls von der tatsächlichen Signalkonstellation ab, die typischerweise im Vorfeld zwar modellierbar, jedoch nur unvollständig bekannt ist. Daher erscheint es zweckmäßig, die Entscheidungsebenen an die jeweilige Signalraumkonstellation anzupassen, um eine hohe Klassifizierungsgenauigkeit zu erreichen. Auf die Adaption von Entscheidungsebenen an die tatsächliche Konstellation des Merkmalsraumes wird in Kapitel 3 genauer eingegangen.

3. Weiterhin gibt es Anwendungen, die eine schnelle Klassifizierung mit hoher Durchsatzrate erfordern. In Maßen ist eine Geschwindigkeitssteigerung durch Verbesserung technologischer Parameter (z.B. Taktfrequenz der eingesetzten Schaltung) zu erzielen. Eine weitere Variante zur Erhöhung der Durchsatzrate kann durch gleichzeitige Klassifizierung mehrerer Signale bewirkt werden. In Kapitel 4 wird eine schnelle Signalraumklassifizierung durch so genannte Multi-Symbol-Detektion erläutert.

4. Schließlich werden mehrere Verfahren miteinander zu einem leistungsfähigen, voll adaptiven Klassifizierer kombiniert, der in Kapitel 5 vorgestellt und analysiert wird.

Kapitel 2

Signalraumdetektion in komplexen Merkmalsräumen

Wie bereits in Abschnitt 1.2.1.3 gezeigt wurde, gibt es Reihe von Klassifizierungsproblemen, die im ursprünglichen Merkmalsraum nicht linear trennbar sind. Daher führt der Wegfall der Kernel-Funktion im Vergleich zur Support-Vektor-Maschine zu einer Einschränkung des Signalraumdetektors in Bezug auf sein Einsatzgebiet. Darüber hinaus werden bei der Signalraumdetektion nur Hyperebenen anstatt der komplexeren Hyperflächen zur Partitionierung verwendet. Somit ist die Einsatzfähigkeit von Signalraumdetektoren prinzipiell begrenzt. Um die Signalraumdetektion trotzdem für komplexe Klassifizierungsprobleme verwenden zu können, ist eine geeignete Vorverarbeitung der Daten notwendig.

Deshalb werden in diesem Kapitel Verfahren vorgestellt, die dazu beitragen, dass auch derartige Problemstellungen mittels Signalraumdetektion gelöst werden können. Außerdem wird gezeigt, wie eine geeignete Vorverarbeitung die Genauigkeit der Klassifizierung erhöhen kann, ohne die algorithmische Komplexität der Signalraumdetektion negativ zu beeinflussen.

2.1 Lineare Signalraumtransformation

Die Hauptidee der SSD ist es, den Signalraum durch Entscheidungsebenen zu partitionieren, wobei jeder Unterraum genau eine Klasse repräsentiert. In der Vergangenheit wurde bereits mehrfach gezeigt, dass die Fehlerrate einer Klassifizierung stark vom verwendeten Signalraum abhängt [SCT02, TFAS00]. So ist die bereits zitierte Verwendung so genannter Voronoi-Regionen (vgl. Abschnitt 1.2.2.3) die für den Fall des weißen Rauschens optimale Partitionierung unter dem Augenmerk einer niedrigen Fehlerrate. Im Folgenden wird gezeigt, wie eine Signalraumtransformation und gleichzeitige Umpartitionierung sich vorteilhaft auf die Detektionsgüte auswirken können, speziell unter Auftreten von farbigem Rauschen.

Außerdem wird demonstriert, wie die Signalraumtransformation zur algorithmischen Komplexitätsreduktion des Klassifizierers beiträgt.

Diese Untersuchungen werden am praktischen Beispiel des magnetischen Lesekanals einer Festplatte durchgeführt [HPG02, Com00].

2.1.1 Der magnetische Kanal

Festplatten stellen zur Zeit das preiswerteste und am weitesten verbreitete nichtflüchtige Massenspeichermedium dar [DIS99]. Sie werden in großem Umfang auch in mobilen Geräten eingesetzt, z.B. in tragbaren Computern, PDAs und Kameras.

Der auf Festplatten bei der Speicherung und Rückgewinnung der Daten auftretende Schreib- und Leseprozess kann als herkömmliches Übertragungssystem aufgefasst werden. Die zu speichernde Informationssequenz besteht aus den Symbolen (+1) und (-1), die der Magnetisierungsrichtung auf der Festplatte entsprechen. Der Abstand zwischen zwei Wechseln der Magnetisierungsrichtung ist als so genanntes Bitintervall T gegeben. Die tatsächliche Breite von T hängt von den technischen Spezifikationen der Magnetplatte ab [GT94].

Für Festplatten ist am Ausgang des Klassifizierers eine Bitfehlerrate von maximal $\rho^{BER} \leq 10^{-5}$ technologisch vorgeschrieben. Somit ist die Bitfehlerrate keine geeignete Kenngröße für den Detektor im magnetischen Kanal. Statt dessen ist die so genannte Performance ein verbreitetes Maß, um die Güte dieser Klassifizierer zu bestimmen. Sie gibt an, welcher Signal-Rausch-Abstand (SNR) in Dezibel (dB) erforderlich ist, um unter gegebenen Randbedingungen die erforderliche Bitfehlerrate auf dem magnetischen Kanal zu erzielen.

Der magnetische Kanal kann als linearer Übertragungskanal modelliert werden, der durch AWGN gestört ist. Einen zusätzlich störenden Einfluss haben das so genannte Transitionsrauschen (Jitter) und die Pulsweitenmodulation [BC93, BMN83, BGM85]. Wie in Abb. 2.1 dargestellt, setzt sich die Sprungantwort des magnetischen Kanals aus der linearen Überlagerung zweier so genannter Lorentzpulse zusammen [Ber94].

Die zeitliche Skalierung der Impulsantwort ergibt sich aus physikalischen Parametern der Festplatte, insbesondere aus der Datendichte. Ein Maß für die Da-

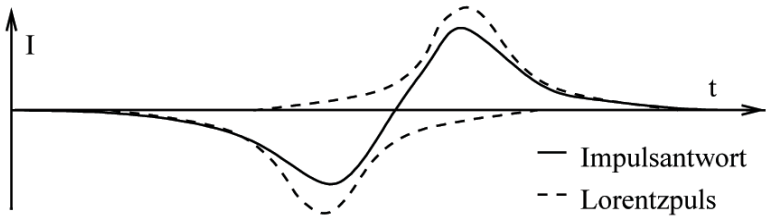

Abbildung 2.1: Darstellung des Zusammenhangs zwischen Lorentzpuls und Impulsantwort des magnetischen Kanals.

tendichte ist die Pulsbreite PW_{50}. Sie gibt an, wie viele Bitintervalle der Puls breit ist, bis seine Amplitude auf 50% des Maximums abgefallen ist. Aktuelle Festplatten arbeiten bei Pulsbreiten von etwa $3,0$ bis $4,5T$ [Gro03]. Das Lesesignal eines Festplattenlesekanals leidet also unter starker Intersymbolinterferenz (ISI).Deshalb ist die Rekonstruktion der Daten aus dem Lesesignal keinesfalls trivial. Vielmehr handelt es sich hierbei um ein komplexes Klassifizierungsproblem, das unter Verwendung der Signalraumdetektion gelöst werden kann [Str00]. Hierzu wird das Signal zunächst abgetastet und vom so genannten Equalizer gefiltert. Dann werden M aufeinander folgende Werte, die so genannten Samples, als Merkmalsvektor der SSD verwendet und klassifiziert. Bei der Signalraumdetektion im Festplattenlesekanal wird eine Kombination aus zwei FIR-Filtern zur Signalverarbeitung eingesetzt [Str00, BM97]: Das vom Lesekopf erfasste Signal wird zur Rauschunterdrückung tiefpassgefiltert und quantisiert. Ein Equalizer-Filter formt die lange, abgetastete Impulsantwort in ein kürzeres Signal, das Target, um. Durch die Filterung wird jedoch auch das dominante elektrische weiße Rauschen gefärbt.

Während das so genannte Forward-Filter das Lesesignal $\vec{s}^{fwd} = (s_0^{fwd}, s_1^{fwd}, \dots s_{(W-1)}^{fwd})^T$ verarbeitet, filtert das Feedback-Filter die detektierte Symbolfolge $\vec{\varepsilon} = \vec{s}^{fbk} = (s_0^{fbk}, s_1^{fbk}, \dots s_{(B-1)}^{fbk})^T$. Hierbei ist W die Anzahl der Filtertaps des Forward Filters und B die des Feedback Filters. Die Ausgänge beider Filter werden addiert; die Impulsantwort des Gesamtsystems wird Target genannt. Wie von *Barret Brickner* und *Jaekyun Moon* gezeigt wurde, ist für Festplattenlesekanäle die Verwendung eines 110-Targets besonders günstig in Bezug auf die Signalraumkonstellation [BM97, BM96], denn sie schlägt sich in einer geringen Schaltungskomplexität nieder [DSS00].

Je nach Auslegung dieser Vorfilter ändern sich die Position der Referenzpunkte $\vec{r}^i$ sowie die Form der sie umgebenden Cluster c^i (vgl. Abbildung 1.10). Daher müssen Vorfilter und Ebenenkonstellation des nachgeschalteten Signalraumdetektors aufeinander abgestimmt werden. Hierzu sind erforderlich die genaue Kenntnis der Rauschfärbung durch die Eingangsfilter und die Implementierung exakter Filterbausteine.

Hardware-Implementierungen sehr genauer Filterbausteine erfordern eine große Chipfläche und bedingen dadurch eine hohen Ausschussquote [Str02]. Insbesondere bei der Implementierung in Analogtechnik stellt die technologisch bedingte Parametervariation und die damit zusammenhängende garantierbare Genauigkeit der Schaltung ein Problem dar [MI92, LHC86]. Es ist also von Interesse, vor der Implementierung des vorgestellten Detektionsverfahrens darüber informiert zu sein, welche Folgen diese Parametervariation auf die Funktionalität des Verfahrens hat.

2.1.2 Einfluss von Parametervariation

Daher wurde der Zusammenhang zwischen Parametervariation der Bauteile einerseits und der Fehlerrate eines Signalraumdetektors andererseits mit Hilfe einer Computersimulation untersucht [DSS00]. Zu diesem Zweck wurde das am Lehrstuhl für Datenverarbeitungssysteme der Universität Dortmund entwickelte Lesekanal-Simulationstool HdSim verwendet [DPS01].

Ausgangspunkt waren ein auf die Pulsweite $PW_{50} = 3,0T$ abgestimmter WSSD-Detektor sowie ein zugehöriger Vorfilter mit 110-Target. Nun wurden bei 500 WSSD-Detektoren sowohl die Konstanten der im Filter enthaltenen Multiplizierer als auch die Schwellwertentscheider mit AWGN gestört. Die Varianz des anliegenden Parameterrauschens wurde im Bereich von $0 \leq \sigma^2 \leq 0,2$ variiert. Um die Ergebnisse beurteilen zu können, wurden sie zu einem unverrauschten Vergleichsdetektor in Bezug gesetzt. Hierbei fand der so genannte EPR4-Viterbi-Detektor Verwendung. Das ist der zum Zeitpunkt der Analyse am häufigsten in Festplatten verbaute Klassifizierer [FHK+98, HHS+98]. Es ergaben sich die folgenden, teilweise in Abbildung 2.2 visualisierten Ergebnisse:

1. Die Signalraumdetektion ist invariant in Bezug auf multiplikativ überlagerte Parametervariationen, die betrags- und richtungsgleich sind. Werden also alle Parameter mit dem gleichen, zufällig erzeugten Störwert multipliziert, so hat das keinen Einfluss auf die Performance des SSD.

2. Stört man die Parameter mit AWGN einer Varianz von $\sigma^2 \leq 0,03$, so geht dieses Parameterrauschen im Kanalrauschen unter. Eine messbare Verschlechterung der Performance ist bei weniger als 25% Prozent der untersuchten Detektoren feststellbar und liegt dort bei unter $0,1dB$.

3. Bis zu einer Varianz von $\sigma^2 \leq 0,10$ erlauben 95% der betrachteten WSSD-Detektoren trotz Parametervariation eine bessere Performance als vergleichbare PRML-Detektoren vom Typ „EPR4-Viterbi“.

4. Bis zu einer Varianz von $\sigma^2 \leq 0,15$ erlauben 75% der betrachteten WSSD-Detektoren trotz Parametervariation eine bessere Performance als vergleichbare PRML-Detektoren vom Typ „EPR4-Viterbi“.

5. Die Performance eines EPR4-Viterbi-Detektors wird erst bei einer Parametervarianz von $\sigma^2 \geq 0,20$ im Mittel unterschritten.

Das nachfolgende Diagramm 2.2 zeigt den Performanceverlust des Detektors (SNR in dB zum Erzielen von $\rho^{BER} \leq 10^{-5}$) in Abhängigkeit von der Stärke des Parameterrauschens bei einer Pulsweite von $3,0T$. Es wird dargestellte, welcher maximale SNR-Verlust in der Simulation messbar wird, wenn die jeweils 5% bzw. 25% schlechtesten aller Detektoren in Bezug auf ihre Fehlerrate als „Ausschuss“ unberücksichtigt bleiben. Bezugsgröße ist ein SNR von $26,8db$ unter dem

der WSSD die Bitfehlerrate von 10^{-5} erreicht. Die Kennlinie des EPR4-Viterbi-Detektors zeigt, dass dieser – unverrauscht – im vorgegebenen Signalraum eine um 0,7dB schlechtere Performance als der WSSD-Detektor aufweist (nämlich $27,5dB$).

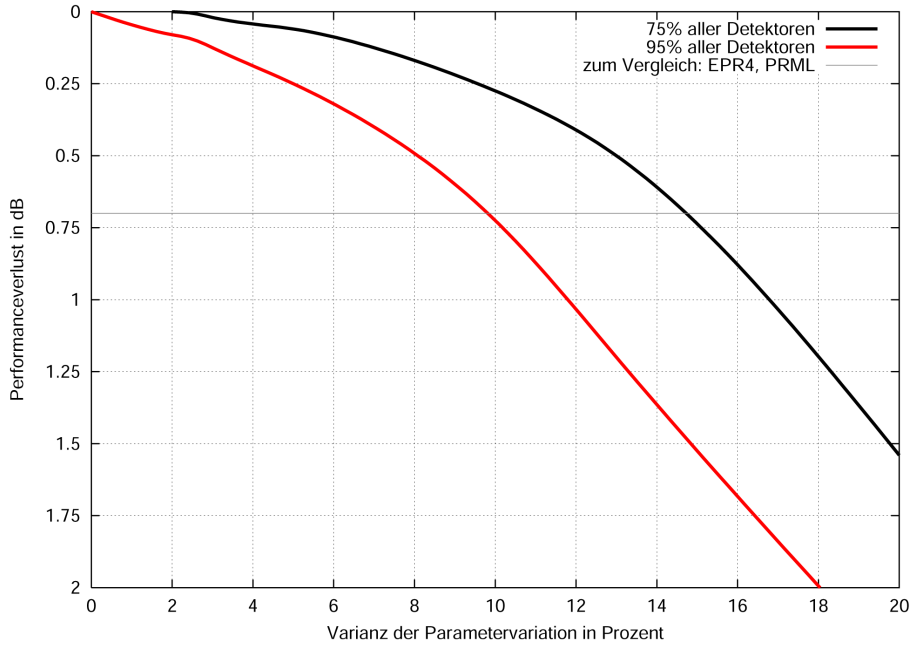

Abbildung 2.2: Verschlechterung des Signal-Rausch-Abstandes (SNR) in Abhängigkeit von der Varianz des Parameterrauschens. Dargestellt sind jeweils die unteren Schranken.

Basierend auf diesen Untersuchungen wurde gezeigt, dass sich der Signalraumdetektor aufgrund seiner Robustheit gegenüber Parametervariationen für eine Implementierung in Analogtechnik eignet [DSS00]. Der Einfluss von Bauteilvariation und Parameterrauschen schlägt sich trotz der Robustheit des Signalraumdetektors dennoch spürbar in einer Verschlechterung der Detektionsgüte nieder.

Das gemessene Verhalten ist die logische Konsequenz daraus, dass der untersuchte WSSD-Detektor besonders viele parallele Entscheidungsebenen aufweist [SHS00]. Parallele Entscheidungsebenen sind durch identische Normalenvektoren $\vec{w}^{(a,b)}$ und verschiedene Abstände zum Ursprung gekennzeichnet. Bei Parametervariation der Schaltungselemente bleibt die Parallelität der Ebenen erhalten, da identische Normalenvektoren nur einmal implementiert worden sind (siehe hierzu Seite 31, Abbildung 1.12, Mitte). Dies begründet einerseits die Robustheit der Schaltung gegen diese Störgröße, andererseits stellt der einzelne Normalenvektor einen so genannten „single-point-of-failure“ dar: Die Wahrscheinlichkeit, dass irgendeine der Ebenen optimal bezüglich der größten Detektionsgenau-

igkeit verläuft, entspricht nun der Wahrscheinlichkeit, dass alle Ebenen optimal verlaufen.

Eine Alternative zur aufgezeigten Verschlechterung des SNR bei Auftreten von Parametervariation ist die Verwendung adaptiver Vorfilter, die dazu beitragen können, die für nachfolgende Schaltungsteile erforderlichen hohen Ansprüche an die Genauigkeit der Filterung trotz Parametervariation zu befriedigen.

2.1.3 Einsatz adaptiver Vorfilter

Ein besonders genauer Vorfilter soll somit dazu beitragen, die Fehlerrate des Gesamtsystems zu reduzieren bzw. die Performance des Detektors zu verbessern.Die Vorgabe eines genau einzuhaltenden Targets impliziert dabei eine Wahl der Filterkonstanten, die abhängig sind vom magnetischen Kanal (und hier insbesondere von der Pulsbreite PW_{50}). Hierzu können entweder für jeden Kanal verschiedene Filter implementiert werden, oder man setzt einen adaptiven Equalizer ein, der zudem eine Kompensation von Bauteilungenauigkeiten und mithin eine reduzierte Fehlerrate ermöglicht. Die Verwendung solcher Filter wurde für andere Detektoren bereits diskutiert [CPO00, KM97, NM94].

In Analogie zu der dort vorgeschlagenen Vorgehensweise wurde ein Equalizer konzipiert, der auf der Struktur eines künstlichen Neurons basiert. Hierzu wurden sowohl Forward- als auch Feedback-Filter durch ein gemeinsames künstliches Perzeptron ersetzt [DS03]. Dessen Gewichtsvektor $\vec{w}$ der Dimension $M = (W + B)$ wird mit dem transponierten Eingangsvektor $(s_0^{fwd}, s_1^{fwd}, \ldots, s_W^{fwd}, s_0^{fbk}, \ldots s_{(B-1)}^{fbk})^T$ multipliziert (siehe Abb. 2.3). Das entstehende Skalarprodukt wird dann durch die Aktivierungsfunktion α gewichtet. Somit wird am Ausgang des Perzeptrons die Funktion 2.1 verfügbar:

$$\alpha(\vec{w}^{fwd} \cdot (\vec{s}^{fwd})^T + \vec{w}^{fbk} \cdot (\vec{s}^{fbk})^T) \tag{2.1}$$

Wenn $\alpha : x \rightarrow x$ gewählt bzw. das Perzeptron durch ein Adaline-Neuron ersetzt wird, so lässt sich – vorausgesetzt, die Eingangsdaten werden in miteinander verketteten Schieberegistern gespeichert – ein konventionelles Filter exakt nachbilden.

Eine Anpassung dieses Equalizers an den Kanal kann z.B. durch die Perzeptronen-Lernregel erreicht werden [Hay94]. Die in anderen Publikationen vorgeschlagenen Lernverfahren, die das Adaptionsverhalten adaptiver Vorfilter verbessern können, sind ebenfalls zum Einsatz geeignet [NM94, KM97, CPO00]. Wie jedoch im nächsten Abschnitt, in dem die erzielten Ergebnisse vorgestellt werden, gezeigt wird, genügt bereits die Anwendung der Perzeptronen-Lernregel, um die gewünschte Anpassung zu erzielen.

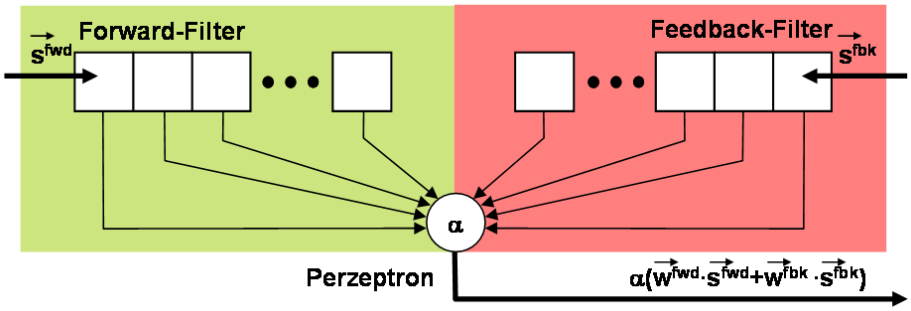

Abbildung 2.3: Durch ein Perzeptron implementierter adaptiver Equalizer. Die Pfeile symbolisieren die Eingangsgewichte des Perzeptrons.

2.1.4 Evaluierung des Systems

Der beschriebene adaptive Vorfilter kann sich, wie in Abschnitt 2.1.2 beschrieben, auf die Detektionsgenauigkeit auswirken. Einige zufällig stark verrauschte Eingangswerte könnten unter Umständen dazu führen, dass während des Trainings eine nicht optimale Ebenenkonstellation eingestellt wird. Daher wird zunächst geprüft, wie stark sich die adaptiv gefilterte Signalfolge von einer mit analytisch optimierten Filtern verarbeitete Signalfolge unterscheidet [Kor91].

Hierzu wurde das Festplattensimulationstool HdSim nacheinander mit einem konventionellen und einem adaptiven Vorfilter bestückt. Das Lesesignal von 100.000 Transitionen, den Übergängen zwischen den Bit-Zellen, wurde für beide Fälle nach der Filterung ausgelesen. Jeweils drei aufeinander folgende Werte wurden als Punkt in $\underline{\underline{R}}^3$ dargestellt und in Abbildung 2.4 visualisiert. Auf der linken Seite ist das vorverarbeitete Lesesignal für den konventionellen Filter zu sehen. Rechts daneben befindet sich ein Lesesignal, welches am Ausgang des Perzeptrons anliegt.

Form und Position der Cluster unterscheiden sich nur unwesentlich: Mit bloßem Auge ist zu beobachten, dass der beschriebene adaptive Vorfilter das Signal im Vergleich zu linearen Vorfiltern nur unwesentlich verzerrt. Da der Signalraum kaum verzerrt wird, ist zu erwarten, dass weiterverarbeitende Komponenten ohne (große) Modifizierung benutzt werden können. Um dies zu überprüfen, wurde in einem weiteren Schritt ein adaptiver Vorfilter in Kombination mit einem unangepassten Signalraumdetektor verwendet. Auf diese Weise sollte untersucht werden, wie die adaptive Vorfilterung mit den statischen Detektorkomponenten zusammenarbeitet. Zu diesem Zweck wurde die Fehlerrate des Gesamtsystems unter dem Einfluss von Störgrößen analysiert.

Dabei kam ein Signalraumdetektor zum Einsatz, der für einen magnetischen Lesekanal mit $PW_{50} = 3,0T$ ohne Jitterrauschen optimiert wurde. Der für diesen Fall optimierte WSSD-Detektor wurde einer Bauteil-Parametervariation von

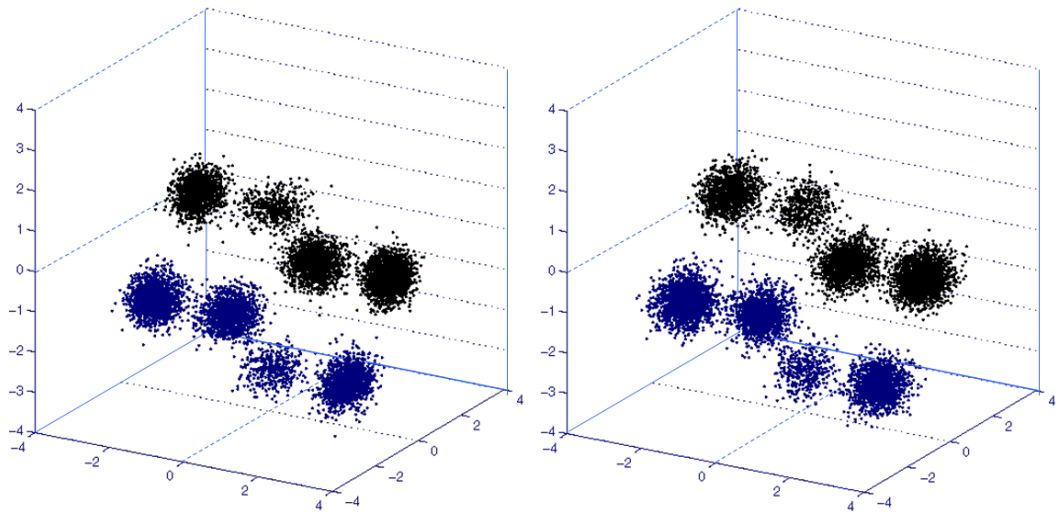

Abbildung 2.4: Verzerrung des Signalraumes durch Verwendung adaptiver Filter. Die linke Abbildung zeigt ein Abbild von 100.000 konventionell gefilterten Signalen, die rechte das Pendant beim Einsatz adaptiver Vorfilter.

$\sigma^2 = 0,05$ und einem Jitterrauschen mit $0 \leq \sigma_J^2 \leq 0,08$ unterzogen. Dann wurde der notwendige Signal-Rausch-Abstand, um eine Bitfehlerrate von $\rho^{BER} \leq 10^{-5}$ zu erzielen, gemessen und in Abbildung 2.5 dargestellt. Als Trainingssequenz kam eine Datenfolge von 50 Millionen zufälligen Transitionen zum Einsatz. Das Training kann bei einer für Festplatten üblichen Datenrate von 500 Mbit/s innerhalb weniger Sekunden abgeschlossen werden[1] [TB00].

Es wurde festgestellt, dass sich das Eingangssignal durch die Filterung mit einem adaptiven Equalizer derart vorverarbeiten lässt, dass eine Anpassung des Filters an die gegebene Parametervariation der verwendeten Bauteile möglich ist. Dies wird im linken Teil der Grafik (Jitter 0,00) deutlich, da bei nicht vorhandenem Transitionsrauschen der mit einem adaptiven Vorfilter bestückte Detektor einen niedrigeren SNR ermöglicht. Weiterhin kann der adaptive Detektor dazu beitragen, den störenden Einfluss nicht perfekter Ebenenkonstellationen zu verringern. Da der Detektor für einen jitterfreien Kanal entwickelt wurde, ist bei Einsatz eines adaptiven Vorfilters unter Einfluss von Transitionsrauschen (Jitter 0,08) eine wesentliche Verbesserung des Signal-Rausch-Abstandes im Vergleich zum Detektor mit nichtadaptivem Vorfilter messbar.

Somit lässt sich zusammenfassen, dass die Kombination aus adaptivem Vorfilter und statischem Signalraumdetektor geeignet ist, den Einfluss störender Um-

[1] In der Softwaresimulation dauert ein derartiger Simulationslauf je nach Hardwareausstattung des Testsystems zwischen 12 und 20 Minuten

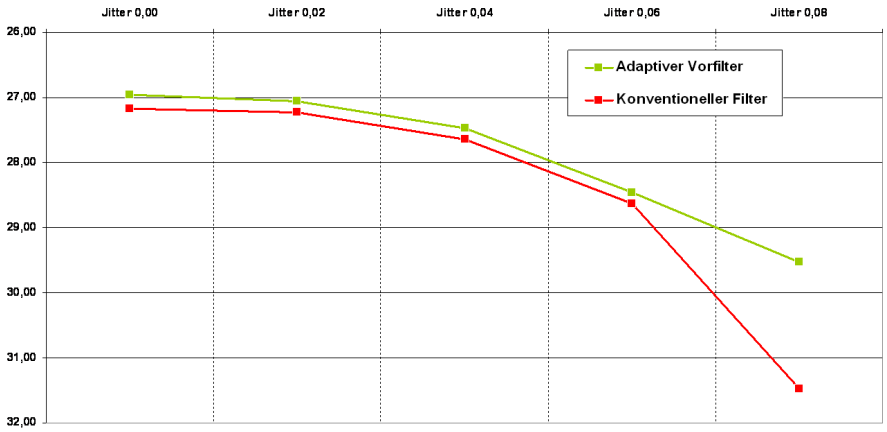

Abbildung 2.5: Nötiger Signal-Rausch-Abstand zum Erzielen einer Bitfehlerrate von 10^{-5} beim Einsatz eines adaptiven Vorfilters und eines statischen WSSD im Vergleich zu WSSD mit konventionellem Filter.

welteinflüsse zu eliminieren und somit eine Anpassung an den tatsächlichen Kanal zu erzielen.

2.2 Komplexe Signalraumtransformation

Wie bereits gezeigt wurde (vgl. Abschnitt 1.2.3.3), kann eine geeignete Signalvorverarbeitung dazu beitragen, die algorithmische Komplexität der Klassifizierung zu verringern [SHS00], da diese stark von der Form der durch die Partitionierung entstehenden Unterräume abhängt. In einer typischen Implementierung (siehe Seite 29, Abbildung 1.11) wird eine Entscheidungsebene durch M Konstantenmultiplizierer und einen Schwellwertschalter implementiert. Wenn zwei Entscheidungsebenen parallel zueinander verlaufen, benötigt man – da die Konstantenmultiplizierer der implementierten Normalenvektoren identisch sind – M Multiplizierer je paralleles Ebenenpaar weniger als bei Verwendung nichtidentischer Normalenvektoren.Dieses Verfahren wird z.B. beim WSSD eingesetzt und ist in Abbildung 1.12 (Seite 31) illustriert.

Eine weitere Variante der Komplexitätsreduktion besteht darin, die Entscheidungsebenen so zu legen, dass sie parallel zu den Einheitsvektoren des Weltkoordinatensystems verlaufen. Dies wird in Abbildung 2.6 dargestellt: Ein Signalraum $\underline{\underline{R}}^M$ wird in einen anderen Signalraum $\tau(\underline{\underline{R}}^M)$ transformiert. Ein Unterraum u^j, der die Klasse $\varepsilon_1(u^j) = k^i$ repräsentiert, wird somit in einen anderen Unterraum $\tau(u^j)$ abgebildet, der in $\underline{\underline{R}}^M$ ebenfalls k^i repräsentiert. Im ursprünglichen Fall erfolgt die Klassifizierung von $\vec{s}^j$ durch Abstandsberechnung zu den Entscheidungsebenen $E^{(a,b)}$ mittels skalarer Multiplikation $\vec{w}^{(a,b)} \cdot (\vec{s}^j)^T$. Nach der

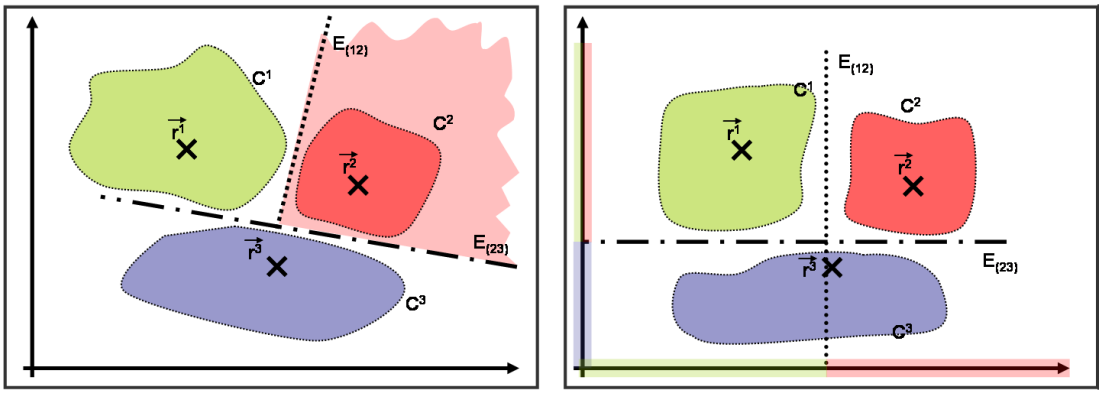

Abbildung 2.6: Komplexitätsreduktion bei der Signalraumdetektion durch Verwendung von Ebenen parallel zu den Einheitsvektoren. Links: Aufwändige Unterraumbestimmung durch 2D-Multiplikation. Rechts: Unterraumbestimmung durch Schwellwertbildung.

Transformation reduziert sich die Abstandsberechnung auf einen Vergleich nach Gleichung 2.2, da $(M-1)$ Elemente des Normalenvektors identisch mit Null sind.

$$\begin{pmatrix} 0 \\ \vdots \\ w_i \\ \vdots \\ 0 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ \vdots \\ s_i \\ \vdots \\ s_{(M-1)} \end{pmatrix} + w_M < 0 \Leftrightarrow \begin{cases} w_i > 0 & : s_i < -\frac{w_M}{w_i} \\ w_i < 0 & : s_i > -\frac{w_M}{w_i} \\ \text{sonst} & : w_M < 0 \end{cases} \tag{2.2}$$

Eine Signalraumtransformation, die den ursprünglichen Signalraum derart verändert, dass die Entscheidungsebenen achsenparallele Normalenvektoren besitzen, ist sinnvoll, da für jede solchermaßen transformierte Ebene M Multiplikationen eingespart werden können.

Hierzu ist eine geeignete Transformationsvorschrift τ erforderlich, wobei sich die Analogie zur Support-Vektor-Maschine und der so genannten Kernel-Funktion aufdrängt. Während jedoch die Transformation des Merkmalsraumes ein Grundprinzip der aus Kernel-Perzeptronen aufgebauten SVM ist, soll bei der SSD strikt zwischen Vorverarbeitung und Klassifizierung unterschieden werden. Insbesondere soll die getrennte Verarbeitung der zu klassifizierenden Daten dazu beitragen, vorhandenes Vorwissen besser in den Klassifikationsprozess zu integrieren, indem die beiden voneinander getrennten Systeme jeweils transparenter sind als ein vereinheitlichtes komplexes System.

Zum Finden einer geeigneten Transformationsvorschrift ist zu unterscheiden, ob Vorwissen über die zu klassifizierenden Objekte vorhanden ist oder nicht. Somit kommen zur Bestimmung von τ mit F Parametern grundsätzlich zwei verschiedene Varianten in Frage:

1. Verwendung einer bekannten, parametrisierbaren Transformationsvorschrift $\tau(\Phi_0, \Phi_1, \cdots \Phi_{(F-1)}, \underline{\underline{R}}^M)$. Voraussetzung ist ein bestehendes Modell oder Vorwissen, aus dem sich diese Transformation ergibt.

2. Transformation ohne Berücksichtigung von (vorhandenem) Vorwissen mittels $\tau := \tau_0(\Phi_0, \tau_1(\Phi_1, \tau_2(...\tau_{(F-1)}(\Phi_{(F-1)}, \underline{\underline{R}}^M))))$ als allgemeine Formulierung der Verknüpfung mehrerer parametrisierbarer Funktionen.

Zur Bestimmung von τ muss im Fall (1) der Parametervektor $\vec{\Phi}$ optimiert werden. Im zweiten Fall lässt sich $\Phi_{(i+F)}$ als Index betrachten, der dazu dient, die mit Φ_i parametrisierte Funktion aus einer Menge von möglichen Funktionen zu wählen, aus denen sich τ zusammensetzt. Dadurch wird die Dimension des Suchraumes für $\vec{\Phi}$ verdoppelt. Als bessere Alternative bietet sich hier der SVM-Einsatz an.

Ein Standardweg zur Optimierung von derartigen Parametervektoren, um ein spezifisches Problem zu lösen, ist der Einsatz so genannter evolutionärer Algorithmen (EA) [Koz92, BNKF98, Ang94]. Dieses aus der Natur bekannte Verfahren basiert auf den Prinzipien von Mutation, Vererbung und Selektion:

Ein (zufällig gewählter) Parametervektor $\langle\vec{\Phi}\rangle_t$ werde als Ausgangspunkt betrachtet. Seine Komponenten $\langle\Phi_i\rangle_t$ werden nun mehr oder weniger stark verändert. Die Art der Veränderung, die z.B. zufällig als Mutation oder basierend auf anderen Parametervektoren als Vererbung erfolgen kann, wird Evolutionsstrategie genannt. Eine sehr einfache Evolutionsstrategie ist die „1+1 Evolutionsstrategie“ (11ES) [Rec94].

Im Rahmen der 11ES wird $\langle\vec{\Phi}\rangle_t$ zuerst mit Zufallszahlen initialisiert. Dann wird ein weiterer Vektor $\langle\vec{\Phi}\rangle_{(t+1)}$ dadurch erstellt, indem ein zufällig bestimmter Vektor $\vec{\eta} = (\eta_0, \eta_1 \ldots, \eta_{(M-1)})^T$ zu $\langle\vec{\Phi}\rangle_t$ addiert wird, wobei alle η_i gaussförmig verteilte Zufallszahlen sind.

Nach der Addition wird die Qualität des neu erstellten Parametersatzes ermittelt. Zu diesem Zweck wird eine so genannte Fitnessfunktion ς benötigt. Diese muss in der Lage sein zu messen, wie gut ein vorliegender Parametersatz die vorgegebene Aufgabenstellung erfüllt. Diese Fitness wird mit der Fitness des vorhergehenden Parametersatzes verglichen. Hat sich die Fitness dabei verschlechtert bzw. gilt $\varsigma(\langle\vec{\Phi}\rangle_t) < \varsigma(\langle\vec{\Phi}\rangle_{(t-1)})$, so wird das neu erstellte Individuum verworfen. Ansonsten wird es als Ausgangspunkt für den nächsten Evolutionsschritt verwendet.

Die Schwierigkeit beim Einsatz von EA besteht im Finden einer geeigneten Fitnessfunktion. Für einfache Probleme ist das Implementieren der Fitnessfunktion oftmals aufwändiger, als das Problem analytisch zu lösen. Im vorliegenden Fall verhält sich das anders. Ziel der Optimierung der Parameter der Transformationsvorschrift τ ist es, die Fehlerrate der Klassifizierung zu senken. Diese lässt sich – wenngleich u.U. zeitlich aufwändig zu berechnen – mathematisch sehr einfach mit Gleichung 1.23 angeben.

Die Fitnessbestimmung kann an spezifische Nebenbedingungen angepasst werden, indem Objekte, die besonders relevante Klassen repräsentieren, überproportional oft im Trainingsdatensatz vertreten sind [Bre93]. Das Erstellen eines relevanten Trainingsdatensatzes ist ein eigenes, sehr komplexes Problem jenseits der Aufgabenstellung dieser Arbeit, und es wird deshalb darauf nicht weiter eingegangen.

2.2.1 Implementierung

Ein anschauliches Beispiel für einen Merkmalsraum, der mit achsenparallelen Entscheidungsebenen nicht sinnvoll partitioniert werden kann, ist der so genannte RGB-Farbraum, der dazu dient, Farbpunkte aufgrund ihrer Eigenschaften geografisch anzuordnen. Hierzu wird jede Farbe in ihre spektralen Bestandteile rot, grün und blau zerlegt. Die jeweilige Intensität der Spektralanteile wird gemessen und (auf Eins normalisiert) als Merkmal eines dreidimensionalen Raumes aufgefasst, so dass jede Farbe als Punkt im RGB-Raum dargestellt werden kann [Poy96].

Beispielhaft sind in Abb. 2.7 alle Bildpunkte aus dem Foto von grünen und roten Tomaten in den RGB-Farbraum eingetragen (links unten). Dieser ist aus Perspektive der RG-Ebene in Blickrichtung der B-Achse aufgetragen. Wie man sieht, ist keine Partitionierung in Unterräume, die rote und grüne Tomaten mittels achsenparalleler Entscheidungsebenen unterscheiden möglich. Abhilfe schafft z.B. die Umwandlung in den so genannten YUV-Farbraum. Neben dem Y-Wert, der die Helligkeit des Bildpunktes repräsentiert, wird die Farbe in den zwei Dimensionen U und V kodiert. Eine Partitionierung in einen roten und grünen Unterraum ist für das Beispielbild mit achsenparallelen Entscheidungsebenen möglich (Abb. 2.7 rechts unten).

Je mehr Farben unterschieden werden müssen, desto aufwändiger und komplexer wird die zugehörige Ebenenkonstellation. Ein Anwendungsfeld, in dem zwischen sehr vielen verschiedenen Farben präzise unterschieden werden muss, ist der so genannte Roboterfußball [KAK+97, KTS+97, AVT+99]. Dort orientieren sich Roboter mit Hilfe einer Kamera aufgrund einzigartiger farbiger Markierungen, so genannter Landmarken, auf einem miniaturisierten Fußballfeld (vgl. Abb. 2.8) [SK94, JCB01].

Die Systeme in der so genannten SONY Legged League[2] werden dabei vor besonders hohe Herausforderungen gestellt: Einerseits sind die von der Kamera gelieferten Bilder von minderer Qualität, andererseits verfügen die Roboter nur über eine vergleichsweise geringe Rechenleistung [Son03, VUF+98, SDO+04]. Eine Hardwaremodifizierung ist gemäß den geltenden Regeln untersagt [Röf05]. Eine präzise Klassifizierung der Objekte im Kamerabild mit herkömmlichen Verfahren ist daher nicht möglich [Que00, BBV00]. Gleichzeitig bedarf es jedoch dringend

[2]Sony Aibo der Typen ERS 7, 210, 210-Supercore und 2100

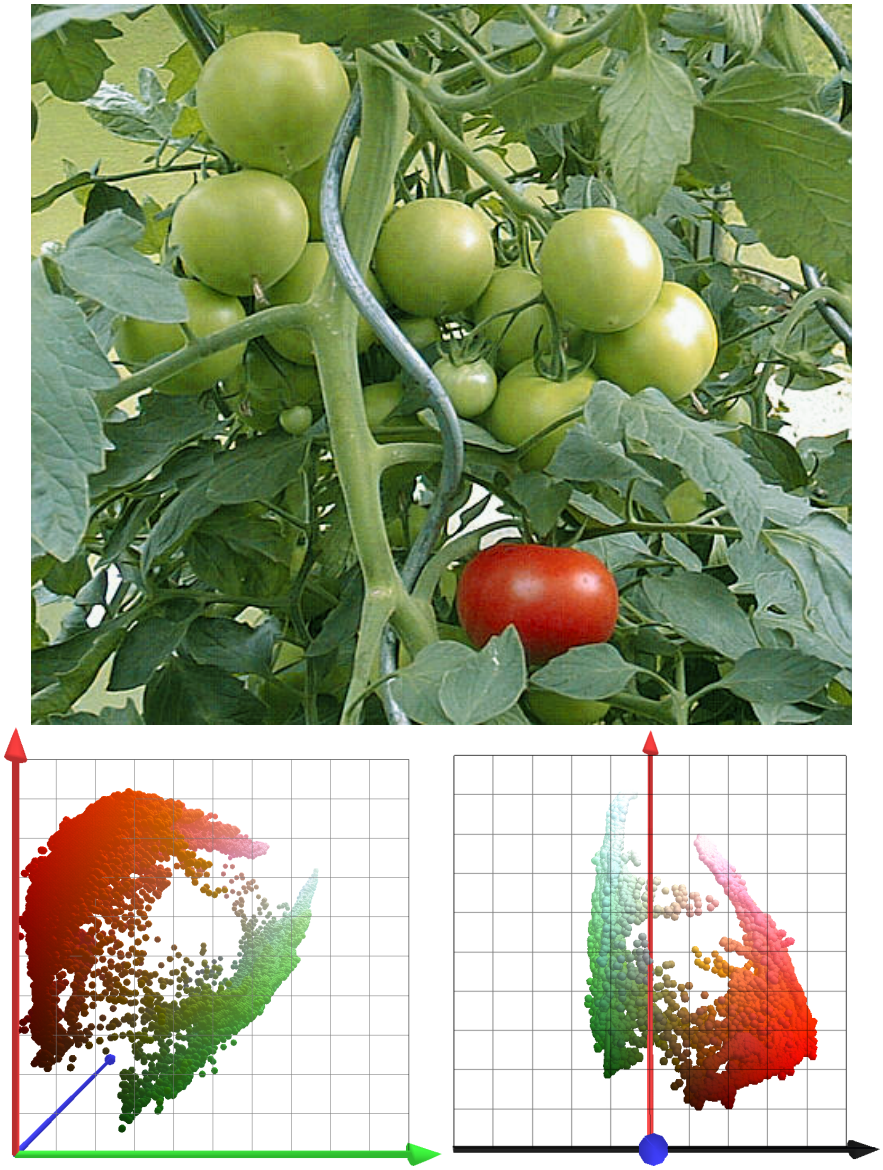

Abbildung 2.7: Anordnung der Farbwerte des mittleren Fotos im RGB- und YUV-Farbraum. Blick auf RG-Ebene (links) und YV-Ebene (rechts).

einer Lösung dieses Klassifikationsproblems, da die Roboter-Navigation auf dem Spielfeld maßgeblich von den Sensordaten abhängig ist [WAD+00, BC00].

Die beim Roboterfußball verwendeten Farben sind auch im YUV-Farbraum nicht mit achsenparallelen Entscheidungsebenen klassifizierbar. Um dies zu verdeutlichen, wurden in Abbildung 2.9 alle Farbpunkte aus Bild 2.8 in einem RGB- und in einem YUV-Farbraum dargestellt. Man stellt fest, dass die einzelnen Farben kegelförmig um den Ursprung der jeweiligen Koordinatensysteme angeordnet

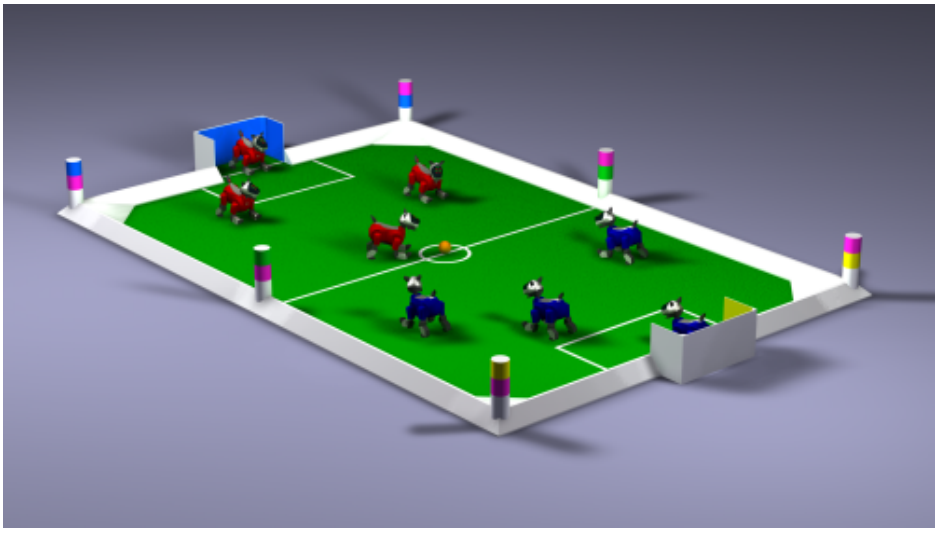

Abbildung 2.8: Turnierfeld in der SONY Four Legged League mit den zugehörigen Toren und Landmarken.

sind. Zwar liegen ähnliche Farben in beiden Merkmalsräumen dicht beieinander, jedoch eine Partitionierung mittels achsenparalleler Ebenen ist weder im RGB- noch im YUV-Raum möglich.

Für eine einfache Klassifizierung wäre es hingegen zweckmäßig, wenn sich zumindest die für den Roboterfußball relevanten Farben durch achsparallele Entscheidungsebenenen klassifizieren ließen. Dies kann dadurch erzielt werden, indem man den Farbraum der Kamera in einen anderen Farbraum konvertiert, in dem es dann möglich ist, die Farbinformationen mit wenig Rechenleistung und hoher Genauigkeit zu extrahieren [DDHO03]. Dabei gilt es eine Reihe von Nebenbedingungen zu beachten:

1. Die Ebenenkonstellation muss robust gegen Helligkeitsänderungen sein: Schattenwürfe und Blitzlicht sind permanente Störfaktoren beim Roboterfußball. Konsequenterweise veranstaltet die RoboCup Federation seit 2004 eine so genannte „Technical Challenge“, bei der Roboter unter wechselnden Beleuchtungsverhältnissen bestimmte vorgegebene Aufgaben lösen müssen.

2. Ausgangpunkt der Transformation soll der YUV-Farbraum sein, da die im Roboter eingebaute Kamera die Bilddaten bereits im YUV-Format ausgibt.

3. Die Transformationsvorschrift muss algorithmisch möglichst einfach, intuitiv und wenig rechenaufwändig sein.

Wenn der Signalraum robust gegen Helligkeitsschwankungen sein soll, so ist es sinnvoll, ihn genau so aufzubauen, dass eine seiner Dimensionen die Helligkeit selbst ist. Diese Dimension kann dann in der späteren Verarbeitung ignoriert

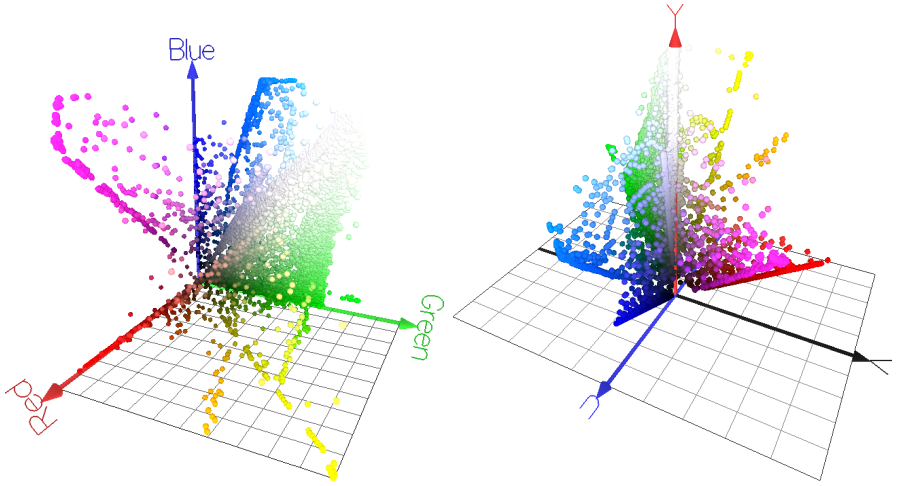

Abbildung 2.9: Darstellung der farbigen Bildpunkte aus Abb. 2.8 im RGB- (links) und YUV-Farbraum (rechts)

werden. Bei dem vorhandenem YUV-Farbmodell liegt die Helligkeit eines Punktes als Y-Wert bereits vor. Dessen Weiterverwendung reduziert den algorithmischen Aufwand im Vergleich zu allen anderen Alternativen erheblich. Betrachtet man Abbildung 2.9 rechts, so lässt sich erkennen, dass gleiche Farben im YUV-Diagramm durch eine besondere Eigenschaft gekennzeichnet sind: Der Winkel der Verbindungsgeraden zwischen ihrer Projektion in die UV-Ebene und dem Koordinatenursprung ist für gleiche Farben identisch. Aufgabengerecht ist es daher, diesen Winkel zu bestimmen und ihn – eventuell gewichtet – als weitere Koordinate aufzunehmen. Aus Kenntnis von Y und dem beschriebenen Winkel kann unter Kentniss einer dritten Koordinate der Raumpunkt $\vec{s}$ rekonstruiert werden. Diese dritte Koordinate ist zunächst (in Grenzen) frei wählbar.

Frühere Publikationen zur effizienten Farberkennung zeigen, dass die Wahl der Länge der genannten Verbindungsgeraden zu einer besonders robusten Farbklassifizierung führt [SCT02, TFAS00]. Der in diesen Publikationen vorgestellte so genannte TSL-Farbraum erscheint daher besonders geeignet als Ausgangsbasis für die weiteren Untersuchungen, denn er verfügt über mehrere der notwendigen Eigenschaften und wurde bereits erfolgreich für eine schnelle und robuste Farbklassifizierung eingesetzt [TA00].

Zur Adaption an oben genannte Randbedingungen und an die Farbtemperatur der Umgebungsbeleuchtung wird außerdem eine Reihe von Koeffizienten Φ_i in die Transformationsvorschrift nach [TA00] eingefügt, so dass der TSL-Ansatz verallgemeinert wird. Die Transformation des YUV-Raumes erfolgt nun in zwei Schritten:

Zuerst wird YUV in einen Hilfsraum, bestehend aus den Komponenten e_1, e_2 und e_3 (gemäß Gleichung 2.3), umgewandelt.

$$e_n = \frac{\Phi_{(2n+1)} \cdot s_U + \Phi_{(2n+2)} \cdot s_V}{\Phi_0 \cdot s_Y + \Phi_1 s_U + \Phi_2 s_V} \; n \in \{1, 2, 3\} \tag{2.3}$$

Danach wird auf den Hilfsraum die parametrisierte TSL-Transformation angewendet (Gleichungen 2.4-2.5). Gemäß der Vorüberlegung bleibt die Y-Komponente erhalten, und L braucht nicht transformiert zu werden. Konsequenterweise wird der neue Farbraum „TSY“ genannt.

$$T = \frac{1}{\pi} \cdot \begin{cases} e_2 > 0 & : atan2(e_1, e_2) + \pi/4 \\ e_2 < 0 & : atan2(-e_1, -e_2) + \pi/4 \\ \text{sonst} & : 0 \end{cases} \tag{2.4}$$

$$S = \sqrt{\Phi_9 \cdot ((e_1)^2 + (e_2)^2 + (e_3)^2)} \tag{2.5}$$

Wie in Abschnitt 2.2 dargelegt, kann die Optimierung eines hochdimensionalen Parametervektors erfolgreich mit evolutionären Algorithmen durchgeführt werden. Hierzu ist die Implementierung einer geeigneten Fitnessfunktion erforderlich, wofür bereits die Verwendung der Fehlerrate des Klassifizierers vorgeschlagen wurde. Zur Bestimmung der Fehlerrate ist die Kenntnis der korrekten Klasse $\widehat{k} = \widehat{\varepsilon}(\vec{s})$ erforderlich. Diese Bestimmung kann mit Hilfe der manuellen Klassifizierung „von Hand“ vor Durchführung der Evolution erfolgen. Zu diesem Zweck wurde ein Satz an Trainingsdaten zur Verfügung gestellt und händisch vorklassifiziert [DDHO03]. Basierend auf diesem Testdatensatz wurde mittels 11ES ein neuartiger angepasster TSY-Farbraum evolviert.

Abbildung 2.10 stellt exemplarisch einen auf diese Weise optimierten Farbraum (links) einem herkömmlichen YUV-Farbraum (rechts) gegenüber. Es ist offensichtlich, dass das Entwurfsziel erreicht wurde und der TSY-Farbraum durch achsenparallele Entscheidungsebenen so partitioniert werden kann, dass die relevanten Roboterfußball-Farben korrekt klassifiziert werden können.

2.2.2 Simulationsergebnisse

Während die prinzipielle Wirksamkeit des vorgeschlagenen Transformationsverfahrens im Abschnitt 2.2.1 illustriert wurde, soll nun die Effizienz des Transformationsverfahrens analysiert werden.

Unter Anwendung der 11ES (mit dem parametrisierbaren TSY-Farbraum als Startpunkt) wird eine Fitness von mehr als 99% (Fehlerrate $\leq$ 0,01) bereits nach 10 Evolutionsschritten erreicht. Eine Fitness von 99,85% wird für den Testdatensatz erreicht, wenn man die Transformationsvorschrift nach Gleichung 2.3-2.5 und die Koeffizienten aus Tabelle 2.1 verwendet.

Wie in Abbildung 2.11 dargestellt, befindet sich die Fitness der Farbklassifizierung schon nach einigen wenigen Evolutionsschritten auf einem hohen Niveau

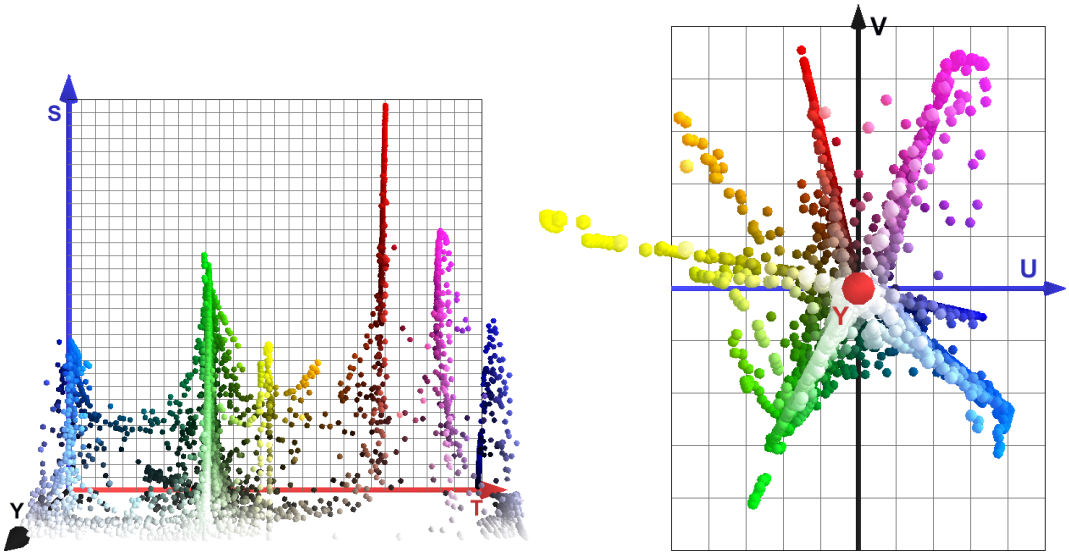

Abbildung 2.10: Ansicht von TSY- und YUV-Farbraum, bezogen auf die Farbpunkte der Abbildung 2.8 mit Blick entlang der Y-Achse.

YUV → TSY Transformation			
$\Phi_0 =$ 4.253156	$\Phi_3 =$ -0.051191	$\Phi_5 =$ 0.004342	$\Phi_7 =$ 0.133934
$\Phi_1 =$ 3.721591	$\Phi_4 =$ 0.186359	$\Phi_6 =$ 0.040280	$\Phi_8 =$ 0.162277
$\Phi_2 =$ 4.260.283			$\Phi_9 =$ 2.603124

Tabelle 2.1: Evolvierter Parametersatz zum Erzielen einer Fitness von 99,85%.

($\varsigma > 0,995$). Ein Wert von Eins bedeutet dabei, dass alle Pixel korrekt klassifiziert wurden. Nach etwa 10.000 weiteren Evolutionsschritten verbessert sich die Fitness nur geringfügig auf 99.85%.

Die Entwicklung der Fitness im zeitlichen Verlauf für 100 Experimente spiegelt Abbildung 2.11 wieder. Dort sind in zwei Kurven die jeweils höchsten Fitnesswerte und die mittlere Fitness für verschiedene Zeitpunkte der Evolution aufgetragen. Zusätzlich ist die Varianz der Abweichung zwischen den einzelnen Experimenten eingetragen. Im zeitlichen Verlauf steigt die Fitness zunächst stark, dann langsamer (aber monoton) mit der Dauer der Evolution an. Der Betrag des Änderungsvektors $||\langle\vec{\Phi}\rangle_t - \langle\vec{\Phi}\rangle_{(t+1)}||$ nimmt dabei ebenfalls ab, was auf ein globales Optimum für die im Experiment verwendeten Testbilder hinweist. Dies wird auch an Abbildung 2.12 deutlich. Aus dem Diagramm kann entnommen werden, dass die Veränderung des Parametervektors (Mutationsstärke) parallel zur Fitness mit der Zeit sinkt. Somit wird die gegen Ende des Experimentes erziel-

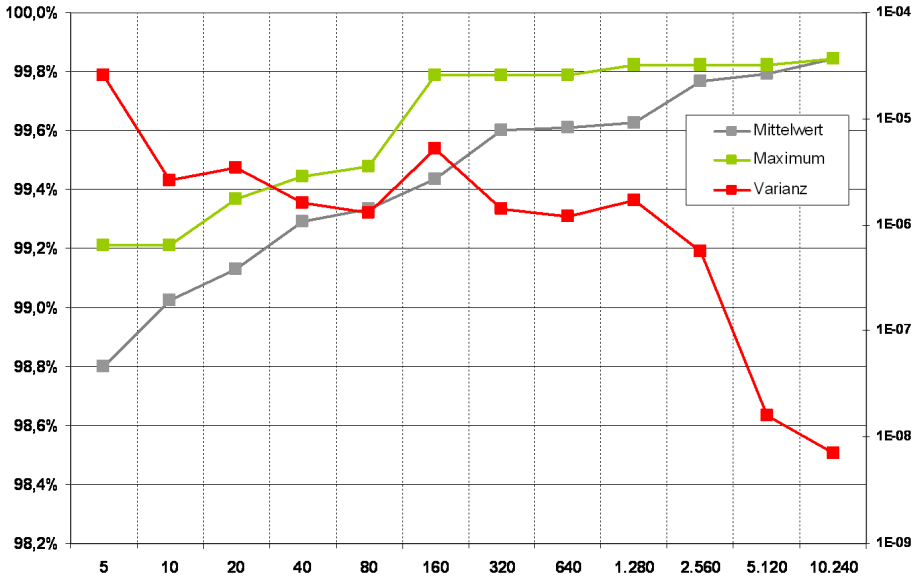

Abbildung 2.11: Mittelwert und Maximum der Fitness während der Evolution über der Anzahl der Evolutionsschritte, dazu die Varianz zwischen den einzelnen Evolutionsläufen in logarithmischer Darstellung.

te leichte Verbesserung „nur“ durch eine kleine Änderung des Parametervektors erreicht.

Nach der Portierung der Farbraumtransformation auf die Roboter wurde eine deutlich verbesserte Farberkennung festgestellt gegenüber der bisher verwendeten, auf dem YUV-Farbraum basierenden Farbklassifizierung [JLB+03]. Dies ist in Abbildung 2.13 dargestellt. Man sieht drei von der internen Roboterkamera aufgenommene Bilder, die anschließend einmal mit dem konventionellen YUV-Ansatz und einmal mit dem Schwellwert-Ansatz nach erfolgter TSY-Transformation farbklassifiziert wurden.

Aus Abbildung 2.13 wird insbesondere die Robustheit des evolvierten Farbraumes gegenüber Helligkeitsschwankungen deutlich [DHN04]. Beispielsweise ist im rechten Bild im Hintergrund eine weiße Wand mit starken Schattierungen zu erkennen. Diese Wand wird im TSY-Farbraum als zusammenhängende Fläche erkannt, während im YUV-basierten Farbraum nur ein kleiner Teilbereich der Wand erkannt wird. Die Farbe des Bodens (Grün) ist im YUV-Farbraum-segmentierten Bild an einigen Stellen von schwarzen Punkten durchbrochen. Diese kennzeichnen Stellen, an denen die Farbe aufgrund der niedrigen Helligkeit nicht mehr eindeutig erkannt wurde. Im TSY-Farbraum hingegen sind auch die dunklen Stellen des Bodens grün markiert. Die Bodenfläche ist nur in unmittelbarer Nähe von Linien nicht durchgängig. Ein vergleichbares Ergebnis ist sowohl am Körper des blauen Roboters (im linken Bild) sowie innerhalb seines Schattens erkennbar.

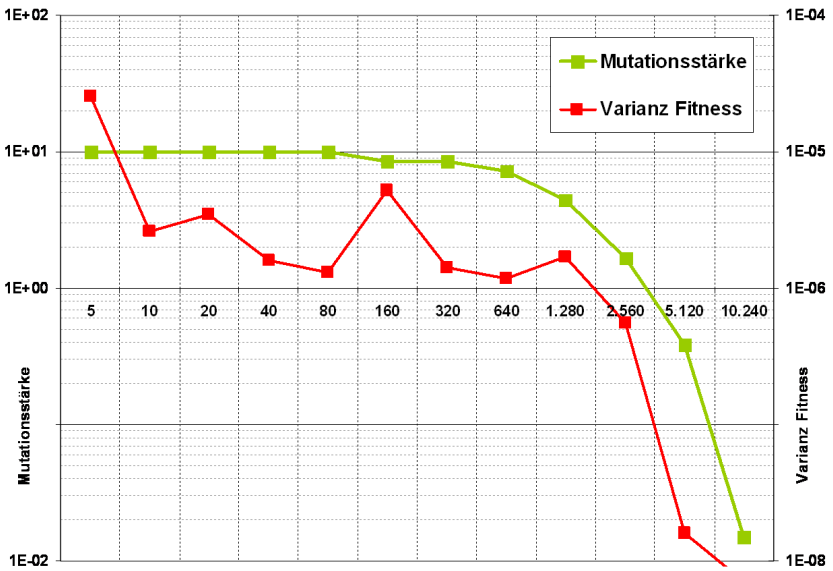

Abbildung 2.12: Mittlere Varianz der Mutationsstärke während der Evolution über der Anzahl der Evolutionsschritte in logarithmischer Darstellung.

Die durchgeführte Signalraumtransformation führt also nicht nur zu einer vereinfachten Konstellation der Entscheidungsebenen, sondern gleichzeitig wird die Fehlerrate der Klassifizierung sichtbar verbessert.

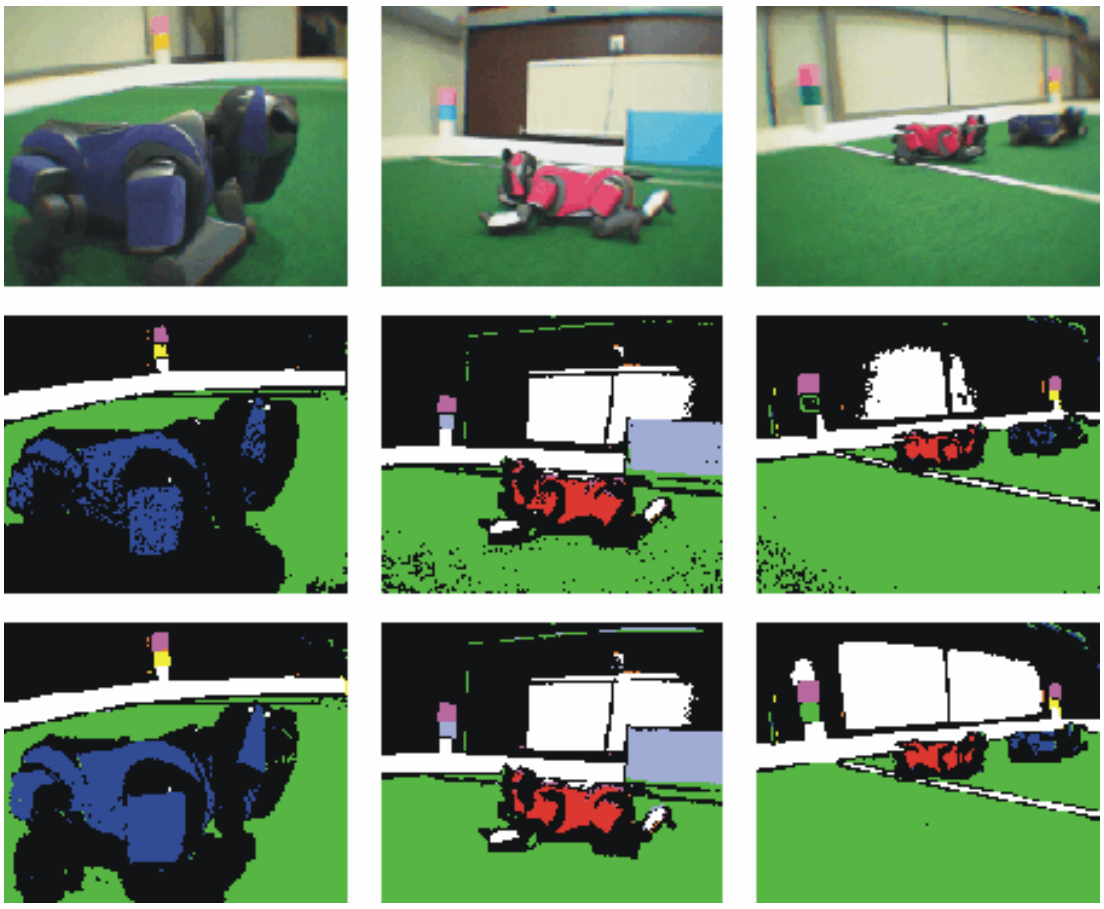

Abbildung 2.13: Von Roboterkamera aufgezeichnete Referenzbilder (oben) und deren zugehörige farbsegmentierte Bilder auf Basis des YUV-Farbraumes (Mitte) und auf Basis des TSY-Farbraums (unten) [RDD+04].

Kapitel 3

Adaptive Signalraumdetektion

Wie in Kapitel 2 gezeigt wurde, kann gemäß Abschnitt 2.1 eine Veränderung von Signalvorverarbeitung, Signalqualität oder Kanalparametern zu einer Modifizierung der Signalraumrepräsentation der zu klassifizierenden Objekte führen. Das heißt im Umkehrschluss, dass die Entscheidungsebenen dann nicht mehr optimal auf die Randbedingungen abgestimmt sind. Dem kann Rechnung getragen werden durch Anpassung der Entscheidungsebenen der ursprünglich berechneten Signalraumkonstellation an diegenigen des veränderten Signalraumes. Ohne Anpassung des Verfahrens an die neu geltenden tatsächlichen Randbedingungen und deren variable Parameter muss unter Umständen eine signifikante Verschlechterung der Fehlerrate in Kauf genommen werden (siehe hierzu auch Abschnitt 2.1.4, Seite 41ff.). Während in Kapitel 2 diskutiert wurde, wie durch eine Transformation des Signalraumes eine gewisse Robustheit gegenüber wechselnden Umwelteinflüsse erzielt werden kann, soll sich nun zeigen, wie ein adaptiver Klassifizierer dazu beitragen kann, variierende Parameter zu kompensieren.

3.1 Motivation

Ohne Erweiterungen ist der Signalraumdetektor nicht in der Lage, sich ändernden Umwelteinflüssen Rechnung zu tragen. Aus diesem Grund ist es ratsam, ihn um zusätzliche Komponenten, z.B. adaptive Vorfilter, zu erweitern. So wurde in Abschnitt 2.1.4 dargestellt, dass bereits ein adaptives Vorfiltern des Lesesignals einer Festplatte den störende Einfluss von starkem Transitionsrauschen reduzieren kann.

Fehlen solche adaptiven Zusatzkomponenten, so erfordert die Änderung der Signalvorverarbeitung bzw. der relevanten Eingangsparameter ein Neudesign der Hardware- oder Softwareimplementierung des Detektors. Um diesen teuren und aufwändigen Entwicklungsschritt zu vermeiden, kann man auf einen lernfähigen Klassifizierer ausweichen. Ziel ist es dabei, die Ebenengleichungen dynamisch für die zu klassifizierenden Objekte zu wählen bzw. diese an die aktuellen Eingangs-

daten und deren Umwelteinflüsse anzupassen. Eine besonders vorteilhafte Ausprägung dieses Systems wäre eine Variante, bei der die Vorverarbeitung der Signale ebenfalls an die aktuelle Konstellation angepasst wird, also eine Kombination aus adaptivem Vorfilter und adaptiven Ebenengleichungen erfolgt.

Es ist mithin zu erwarten, dass sich mit einem derartigen adaptiven Konzept Entwicklungskosten reduzieren lassen und gleichzeitig die Performance eines statischen Klassifizierers übertroffen werden kann: Ein solches System wäre einerseits in der Lage, auf nicht modellierte Umwelteinflüsse zu reagieren, andererseits kann es technologisch bedingte Bauteiltoleranzen innerhalb eines gewissen Toleranzbereiches ausgleichen.

Daher ist es eine Herausforderung, den Signalraumdetektor um adaptive Komponenten zu erweitern. Derartige Ansätze werden in diesem Kapitel vorgestellt. Schließlich wird gezeigt, wie die Signalvorverarbeitung in das adaptive Konzept einbezogen werden kann. Ein entsprechender voll-adaptiver Klassifizierer wird in Abschnitt 3.3 vorgestellt und analysiert.

Soll die Konstellation der Entscheidungsebenen in $\underline{\underline{R}}^M$ an zeitvariante Nebenbedingungen angepasst werden, so bieten sich prinzipiell zwei Verfahren an: Einerseits kann das Detektionskonzept komplett aus primitiven adaptiven Komponenten aufgebaut werden (vergleiche hierzu Abschnitt 1.2.2, Seite 15ff.). Eine weitere Variante besteht darin, die vorhandenen Subkomponenten bestehender Implementierungen zu parametrisieren. Dieser Ansatz soll zunächst verfolgt werden.

3.2 Drehen der Entscheidungsebenen

Für das erste vorgestellte Adaptionsverfahren sei zunächst AWGN angenommen. Somit ergeben sich kreisförmige Cluster c^j um die Referenzpunkte $\vec{r}^j$. Nun werden zwei beliebige Referenzpunkte $\vec{r}^a$ und $\vec{r}^b$ ausgewählt, die unterschiedliche Klassen k^a und k^b repräsentieren. Weiterhin sei $E^{(a,b)}$ die Entscheidungsebene, die gemäß Abschnitt 1.2.2.3 zwischen diesen Klassen unterscheidet. Unter den genannten Voraussetzungen wäre $E^{(a,b)}$ die verlängerte Seite eines Voronoi-Polygons. Für diesen Fall macht eine Adaption wenig Sinn, denn es ist nachweisbar, dass mit den Voronoi-Ebenen eine optimale Partitionierung des Merkmalsraumes bei Störung durch AWGN gegeben ist. Anders verhält es sich, wenn die Merkmalsvektoren gemäß Abschnitt 2 vorverarbeitet sind und dadurch das ursprünglich weiße Rauschen gefärbt wird.

Im Folgenden wird die Vorverarbeitung als lineare Transformation angenommen. Somit sind die ursprünglich kreisförmigen EQL zu Ellipsen deformiert. Die Hauptachsen von je zwei Ellipsen liegen dabei parallel zueinander [SHS00]. Das ist für die Wahl der Entscheidungsebenen von besonderer Bedeutung:

1. Der Mittelpunkt $\vec{m}^{(a,b)}$ der gedachten Verbindungsline zwischen $\vec{r}^a$ und $\vec{r}^b$ liegt auf der für diesen Fall optimalen Entscheidungsebene (unabhängig von SNR und Rauschkorrelation).

$$\vec{m}^{(a,b)} = \frac{1}{2} \cdot (\vec{r}^a + \vec{r}^b) \tag{3.1}$$

   Erklärung: Durch die Voronoi-Regionen sind im AWGN-Fall optimale Entscheidungsebenen gegeben [JM97]. Die Voronoi-Regionen verlaufen im AWGN-Fall durch den Mittelpunkt $\vec{m}^{(a,b)}$ der Strecke $\vec{r}^a\vec{r}^b$ [PS93]. Die lineare Signalraumtransformation ist umkehrbar durch eine lineare Rücktransformation in den AWGN-Fall [SHS00]. Bei einer linearen Transformation bleiben Abstandsverhältnisse erhalten [NAMM93]. Somit wird $\vec{m}^{(a,b)}$ in den Mittelpunkt der Strecke $\tau(\vec{r}^a\vec{r}^b)$ transformiert.

2. Aufgrund der Linearität der Transformationsvorschrift haben die Punkte $\tau(\vec{r}^a+\vec{x})$ und $\tau(\vec{r}^b+\vec{x})$ die gleiche Auftrittswahrscheinlichkeit. Diese Aussage wird in Abbildung 3.1 durch die Punkte s^j und $(r^b + \vec{x})$ illustriert.

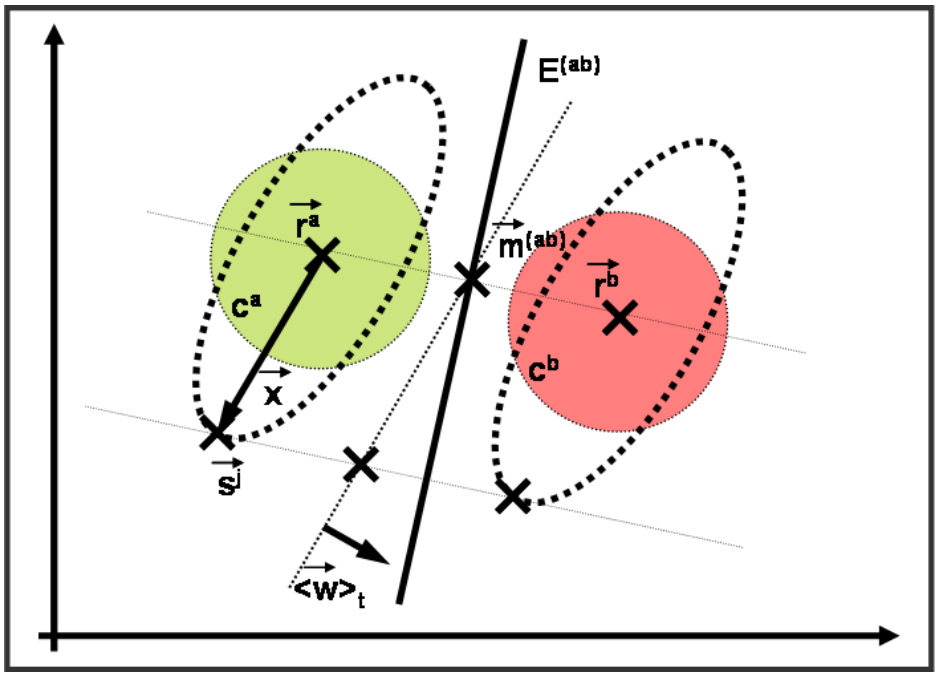

Abbildung 3.1: Adaptive Entscheidungsebene für SSD mit $M = 2$.

Die optimierte Entscheidungsebene kann demnach durch Gleichung 3.2 bestimmt werden, da $\vec{m}^{(a,b)} = (\vec{r}^a + \vec{r}^b)/2$ offenbar auf $E^{(a,b)}$ liegen muss.

$$E^{(a,b)} : \vec{s} \cdot \vec{w} = \frac{1}{2} \cdot (\vec{r}^{\,a} + \vec{r}^{\,b}) \cdot \vec{w} \tag{3.2}$$

Für die optimierte Entscheidungsebene muss noch der Normalenvektor $\vec{w}$ bestimmt werden, der sich typischerweise von dem der für den AWGN-Fall optimierten Entscheidungsebene unterscheidet. Er ist so auszurichten, dass die Summe der Abstände zwischen Entscheidungsebene und *jedem* beobachteten und durch diese Ebene zu klassifizierenden Punkt $\vec{s}^{\,j}$ maximiert wird. Einen ähnlichen Ansatz findet man bei den Support-Vektor-Maschinen. Da der Trainingssatz hier jedoch a priori unbekannt ist, kann die obige Forderung nicht nur auf Support-Vektoren beschränkt werden, da diese hier nicht bestimmbar sind. Vielmehr wird eine Heuristik verwendet, die alle zu klassifizierenden Merkmalsvektoren gleich behandelt.

Diese Abstandsmaximierung kann für $E^{(a,b)}$ in $\underline{\underline{R}}^2$ durch folgenden Algorithmus erfolgen:

1. Bestimme $\vec{x} = \vec{s}^{\,j} - \vec{r}^{\,a}$ für $\varepsilon(\vec{s}^{\,j}) = k^a$.

2. Drehe $E^{(a,b)}$ derart um $\vec{m}^{(a,b)}$, dass sie durch $\frac{1}{2} \cdot (\vec{s^j} + \vec{r}^{\,b} + \vec{x})$ verläuft[1].

3. Bestimme den optimierten Normalenvektor $\langle \hat{\vec{w}} \rangle_t$ als den Vektor, der senkrecht auf der Verbindungsgeraden beider Aufpunkte steht.

Für Signalraumdetektoren höherer Dimension ist die optimierte Entscheidungsebene nicht durch zwei, sondern durch M Aufpunkte gegeben. Diese können gewonnen werden, indem das beschriebene Verfahren auf die Punktmenge $\{\vec{m}, \vec{s}^{\,0}, \ldots \vec{s}^{\,(M-2)}\}$ angewendet wird. In jedem Schritt wird die perfekte Partitionierung zum Trennen der letzten M Punkte gefunden. Allerdings wird gleichzeitig sämtliches Vorwissen aus den vorhergehenden Adaptionsschritten verworfen.

Als Konsequenz ist davon auszugehen, dass die Adaption nicht konvergiert, sondern dass der Normalenvektor in jedem Lernschritt in Abhängigkeit von den Eingangsdaten springt. Dies ist insbesondere dann kritisch, wenn die Eingangsdaten fehlerhaft bzw. verrauscht sind. Dann wird die Adaption des Normalenvektors durch den jeweiligen Rauschanteil des aktuell zu klassifizierenden Objektes bestimmt. Ein derartiges Verhalten ist nicht wünschenswert.

Darum sollte das Vorwissen der bisherigen Adaptionsschritte berücksichtigt werden. Dieses Wissen ist in $\langle \hat{\vec{w}} \rangle_{t-1}$ gespeichert. Somit ist es sinnvoll, den Normalenvektor nicht auszutauschen, sondern $\langle \vec{w} \rangle_t$ aus dem vorherigen $\langle \vec{w} \rangle_{(t-1)}$ und $\langle \hat{\vec{w}} \rangle_t$ zu berechnen. Dazu ist festzulegen, wie dabei die Anpassung an das lokale Optimum $\langle \hat{\vec{w}} \rangle_t$ erfolgen soll.

[1] Auch wenn sich hierdurch die Ebenenausrichtung in bestimmten Fällen verschlechtert, kann bei einer genügend großen Trainingsmenge davon ausgegangen werden, dass sich Drehungen in die „falsche“ Richtung nach links bzw. rechts im Mittel ausgleichen.

3.2.1 Bestimmung des Normalenvektors

Die optimale Ebenengleichung bezüglich einer Menge von S zu klassifizierenden Objekten erhält man anschaulich, indem man alle beobachteten Signalpunkte, gemäß ihrer Auftrittswahrscheinlichkeit gewichtet, in die Optimierung der Ebenen einbezieht (nachfolgend Gleichung 3.3). Betrachtet man sehr viele Signalpunkte, so ist die Auftrittswahrscheinlichkeit implizit über deren Verteilung in der Trainingsmenge gegeben: Niedrig wahrscheinliche Punktkonstellationen kommen seltener vor als höher wahrscheinliche und sind damit weniger stark berücksichtigt.

$$\langle \vec{w} \rangle_t = \frac{1}{S} \left(\langle \hat{\vec{w}} \rangle_t + \sum_{i=1}^{S-1} \langle \vec{w} \rangle_{(t-i)} \right) \tag{3.3}$$

Der Speicheraufwand der durch Gleichung 3.3 beschriebenen Vorgehensweise steigt linear mit dem Produkt aus der Anzahl S der zur Adaption verwendeten Merkmalsvektoren und der Dimension M des Signalraumes. Er wächst außerdem quadratisch mit der Anzahl R relevanter Entscheidungsebenen auf $o(S \cdot M \cdot R^2)$. Eine Möglichkeit, den Speicherbedarf zu verringern, besteht sowohl im Einsatz einer anderen Gewichtungsfunktion als auch in der Verwendung eines geeigneten Abbruchkriteriums, resp. der Beschränkung von S.

Ein weiterer Schritt, den Speicherbedarf zu reduzieren, besteht darin, zwischen entscheidenden und nicht entscheidenden Entscheidungsebenen zu filtern [SSHS03]. Hierzu sei auf Abbildung 3.2 vorweg gewiesen: Ist ein Objekt s^j der grün dargestellten Klasse k^1 zuzuordnen, so ist $E^{(2,3)}$ eine nicht entscheidende Ebene. Ausschließlich $E^{(1,2)}$ und $E^{(1,3)}$ leisten zur Klassifizierung einen relevanten Beitrag. Daher muss $w^{(2,3)}$ nicht in Abhängigkeit vom dargestellten Punkt $\vec{s}^j$ geändert werden. Also muss zwingend zwischen entscheidenden und nicht entscheidenden Ebenen unterschieden werden [SSHS03].

Eine derartige Filterung ist leicht, sofern ein Signalraumdetektor mit Ausgabe von Softinformation (S3D-Detektor) verwendet wird. Dieser besitzt bereits ein Modul zum Filtern zwischen entscheidenden und nicht entscheidenden Ebenen und weist durch die Extraktion der so genannten Zuverlässigkeit einen weiteren Vorteile auf, der bei einer nachgelagerten Datenverarbeitung zum Tragen kommen kann [SSHS03, SD02].

Extrahierte Zuverlässigkeitsinformation kann zum Beispiel zum Markieren von unsicheren Klassifizierungen bzw. möglichen Fehlklassifizierungen verwendet werden. Beispielsweise wird bei Festplatten ein Fehlerkorrekturcode (ECC) eingesetzt, der um so mehr Fehler korrigieren kann, je genauer mögliche Detektionsfehler eingegrenzt sind [Ber68]. Bei praktischen Implementierungen kann durch Einsatz eines S3D-Detektors die Korrekturfähigkeit der ECC nahezu verdoppelt werden [Sch04].

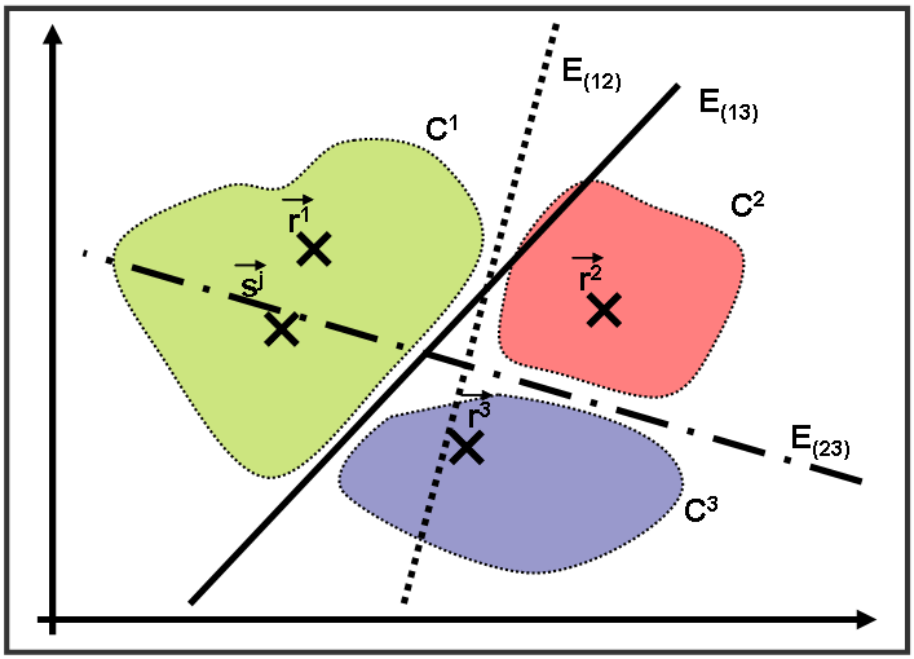

Abbildung 3.2: Darstellung entscheidender und irrelevanter Entscheidungsebenen.

Beim Einsatz eines S3D-Detektors ergibt sich noch ein weiterer Synergie-Effekt. Mit Hilfe der Zuverlässigkeitsinformation kann auch während der Adaption zwischen sicheren und unsicheren Klassifizierungen unterschieden werden: Sofern ausschließlich unsichere Klassifizierungen zur Adaption berücksichtigt werden, kann der zur Ebenenanpassung erforderliche zusätzliche Leistungs- und Zeitbedarf signifikant reduziert werden [SD02, DS03]. Eine Verringerung der Fehlerrate ist ebenfalls zu erwarten, wenn man zum Training des Detektors die fälschlicherweise als „sicher“ erkannten Fehlklassifizierungen benutzt.

3.2.2 Simulationsergebnisse

Das beschriebene Verfahren zur adaptiven Signalraumdetektion soll abermals am Beispiel eines Festplattenlesekanals untersucht werden. Zur Analyse wird wiederum das Festplattensimulationstool HdSim eingesetzt [DPS01]. Um die Fehlerrate bei Einsatz der vorgestellten Verfahren bestimmen und vergleichen zu können, werden das WSSD-Detektionsverfahren und zwei adaptive Klassifizierer nacheinander simuliert.

In einem Fall wird der vorgestellte adaptive Detektor zunächst trainiert und dann eingesetzt. Aufgrund dieses Verhaltens wird der Detektor im Folgenden als offline adaptiver Signalraumdetektor (OF-ASSD) bezeichnet. Zum Vergleich wird ein weiterer Klassifizierer eingesetzt, der nach dem gleichen Verfahren arbeitet,

jedoch immer dann zurück in den Trainingsmodus springt, wenn die Zuverlässigkeit einen Schwellwert ζ_{min} unterschreitet. Analog zum OF-ASSD wird dieser online adaptive Klassifizierer hier ON-ASSD genannt.

Es ist zu erwarten, dass der OF-ASSD sich etwas performanter als der ON-ASSD Detektor verhält, während der ON-ASSD auf Änderungen von Kanalparametern zur Laufzeit reagieren kann. Demgegenüber ist zu erwarten, dass der OF-ASSD nach dem Absolvieren der Trainingsroutine ein quasi-statisches Verhalten aufweist.

In einem ersten Experiment wurde die Adaptionsfähigkeit der Detektoren untersucht. Hierzu wurden die drei Detektoren auf Lesekanälen mit unterschiedlicher Bitdichte PW_{50} betrieben und jeweils mit einem Vorfilter ausgestattet, dessen Parametervariation $\sigma^2 = 0,05$ betrug. Schließlich wurde der notwendige Signal-Rausch-Abstand in dB, welcher zur Gewährleistung einer für Festplatten-Lesekanäle typischen Bitfehlerrate von 10^{-5} erforderlich ist, für alle drei Detektoren gemessen.

Die hierbei erzielten Messergebnisse sind in Tabelle 3.1 dargestellt. Es ist zu beobachten, dass der offline-adaptive Detektor – wie erwartet – eine etwas höhere Detektionsgenauigkeit erzielt als der ON-ASSD. Beide adaptiven Verfahren erzielen signifikant bessere Ergebnisse als der WSSD-Detektor.

PW_{50}	WSSD	ON-ASSD	OF-ASSD
2.75	26.8	26.2	26.2
2.81	26.8	26.4	26.4
2.92	26.9	26.6	26.5
3.00	27.2	27.0	26.8
3.25	28.5	28.0	27.8

Tabelle 3.1: Adaptionsverhalten, um Bauteilvarianzen beim Equalizer zu kompensieren. Dargestellt ist der notwendige SNR für drei Detektoren, um eine BER von 10^{-5} zu gewährleisten.

Im zweiten Experiment wurde die Robustheit der Detektoren gegen Pulsweitenvariation getestet. Diese Störung, die durch Fehler bei der Taktrückgewinnung des Lesesignals entsteht, wird sich in den kommenden Jahren bei steigender Datendichte zu einem signifikanten Störfaktor im Lesekanal entwickeln [Gro03, GH96, TB00].

Hierzu wurden auf simulierten Lesekanälen mit einer Pulsweite von $2,75 \leq PW_{50} \leq 3,50T$ sowohl der WSSD als auch der ON-ASSD (die Variante aus dem ersten Experiment mit $PW_{50} = 3,00T$) verwendet. In Abbildung 3.3 ist die Performance gegenüber der Bitdichte aufgetragen. Um die Ergebnisse zu bewerten, ist zusätzlich die theoretische untere Grenze für Signalraumdetektoren eingezeichnet [Str00].

Aus dem Diagramm 3.3 ist ersichtlich, dass der ON-ASSD eine verbesserte Fehlerrate im Vergleich zum statischen WSSD gewährleistet. Während dieser im Bereich von $PW_{50} = 3,00T$ den kürzesten Abstand zum theoretischen Optimum aufweist, zeigt der ON-ASSD für den gesamten Bereich $2,75 \leq PW_{50} \leq 3,17T$ ein um nur $0,2dB$ schlechteren SNR als die theoretische untere Grenze.

Somit lässt sich schlussfolgern, dass der ON-ASSD in der Lage ist, auf zeitvariante Einflüsse zu reagieren. Die Adaption kann sowohl online während des normalen Detektionsprozesses durchgeführt werden als auch offline (mit einer auf der Festplatte gespeicherten Trainingssequenz).

Nachteilig am vorgestellten Verfahren ist zum einen die Beschränkung auf lineare Signalraumtransformationen. Zum anderen ist zur Adaption die Kenntnis der Positionen aller $\vec{r}^n$ erforderlich.

Für genügend genau bekannte Signalräume, wie bei der Klassifizierung des Lesesignals einer Festplatte, ist dies kein Problem. Für andere Anwendungen ist dieses Verfahren unter Umständen untauglich. In solchen Fällen wird ein voll adaptives Verfahren benötigt, das ohne Vorwissen über den Signalraum funktioniert. Ein derartiges Verfahren wird im nächsten Abschnitt vorgestellt.

3.3 Adaption mit Perzeptronen

Für die Signalraumdetektion auf a priori unbekannten Kanälen bietet sich der Aufbau eines Signalraumdetektors aus primitiven adaptiven Komponenten an.

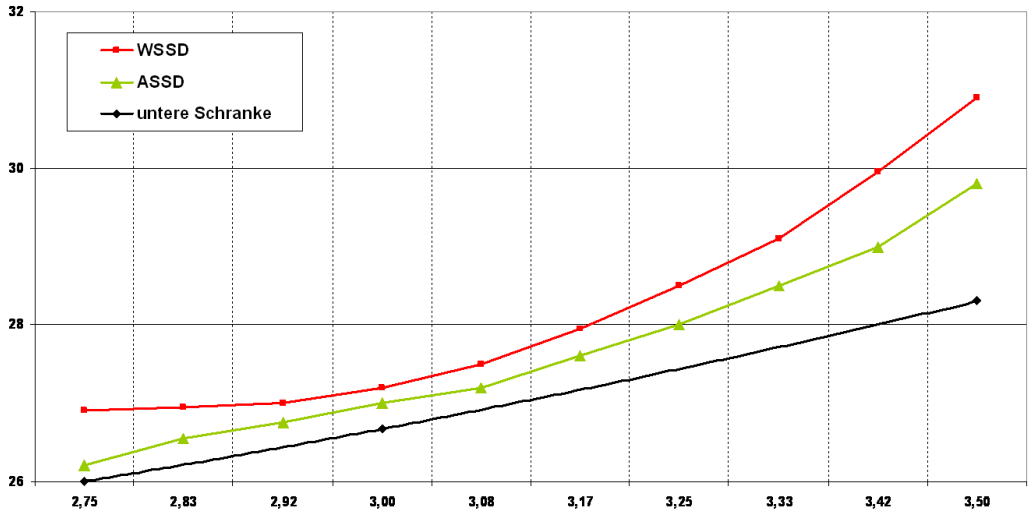

Abbildung 3.3: Robustheit gegen Pulsweitenvariation ohne Verwendung eines adaptiven Equalizers für WSSD und ON-ASSD.

Dies lässt sich beispielsweise mit adaptiven linearen Elementen oder künstlichen Perzeptronen in Analogie zum adaptiven Vorfilter aus Kapitel 2.1.3 (Seite 40f.) durchführen.

Zunächst ist darzustellen, wie eine Entscheidungsebene durch ein primitives adaptives Element implementiert werden kann. Sowohl beim Adaline als auch beim Perzeptron wird das Skalarprodukt aus Eingangsvektor $\vec{s^j}$ und Gewichtsvektor $\vec{w} = (w_1, w_2, \ldots w_M)^T$ gebildet. Schließlich wird eine Konstante w_{M+1}, der so genannte Bias, addiert. Wenn $\vec{s^j}$ den Eingangsvektor dieses Perzeptrons darstellt, so kann $\vec{w}$ gemäß Gleichung 1.17 als Normalenvektor einer Hyperebene interpretiert werden.

Darüber hinaus wird Das Skalarprodukt $\vec{s^j} \cdot \vec{w}$ beim künstlichen Perzeptron durch die Aktivierungsfunktion α gewichtet [Hay94]. Somit ist die Aktivität des Perzeptrons durch Gleichung 1.5 gegeben.

Insgesamt können n Perzeptronen ebenso viele Entscheidungsebenen modellieren. Zur Initialisierung der Entscheidungsebenen kann man zwischen zwei prinzipiell verschiedenen Verfahren unterscheiden:

(a) Vorwissen wird nicht berücksichtigt, die Gewichte werden mit zufälligen Werten initialisiert

(b) Vorwissen wird berücksichtigt, und die Gewichte bekommen „sinnvolle“ Startwerte.

Der zu entwickelnde Detektor soll insbesondere zur Klassifizierung dienen, wenn kein oder nur wenig Vorwissen über das Eingangssignal vorliegt. Dem entspricht gemäß Fall (a) eine Besetzung der Gewichtsvektoren mit zufälligen Startwerten.

Für die Alternative (b) bietet sich eine Nutzung des Vorwissens zur Beschleunigung des Trainingsverfahrens an. Es ist nahe liegend, die Gewichtsvektoren so einzustellen, dass sie die Entscheidungsebenen eines WSSD möglichst genau nachbilden.

Unabhängig von der Initialisierung der Gewichte ist ein anschließendes Training erforderlich, um die Ebenengleichungen an die tatsächlichen Bauelemente- und Kanalparameter anzupassen. Hierzu bietet sich der so genannten Backpropagation-Algorithmus [Hay94] als einfacher und effektiver Ansatz an, um die Gewichte bzw. die Ebenenkonstellation zu trainieren. Dieses Verfahren ist besonders einfach anzuwenden, wenn eine so genannte sigmoide Aktivierungsfunktion nach Gleichung 1.6 verwendet wird, da sich hierdurch die Lernregel stark vereinfacht [Hay94].

Während der Lernphase werden die Gewichtsvektoren justiert. Dabei werden die Hyperebenen durch Anpassung des Bias verschoben und durch geänderte Normalenvektoren rotiert. Hierdurch ergibt sich eine neue Partitionierung des Merkmalsraumes.

Zur Bestimmung des $\vec{s}^j$ umhüllenden Unterraumes $\varepsilon_0(\vec{s})$ wird beim WSSD der Abstand δ des zu klassifizierenden Punktes von den jeweiligen Entscheidungsebenen bestimmt. Sind die Normalenvektoren auf $||\vec{w}|| = 1$ normiert, so ergibt sich δ aus dem Skalarprodukt $\vec{w} \cdot \vec{s}^j$. Wird der Gewichtsvektor dementsprechend nach jedem Trainingsschritt normiert, so kann durch Anwendung der inversen Aktivierungsfunktion α^{-1} (siehe Gl. 1.6 und Gl. 3.4) dieser Abstand als $\alpha^{-1}(\pi)$ bestimmt werden:

$$\alpha^{-1} : x \rightarrow \Psi_1 - \frac{1}{\Psi_0} \cdot ln\left(\frac{1}{x} - 1\right) \tag{3.4}$$

Die Anwendung der Aktivierungsfunktion und ihrer Inversen stellt jedoch jeweils Operationen von erheblicher Komplexität dar. Somit erscheint es sinnvoll, durch Modifikation der Struktur eines konventionellen Perzeptrons auf diese Doppelberechnung zu verzichten. Eine derartige Struktur wird in Abbildung 3.4 präsentiert [Dah04]. Nach der Bestimmung des Skalarproduktes $\vec{w} \cdot \vec{s}$ liegt der Ebenenabstand δ bereits vor, sofern der Normalenvektor auf $||\vec{w}|| = 1$ normiert ist. Also wird δ auf einen zusätzlichen Ausgang des Perzeptrons gelegt, dessen Aktivität nun als $\pi = \alpha(\delta)$ bestimmt wird.

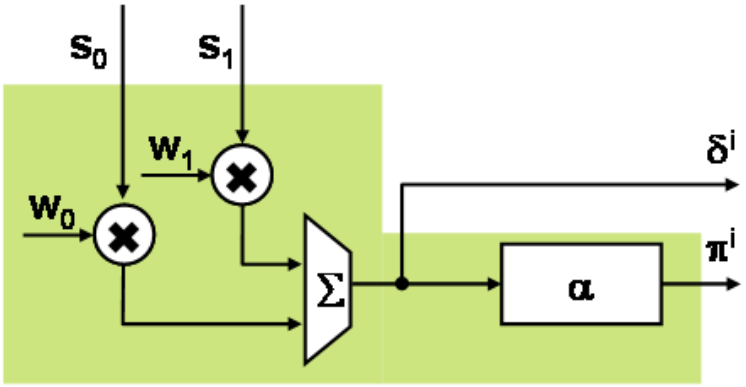

Abbildung 3.4: Prinzipskizze einer niedrig komplexen Erweiterung eines Perzeptrons, so dass Ebenenabstände bestimmt werden können.

Hierdurch sind die Ebenengleichungen in Perzeptronen und Perzeptronen-Ausgangssignale in Ebenengleichungen bijektiv abbildbar. Durch Anwendung dieses Verfahrens ist es nun prinzipiell möglich, eine adaptive Signalraumdetektion mit Perzeptronen vorzunehmen [DS03]. Moderne Detektoren bieten darüber hinaus auch die Möglichkeit der Extraktion von Zuverlässigkeitsinformation. Daher soll nun gezeigt werden, wie der adaptive Detektor um ein Modul zur Bestimmung dieser so genannten Soft-Information erweitert werden kann.

3.3.1 Adaptive Extraktion der Soft-Information

Bereits mehrfach wurde gezeigt, dass durch Extraktion von Soft-Information ohne großen algorithmischen und schaltungstechnischen Mehraufwand die Korrekturfähigkeit eines der Detektion nachgeschalteten Fehlerkorrekturcodes signifikant gesteigert werden kann [Ber68, LC83]. Derartige so genannte Soft-Information kann durch eine Reihe bekannter Verfahren aus dem Signalraumdetektor extrahiert werden [SSHS03, Ste02]. Grundsätzlich dient der Abstand eines Signalpunktes zur Entscheidungsebene als Maß für die Zuverlässigkeit: Punkte mit geringem Abstand zur Entscheidungsebene gelten als unzuverlässig, und Punkte, die dicht am Referenzpunkt liegen, werden mit einer hohen Zuverlässigkeit gekennzeichnet [SSHS03, DS03].

Dabei ist zu beachten, dass – wie in Abschnitt 3.2.1 dargestellt – einige Entscheidungsebenen keine relevante Information für die Detektion spezieller Signalpunkte zur Verfügung stellen, wie z.B. $E^{(2,3)}$ bezüglich $\vec{s}^j$ (in Abbildung 3.2). Die Hyperebene befindet sich zwar nahe $\vec{s}^j$, was eine geringe Zuverlässigkeit impliziert. Andererseits ist $\vec{s}_j$ weit vom c^2 und c^3 entfernt, was eine sichere Entscheidung kennzeichnet. Um diesen Widerspruch zu lösen, muss zwingend zwischen entscheidenden und irrelevanten Entscheidungsebenen unterschieden werden [SSHS03].

In der S3D-Architektur wird die Relevanz $\gamma \in \{0, 1\}$ einer Entscheidungsebene folgendermaßen bestimmt: Der Signalpunkt wird auf die gegenüberliegende Seite einer Entscheidungsebene gespiegelt. Ändert sich dadurch das Detektionsergebnis, so wird diese Ebene als „entscheidend“ klassifiziert. Hierzu repräsentiert ein Boolescher Wert $\varepsilon_0^{(a,b)}(\vec{s}^j)$ die Seite der Entscheidungsebene $E^{(a,b)}$, auf der sich $\vec{s}^j$ befindet. Dieser Wert wird durch die Boolesche Algebra β propagiert. Ändert sich das Ergebnis am Ausgang der Algebra, wenn die Eingangsgröße invertiert wird, so ist diese Ebene „entscheidend“:

$$\gamma = \left(\beta(\cdots, \varepsilon_0^{(a,b)}(\vec{s}^j), \cdots) \neq \beta(\cdots, \overline{\varepsilon_0^{(a,b)}(\vec{s}^j)}, \cdots)\right) \tag{3.5}$$

Wird das vorgeschlagene adaptive System verwendet, so ist eine Boolesche Logik nicht anwendbar, da die Perzeptronen keine binäre Ergebnisse liefern. Statt dessen könnte die Funktion β durch ein MLP gemäß Abschnitt 1.2.2.2 (Seite 18) approximiert werden. Aufgrund der Modellierbarkeit nichtlinearer Trennungen, wie der XOR-Funktion, ist die Verwendung von mindestens zwei Schichten ratsam. Die Boolesche Algebra muss auf diese Weise durch das MLP vollständig ersetzt werden.

Vorhandenes Wissen kann berücksichtigt werden, indem man die Gewichtsvektoren des MLP genau so initialisiert, dass die Booleschen Funktionen der S3D-Architektur nachgebildet werden. Tabelle 3.2 zeigt eine hierzu verwendbare Ersetzungstabelle.

Boolesche Operation		w_0	w_1	w_2
$s_0 \bigvee s_1$	OR	1	1	1
$s_0 s_1$	AND	1	1	-1
$s_0 \bigvee \overline{s_1}$		1	-1	1
$s_0 \overline{s_1}$		1	-1	-1
$\overline{s_0} \bigvee s_1$		-1	1	1
$\overline{s_0} s_1$		-1	1	-1
$\overline{s_0} \bigvee \overline{s_1}$	NAND	-1	-1	1
$\overline{s_0 s_1}$	NOR	-1	-1	-1

Tabelle 3.2: Gewichtsvektoren zur Approximation Boolescher Operationen. Die Tabelle gilt für $s_i = 1 \Rightarrow TRUE$, $s_i = 0 \Rightarrow FALSE$. Das Gewicht w_2 ist als Bias zu verstehen.

Zur Bestimmung der Relevanz einer Entscheidungsebene wird die Aktivität des betreffenden Perzeptrons π^i durch Bildung von $(1 - \pi^i)$ invertiert. Dieser Wert wird dann durch das MLP propagiert. Ändert sich durch die Invertierung der Aktivität π^i das Endergebnis $\mu = \varepsilon(\vec{s}^j)$ des Detektionsprozesses, so ist die durch π^i approximierte Ebene als „entscheidend“ zu deklarieren.

Das Blockschaltbild eines MLP mit Modul zur Bestimmung der Relevanz der Entscheidungsebenen und Weiterleitung von Zuverlässigkeitsinformation ζ findet sich in Abbildung 3.5. Die Aktivität der Perzeptronen π^2 bis π^5 wird berechnet. Um beispielsweise zu ermitteln, ob π^3 eine relevante Entscheidungsebene repräsentiert, wird $(1 - \pi^3)$ bestimmt und propagiert. Dann wird geprüft, ob sich π^5 signifikant geändert hat $\xi(\langle\pi^5\rangle_t) \neq \xi(\langle\pi^5\rangle_{(t-1)})$. Ist dies der Fall, wird γ^3 gesetzt.

Spätestens am Strukturbild in Abb. 3.5 wird deutlich, dass nicht nur die Aktivierung der Perzeptronen π^i, sondern auch die zugehörigen Zuverlässigkeitswerte ζ^i sinvoll zusammengeführt werden müssen. Als besonders geschickt erweist es sich, zwei Zuverlässigkeitswerte ζ^i und ζ^j so zusammen zu fassen, dass diese Fusion mit der durch das Perzeptron abgebildeten Booleschen Verknüpfung vergleichbar wird, denn so wird die Berechnung transparent uns anschaulich nachvollziehbar.

Zur quasi-Booleschen Verknüpfung nicht-binärer Werte existiert eine Reihe von Verfahren, die insbesondere aus der Fuzzy-Logik bekannt sind [Zad65, DP80]. Beispiele vermittelt Tabelle 3.3. Aufgeführt sind jeweils der Name der Verknüpfung, die zugrunde liegende mathematische Funktion und eine Beispielrechnung für die Werte $x = 0,4$ und $y = 0,6$.

Welches der in Tabelle 3.3 dargestellten Verfahren im jeweiligen Anwendungsfall zum Einsatz kommen soll, hängt von verschiedenen Randbedingungen ab. Eine wesentliche und vielfach diskutierte Randbedingung bei der Klassifizie-

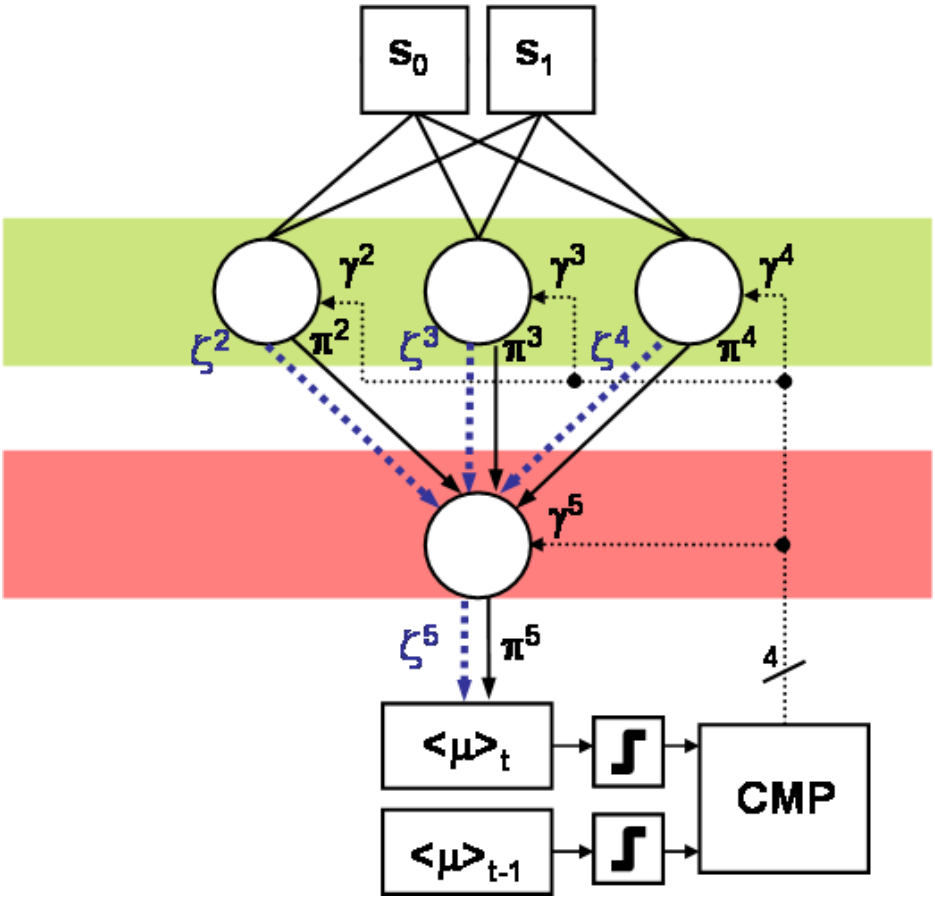

Abbildung 3.5: Propagieren und Rückkoppeln von Zuverlässigkeitsinformation im erweiterten Perzeptronennetzwerk.

rung von Signalfolgen ist die Schaltungskomplexität des implementierten Verfahrens [HF92, BM98, EPG94, KM98b].

Unter der Nebenbedingung einer niedrig komplexen Implementierung scheidet das algebraische Verfahren wegen der benötigten Multiplizierer aus. In der Digitaltechnik ist die ausschließliche Verwendung nichtnegativer Wertefolgen prinzipiell vorteilhaft, da Komplexität für die getrennte Vorzeichenbetrachtung gespart werden kann. In diesen Fällen lässt sich die Bold-Verknüpfung ebenfalls nicht einsetzen. Darüber hinaus weist die kombinierte Verwendung von min/max eine Reihe von algebraischen Vorteilen auf, die ihre Verwendung nahe legen [SK98]. Daher verwendeten wir in unseren Arbeiten die Näherungsfunktion (min/max), um die extrahierten Zuverlässigkeitswerte zu verschmelzen [DS03, Dah04, DZ02a].

Eine Alternative zur exakten Bestimmung der Zuverlässigkeitswerte stellt deren Approximation dar. Hierzu wurde vorgeschlagen, ζ^i als Minimum aller vorhergehenden relevanten abhängigen Zuverlässigkeitswerte ζ^j mit $j < i$ zu approxi-

mieren [SSHS03]. Entscheidet man sich zur Verwendung dieser Näherungswerte, so lässt sich die Komplexität der Gesamtstruktur nochmals reduzieren, indem man die Perzeptronen erneut modifiziert (siehe Abbildung 3.6). In Analogie zu Abb. 3.4 wird vor Anwendung der Aktivierungsfunktion ein zusätzlicher Zweig eingeführt, der gemäß Abbildung 3.5 mit den anderen auf diese Weise modifizierten Perzeptronen zu verdrahten ist.

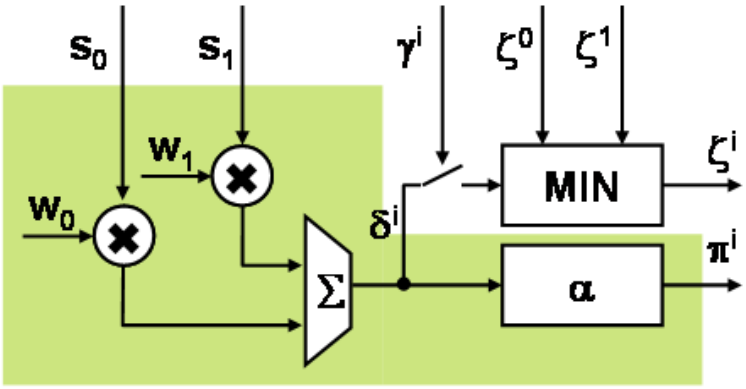

Abbildung 3.6: Blockschaltbild einer niedrig komplexen Erweiterung eines Perzeptrons, so dass Zuverlässigkeitsinformation extrahiert werden kann.

3.3.2 Simulationsergebnisse

Der konstruierte voll adaptive Detektor, im Folgenden AS3D genannt, wurde erneut als Detektor für den magnetischen Kanal einer Festplatte eingesetzt. Zur Bestimmung der Qualität dieses neuartigen Klassifizierers wurden drei Punkte untersucht:

1. Fehlerrate unter Einfluss von Parametervariation,
2. Fehlerrate unter Einfluss von Transitionsrauschen,
3. Qualität der extrahierten Zuverlässigkeitsinformation.

	x AND y	x OR y	AND*	OR*
Näherung	min(x,y)	max(x,y)	0.40	0.60
Algebraisch	x · y	(x+y)-(x·y)	0.24	0.86
Bold-Verknüpfung	x+y-1	x+y	0.00	1.00

Tabelle 3.3: Verknüpfungen zur Übertragung Boolescher Algebra auf nicht-diskrete Werte.

Zur Untersuchung der Robustheit des AS3D gegenüber Parametervariation wurden nichtadaptive Vorfilter verwendet, die für eine Pulsweite $PW_{50} = 3,0T$ ausgelegt waren. Anschließend wurden die Koeffizienten der Konstantenmultipilizierer der Vorfilter gestört mit AWGN der Varianz $\sigma^2 = 0,025$. Somit zeigt die Simulation die Robustheit des Detektors gegenüber Parametervariationen und Fertigungstoleranzen. In der Simulation von 500 Millionen Transitionen wurde die Bitfehlerrate bei einem jeweils vorgegebenem Signal-Rausch-Abstand zwischen 26 und 28 dB sowohl für den WSSD als auch für den AS3D bestimmt.

Abbildung 3.7 zeigt den Unterschied der Bitfehlerrate von WSSD und AS3D in Prozent. Zum einen bestätigt sich – unter Berücksichtigung von Parametervariation – die Erwarung, dass der AS3D in jedem Fall eine niedrigere Bitfehlerrate als der WSSD gewährleistet. Außerdem ist zu erkennen, dass der Vorteil des AS3D mit steigendem Signal-Rausch-Abstand wächst. Dies ist darauf zurück zu führen, dass bei hohem SNR der Einfluss der (konstanten) Parametervariation gegenüber dem AWGN wächst. Das erleichtert dem Adaptionsalgorithmus die Anpassung an diese Störgröße.

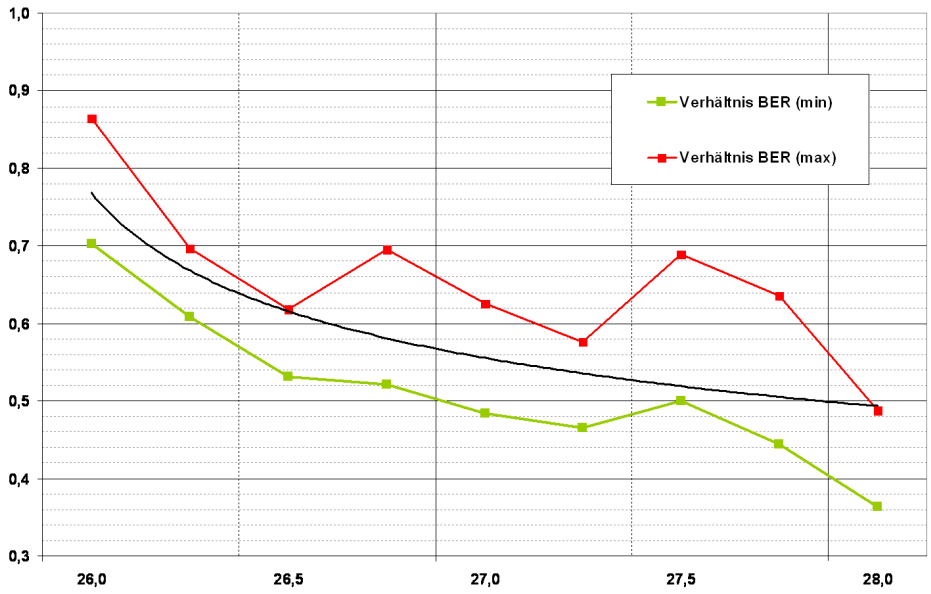

Abbildung 3.7: Verbesserung der Bitfehlerrate des AS3D im Vergleich zum WSSD.

In einem zweiten Experiment wurde ebenfalls die Fehlerrate gemessen, diesmal allerdings unter Einfluss von Transitionsrauschen. In der Simulation von jeweils 50 Millionen Transitionen wurde die Bitfehlerrate (bei einem jeweils vorgegebenem Signal-Rausch-Abstand) sowohl für den WSSD als auch für den AS3D gemessen. Die Ergebnisse sind in Abbildung 3.8 aufgetragen. Es ist erkennbar, dass der AS3D in allen simulierten Fällen eine bessere Performance als der WSSD gewährleistet.

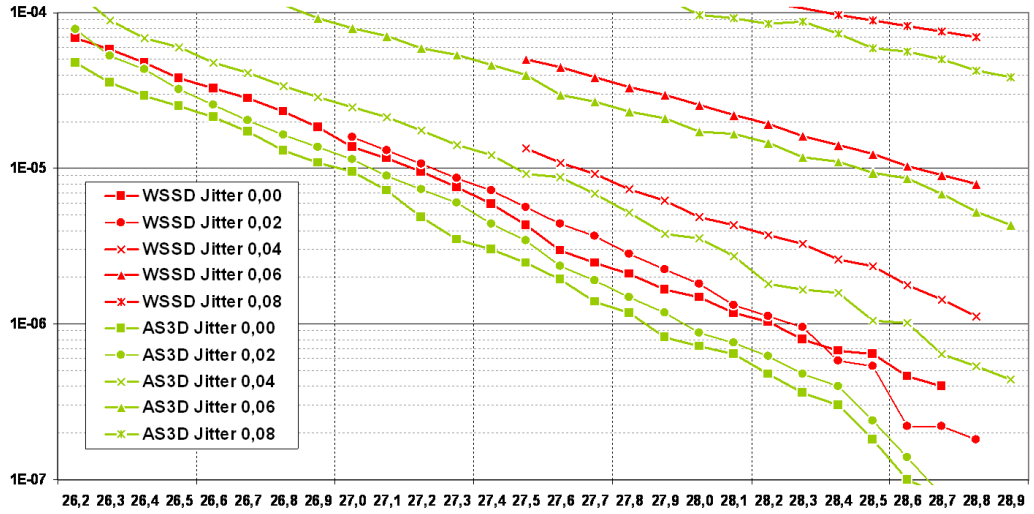

Abbildung 3.8: Vergleich der BER von adaptivem SSD und WSSD in Abhängigkeit von Jitterrauschen bei einer Pulsbreite $PW_{50} = 3,0T$.

Aufgrund seiner Anpassungsfähigkeit ist die Detektionsgüte des AS3D deutlich besser als die des WSSD. Andererseits würde ein WSSD, der für genau diesen Kanal optimiert worden wäre, eine bessere Detektionsqualität bieten. Daher ist der adaptive Detektor besonders dann vorteilhaft einsetzbar, wenn die Kanalparameter zum Zeitpunkt der Implementierung nicht oder zumindest nur unvollständig bekannt sind. Zusätzlich ist der Vorteil der reduzierten Entwurfskosten zu berücksichtigen, da der ohnehin gegen Fertigungstoleranzen robuste Detektor [DSS00] sich automatisch an Schwankungen in der Qualität der Bauelemente (z.B. der eingesetzten Filterbausteine) anpassen kann [SD02]. Verbesserte Komponenten des Lesekanal schlagen sich also direkt in einer verringerten Bitfehlerrate des Detektors nieder.

Schließlich ist die Qualität der extrahierten Zuverlässigkeitsinformation zu bestimmen. Sinnvollerweise müssten als „unzuverlässig“ markierte Bits häufiger fehlerhaft sein, als die als „zuverlässig“ markierten. Um diese Quote zu bestimmen, werden zwei Messwerte verwendet (vgl. Tabelle 3.4): Ein erstes Maß ist durch die Trefferrate gegeben, wie viele fehlerhafte Klassifizierungen korrekterweise als „unzuverlässig“ markiert wurden (Spalte „Richtig“). Ebenso wichtig ist die Frage, wie viele fehlerhafte Klassifizierungen fälschlicherweise als „zuverlässig“ klassifiziert wurden (Spalte „Falsch“). Um die angegebenen Zahlenwerte vergleichen zu können, werden sie mit den Werten des S3D verglichen.

Desweitern ist der Übersicht zu entnehmen, dass bei einem SNR von 28dB fast 80% aller Klassifikationsfehler durch den AS3D korrekt erkannt bzw. mit einer niedrigen Zuverlässigkeit markiert werden. Demgegenüber garantiert der S3D eine

SNR (dB)	S3D		AS3D	
	Richtig markiert	Falsch markiert	Richtig markiert	Falsch markiert
23	28.5	32.6	29.2	31.9
24	62.2	24.6	60.3	24.5
25	77.2	19.1	71.8	19.6
26	82.0	12.0	74.0	12.6
27	85.5	6.1	77.1	6.9
28	90.3	3.5	78.6	4.2

Tabelle 3.4: Korrektheit der extrahierten Zuverlässigkeitsinformation in Prozent.

Trefferquote von ca. 90% [SSHS03]. Bei gleichem Signal-Rausch-Abstand liegt die Rate der fehlerhaft markierten Klassifizierungen bei 4.2% – nur 0.7% mehr als beim statischen S3D.

Somit zeigt sich, dass durch den Einsatz des AS3D sinnvoll verwertbare Zuverlässigkeitsinformation gewonnen werden kann. Demnach kann z.B. in einer Festplatte ein Fehlerschutzcode mit geringerer Redundanz eingesetzt werden [Sch04]. Die ECC kann somit weniger komplex ausfallen und daher mit einem geringerem Leistungsbedarf implementiert werden. Im Sinne eines leistungsarmen Gesamtsystems ist die Verwendung eines adaptiven Detektors (ASSD) sogar trotz der erhöhter Schaltungskomplexität sinnvoll [Sch04, DS03].

Darüber hinaus öffnet die Adaptionsfähigkeit des entwickelten Klassifizierers dem Signalraumdetektor neue Anwendungsgebiete. Insbesondere ist der AS3D in der Lage, Parametervariation und Bauteiltoleranzen zu kompensieren und effektiv auszugleichen.

Kapitel 4

Multi-Symbol-Detektion

Eine wichtige Eigenschaft von Klassifizierern ist die erzielbare Durchsatzrate. Daher wird im Folgenden die Signalraumdetektion insbesondere unter dem Aspekt betrachtet, wie sie so modifiziert werden kann, damit sie den Bedürfnissen jener Anwendungsfelder gerecht wird, die eine besonders hohe Klassifikationsgeschwindigkeit erfordern.

Die Durchsatzrate des Signalraumdetektors ist technologisch bedingt begrenzt. Ursache ist die hohe Anzahl an Multiplizierern. Diese benötigen pro Rechenschritt eine längere Laufzeit als Addierer, was sich negativ auf die Laufzeit der Gesamtschaltung auswirkt [Str02]. Daher soll im Unterschied zu einem konventionellen SSD, der pro Taktschritt ein Symbol klassifiziert, ein so genannter Multi-Symbol-Detektor (MSD) in der Lage sein, in jedem Takt mehrere Symbole gleichzeitig zu erkennen.

Liegt der Signalraumdetektor als Digitalschaltung vor, so lässt sich auf diese Weise eine weitere Randbedingung erfüllen: Anstatt den Vorzug der Parallelität der getroffenen Entscheidungen für eine Erhöhung der Geschwindigkeit zu nutzen, könnten alternativ (bei gleichbleibender Datenrate) sowohl Taktrate als auch Versorgungsspannung herabgesetzt werden. Dies kann zu einer Reduktion des Leistungsbedarfs führen.

4.1 Konzeption des Multi-Symbol-Detektors

Ein Anwendungsfall, der eine besonders hohe Durchsatzrate erfordert, ist der bereits mehrfach zitierte Festplattenlesekanal [Gro03]. Als Besonderheit der Merkmalsvektoren des dortigen Klassifikationsprozesses ist festzustellen, dass diese stark miteinander korreliert sind. Liegt am Ausgang des Equalizers die Signalfolge $s_0, s_1, s_2, s_3, s_4 ...$ an, so wird der Signalraumdetektor mit den Merkmalsvektoren $(s_0, s_1, s_2)^T, (s_1, s_2, s_3)^T, (s_2, s_3, s_4)^T ...$ gespeist.

Eine weitere Spezialisierung des SSD für den Festplattenlesekanal besteht darin, dass das klassifizierte Signal zum Eingang des SSD zurückgekoppelt wird, um

den Einfluss der für den magnetischen Kanal typischen Intersymbolinterferenz zu reduzieren. Die beschriebenen Veränderungen sind im Vergleich zum allgemeingültigen SSD als Blockschaltbild in Abbildung 4.1 abgebildet und grün hinterlegt.

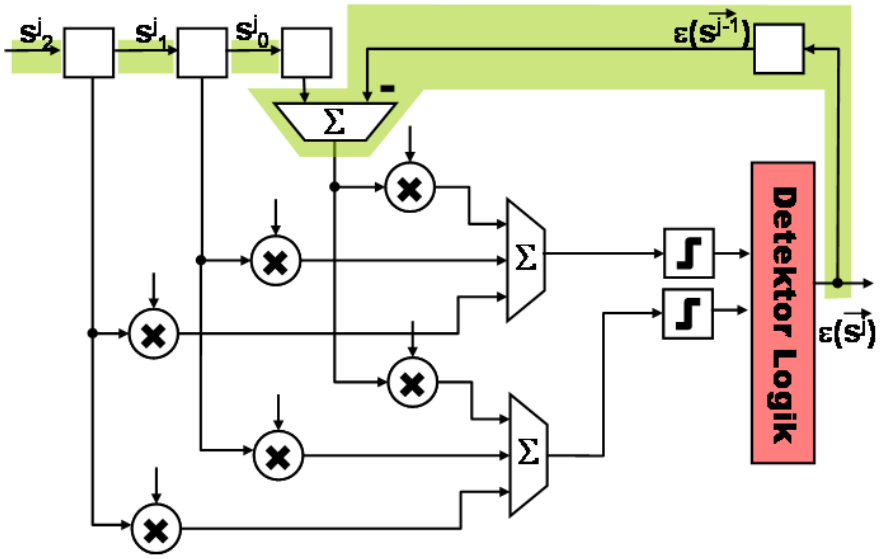

Abbildung 4.1: Prinzipskizze des SSD-Detektors für den magnetischen Kanal.

Um für den abgebildeten Detektor die gewünschte Parallelität zu erreichen, nämlich eine Anzahl G Symbole quasi gleichzeitig zu detektieren, werden G zunächst unabhängig voneinander arbeitende Klassifizierer von der gleichen, um einen Takt versetzten Eingangsfolge gespeist und miteinander verschaltet. Diese Technik ist u.a. aus dem Compilerbau als loop-unroll bekannt [ASU86, BGS94]. Durch ihren Einsatz wird einerseits die Parallelität erhöht, was einen G-fach höheren Hardwareaufwand bedeutet. Mittels Wiederverwendung von Zwischenergebnissen (common subexpression elimination) in Kombination mit algorithmischen Vereinfachungen (algebraic simplification) soll im weiteren Verlauf ein Teil der zusätzlich benötigten Hardware wieder eingespart werden, so dass die Komplexität des Verfahrens (bezogen auf die Anzahl gleichzeitig erkannter Symbole) insgesamt reduziert wird.

Die Eingangsdatenfolge eines MSD besteht aus $(M + G - 1)$ vorgefilterten Samples $(s_0, s_1, s_2, ...s_{M+G-2})$. Sie werden in jedem Taktschritt um G Stellen geschoben. Jeweils M aufeinander folgende Samples werden der Eingangsfolge entnommen und sind somit Eingang eines separaten Signalraumdetektors. Diese Signalraumdetektoren seien mit $D^0, D^1, ...D^{G-1}$ bezeichnet, wobei der Detektor D^i

auf dem Merkmalsvektor $(s_i, s_{i+1}, ...s_{i+M-1})^T$ arbeitet. Die Ergebnisse der Klassifizierer werden im Zeitmultiplexverfahren gemäß ihrer jeweiligen Ordnungszahl D^i verschränkt und als Ausgangsfolge des Gesamtsystems ausgegeben.

Im Festplattenlesekanal wird mit Hilfe einer Feedback-Schleife jede bereits getroffene Entscheidung zur Verringerung von Intersymbolinterferenz zurück gekoppelt. Da das jeweils letzte detektierte Symbol den größten Einfluss auf die aktuelle Entscheidung hat, wird genau dieses Symbol zur Feedback-Filterung herangezogen. Die G Detektoren D^i müssen also nicht ihre eigene letzte Entscheidung, sondern die des jeweils vorhergehenden Detektors zur Eliminierung der Inter-Symbol-Interferenz (ISI) benutzen. Abbildung 4.2 illustriert das beschriebene Vorgehen für $M = 3$ und $G = 2$.

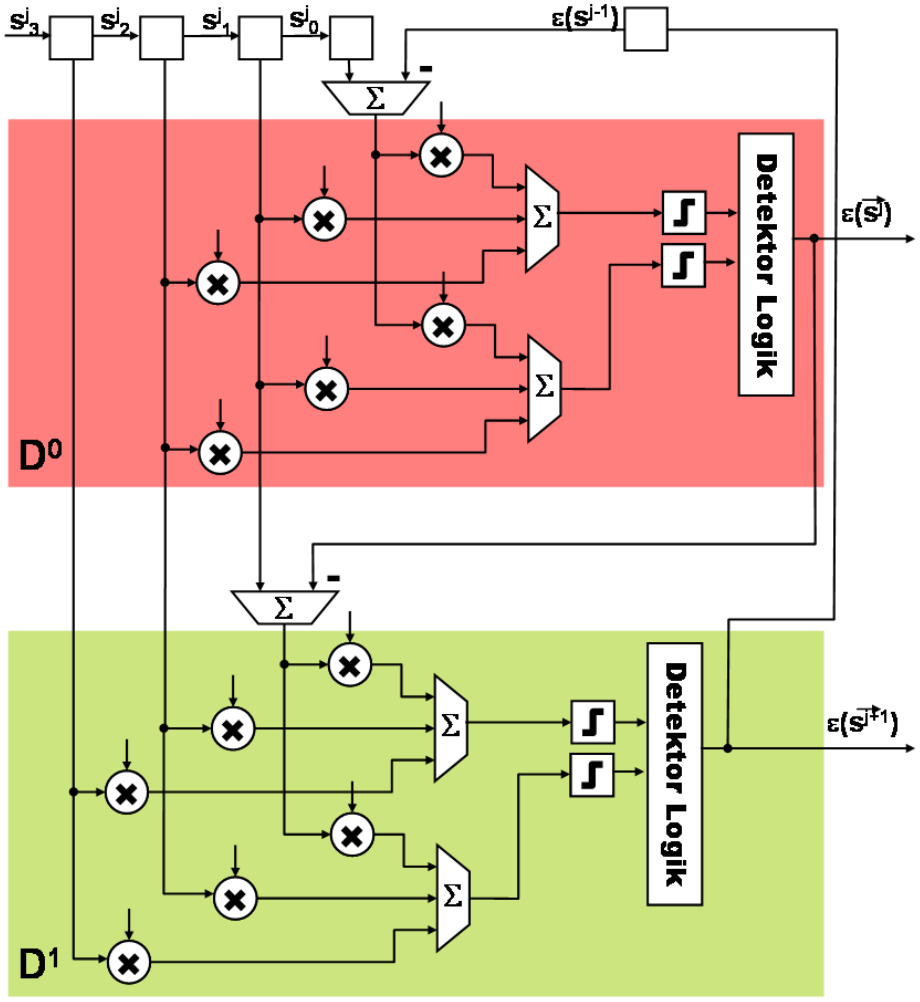

Abbildung 4.2: Prinzipskizze eines MSD ($G = 2$) mit den Teildetektoren D^0 und D^1.

Prinzipiell gibt es bei dieser Vorgehensweise ein Timing-Problem: Der Detektor D^i benötigt als Eingangsgröße das Ergebnis von $D^{(i-1)}$. Dieses liegt aber erst dann an, wenn die Klassifizierung des Teildetektors $D^{(i-1)}$ komplett abgeschlossen ist. Eine Ausnahme bildet D^0, der das zeitunkritische Ergebnis von $D^{(G-1)}$ des *vorherigen* Taktes benötigt.

Die Gesamtlaufzeit t_{ges} des MSD setzt sich also aus der Gatterlaufzeit t_G des unabhängigen Schaltungsteils und der Laufzeit t_L des abhängigen Schaltungsteils zusammen.

$$t_{ges} = t_G + (G-1) \cdot t_L \tag{4.1}$$

Bezogen auf Abbildung 4.2 kann der rot hinterlegte Schaltungsteil unabhängig ausgeführt werden und hat die Laufzeit t_G. Parallel dazu können die Operationen der im Bild ersten vier Multiplizierer von links, die dem Teildetektors D^1 zuzuordnen sind, ausgeführt werden. Der restliche Schaltungsteil im grün hinterlegten Bereich ist abhängig davon, wann $\varepsilon(s^j)$ anliegt und dauert t_L. Eine hohe Laufzeit t_L der abhängigen Komponenten schlägt sich $(G-1)$-fach in der Gesamtlaufzeit nieder. Um dieses Timingproblem zu lösen, muss die Laufzeit t_L so stark reduziert werden, dass sie im Vergleich zu t_G vernachlässigbar klein wird. Hierzu bieten sich folgende Möglichkeiten an:

- **Spekulative Berechnung.** Durch Replikation der Hardware des abhängigen Schaltungsteils kann das Ergebnis für jeden möglichen Feedbackwert (hier: $+1$ bzw. -1) spekulativ ermittelt werden, ohne dass dieser Wert tatsächlich bereits vorliegt. Sobald das Ergebnis des vorgehenden Detektors feststeht, kann über einen Dekoder das korrekte Zwischenergebnisse durchgeschaltet werden. Dieser Fall ist in Abbildung 4.3 skizziert. Durch die spätere Einkopplung des Ergebnisses von $D^{(i-1)}$ wird der abhängige Schaltungspfad verkürzt; t_L wird zu Lasten von t_G und einem erhöhten Hardwareaufwand verringert. Werden alle $(G-1)$ abhängigen Detektionsergebnisse spekulativ ermittelt, so ergeben sich $1 + 2^{(G-1)}$ Ausführungseinheiten. Somit erfordert diese Variante, insbesondere für große G, einen hohen Hardwareaufwand.

- **Algebraische Vereinfachung.** Die alternative Variante besteht darin, die Laufzeit t_L durch mathematische Vereinfachungen zu reduzieren bzw. in den unabhängigen Schaltungszweig *ohne* zusätzlichen Hardwareaufwand zu verschieben. Im Vergleich zur ersten Variante ist dieses Verfahren durch eine längere Laufzeit bei einem geringeren Hardwareaufwand gekennzeichnet. Voraussetzung hierfür ist jedoch, dass eine Reduktion der Laufzeit durch algebraische Vereinfachung des abhängigen Schaltungsteils überhaupt möglich ist.

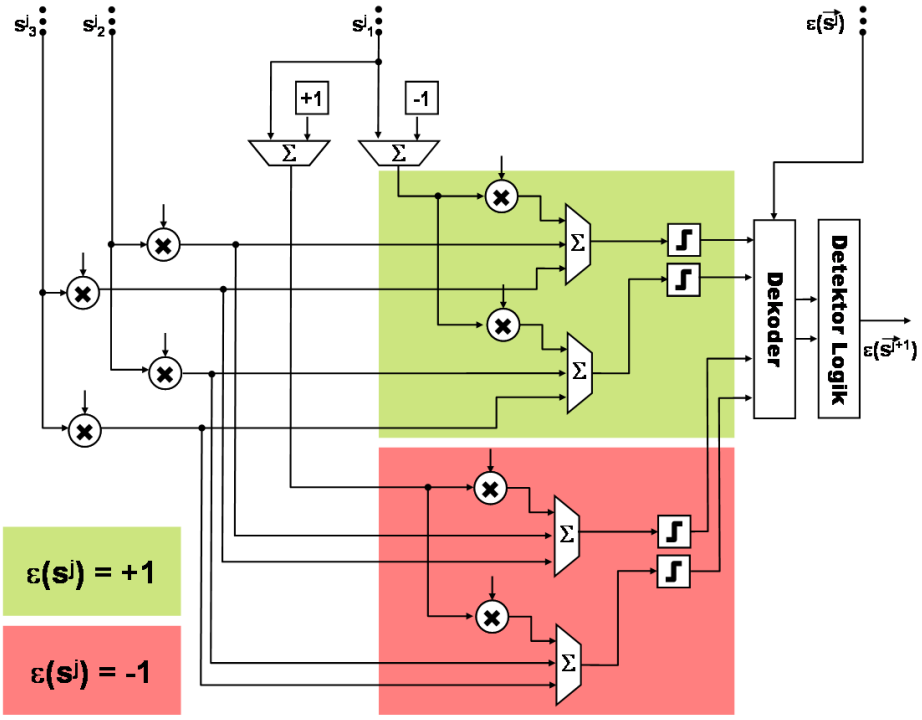

Abbildung 4.3: Prinzipskizze der spekulativen Ausführung im Teildetektor D^1 für $G = 2$ und $M = 3$ in Anlehnung an Abbildung 4.2.

Schließlich bietet sich eine Kombination beider Verfahren an, wobei sich aufgrund des Vorteils des geringeren Hardwareaufwandes die folgenden Untersuchungen vor allem auf die algebraische Vereinfachung konzentrieren.

4.2 Implementierung

Der Signalraumdetektor besteht aus Multiplizierern, Addierern, Schwellwertschaltern und einer Booleschen Algebra (vgl. Abbildung 4.2). Das größte Potential zur Komplexitätsreduzierung liegt in der Eliminierung von Multiplizierern. Dem kommt unter der Berücksichtigung von Laufzeit und Chipfläche ohnehin eine besonders hohe Bedeutung zu: Bei der Wahl eines Multiplizierertyps muss sich der Designer für einen Kompromiss zwischen Geschwindigkeit und benötigter Chipfläche entscheiden. Unter der Zielsetzung einer möglichst schnellen Detektion, muss der Designer auf schnelle und somit aufwändige bzw. teure Hardware zurückgreifen, denn Multiplizierertypen mit geringem Platzbedarf haben eine vergleichsweise lange Gatterlaufzeit [Str02]. Eine Senkung des Zeitbedarfs eines Multiplizierers kann daher nur durch überproportionalen Einsatz von Chipfläche realisiert werden.

Da der Anwendungsfall ohnehin eine hohe Durchsatzrate erfordert, ist davon auszugehen, dass die verwendeten Multiplizierer viel Chipfläche in Anspruch nehmen und somit hohe Kosten verursachen. In Tabelle 4.1 wird der Zusammenhang zwischen Gatterlaufzeit und Chipfläche für verschiedene Multiplizierer verdeutlicht und mit einem 8-Bit-Addierer verglichen:

Bauelement	Bauform	Fläche	Laufzeit
Multiplizierer	sequentiell	512	2112
Multiplizierer	Carry-Save	1472	255
Multiplizierer	parallel	15516	154
Multiplizierer	Wallace-Tree	17238	85
Addierer	Carry Ripple	112	15
Addierer	Carry Look Ahead	104	7

Tabelle 4.1: Normalisierter Flächenbedarf und normalisierter Zeitbedarf für verschiedene Schaltungselemente [Str02].

Durch Verringerung der Anzahl der benötigten Multiplizierer kann also einerseits die Laufzeit deutlich reduziert werden. Andererseits wird das Schaltungslayout des Detektors dadurch platz- und energiesparender aufgebaut. Diese Verringerung kann durch folgende schrittweise Optimierungen erreicht werden:

1. Durch Umstellung der Ebenengleichungen können unter Umständen einige Multiplizierer auf Eins oder Null normiert und somit eliminiert werden.

2. Die Wiederverwendung von Zwischenergebnissen kann durch Substitution von Multiplizierern mit Verzögerungselementen erzielt werden, wodurch die Schaltungskomplexität weiter gesenkt werden kann.

3. Ähnliche Gewichte können durch eine Multiplikation mit ihrem Mittelwert ersetzt werden, wodurch die Hardwarekomplexität auf Kosten der Genauigkeit weiter reduziert wird.

Die Ebenengleichungen werden durch Multiplizierer und Schwellwertschalter implementiert. Jeweils M Multiplizierer repräsentieren dabei den Normalenvektor $\vec{w}^j = (w_0^j, w_1^j, \ldots w_{M-1}^j)$ einer Hyperebene. Der zugehörige Schwellwertschalter w_M^j definiert das Maß für den Abstand dieser Ebene vom Ursprung des Koordinatensystems. Sofern sowohl $\vec{w}^j$ als auch w_M^j mit einer Konstanten Γ multipliziert werden, ändert sich die Lage der implementierten Ebene nicht.

4.2.1 Algebraische Vereinfachung

Eine Komplexitätsreduktion ergibt sich insbesondere dann, wenn $\Gamma = (w_i^j)^{-1}$ für $i \in [0, (M-1)]$. Somit kann mindestens ein Konstantenmultiplizierer pro Entscheidungsebene eliminiert werden.

Grundsätzlich ist die Wahl des zu eliminierenden Faktors frei. Um das im Abschnitt 4.1 diskutierte Timing-Problem zu lösen, ist es besonders vorteilhaft, w_0^j zu substituieren:

1. Auf diese Weise werden genau die Multiplizierer eliminiert, die sich im abhängigen Schaltungsteil befinden. Da die Gatterlaufzeit des Multiplizierers gemäß Tabelle 4.1 hoch ist im Vergleich zu Addierer und Boolescher Algebra, können dadurch die Laufzeit t_L des abhängigen Schaltungsteils und somit die Gesamtlaufzeit t_{ges} gesenkt werden.

2. Dadurch, dass (gemäß Abb. 4.2) ein Faktor der Multiplikation, nämlich $s^j - \varepsilon(s^{(j-1)})$, erst bestimmt werden muss, ist die Laufzeit für die jeweils M-te Multiplikation im Teildetektor länger als die aller anderen Multiplikationen, und sie gibt damit die maximale Taktgeschwindigkeit vor. Dies gilt insbesondere auch für den Teildetektor D^0.

Die Eliminierung von w_0^j ist unter Berücksichtigung dieser Seitenbedingungen besonders sinnvoll und daher zu bevorzugen. Eine Prinzipskizze der entsprechenden Schaltungsimplementierung ist in Abbildung 4.4 dargestellt. Die beschriebenen Konstantenmultiplizierer wurden jeweils auf Eins normiert und somit eliminiert. Gleichzeitig wurde die erläuterte spekulative Abstandsbestimmung für den Teildetektor D^1 umgesetzt. Die farbig hinterlegten Schaltungsteile von D^1 bestimmen die Ebenenabstände für die Zwischenergebnisse $\varepsilon = +1$ (grün) bzw. $\varepsilon = -1$ (blau).

4.2.2 Wiederverwendung von Zwischenergebnissen

Die Wiederverwendung gleicher oder ähnlicher Zwischenergebnisse stellt eine weitere Möglichkeit zur Komplexitätsreduktion dar. Eine Komplexitätsreduktion ergibt sich immer dann, wenn die bereits normalisierten Konstanten w_i^a und w_i^b von zwei verschiedenen Entscheidungsebenen oder aber die Komponenten w_i^a und $w_{(i+x)}^a$ einer Entscheidungsebene identisch oder zumindest näherungsweise gleich sind.

Im ersten Fall können die Multiplikationen $w_{(M-1)}^a \cdot s_{(M-1)}$ und $w_{(M-1)}^b \cdot s_{(M-1)}$ ohne zusätzlichen Hardwareaufwand zu einem Produkt zusammengefasst werden. Die Darstellung dieser Modifikation erfolgt im linken Teil von Abbildung 4.5 (auf Seite 82).

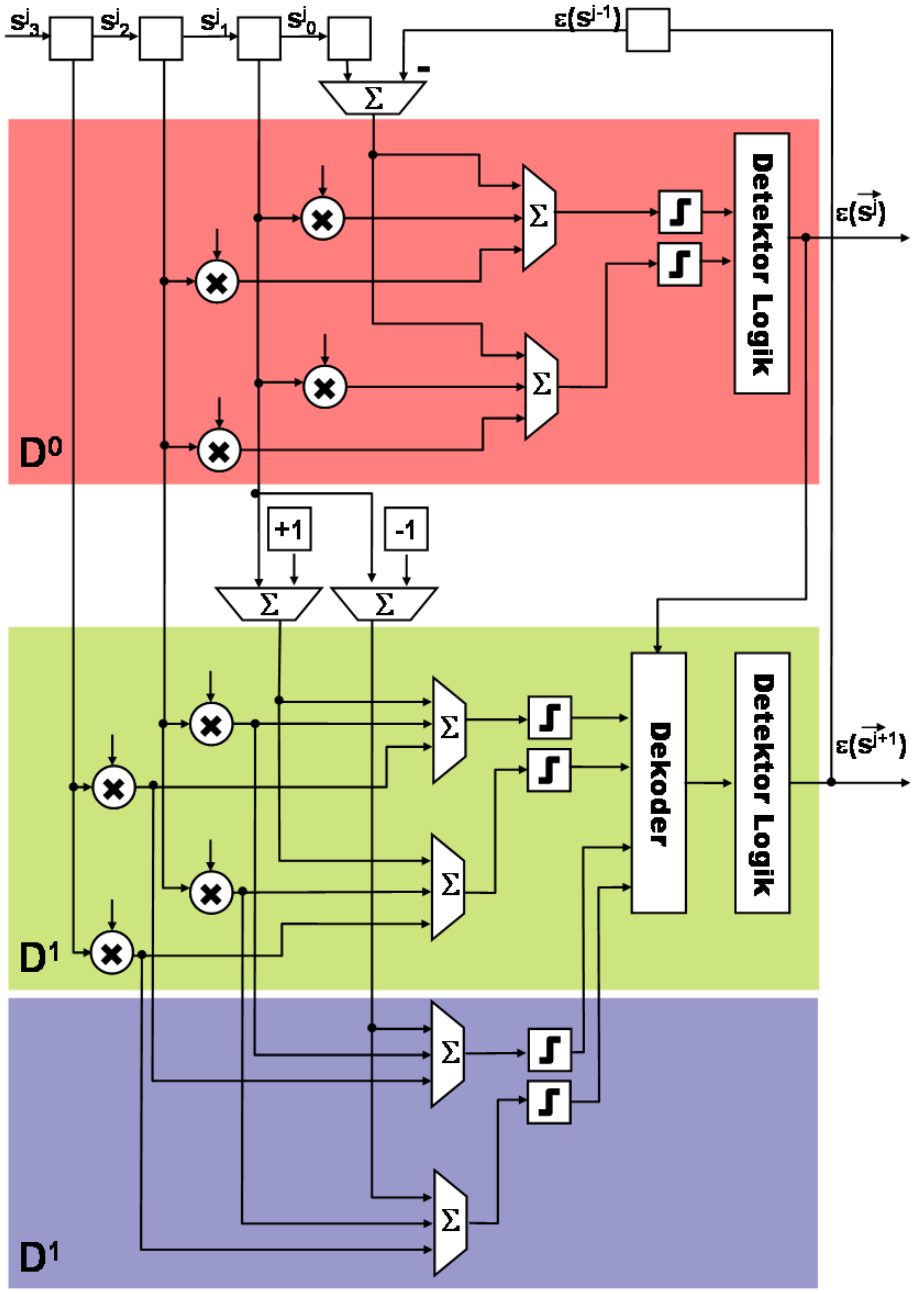

Abbildung 4.4: Multi-Symbol-Detektor mit optimierten Ebenengleichungen und spekulativer Abstandsbestimmung.

Sind die Faktoren w_i^a und $w_{(i+x)}^a$ gleich, so sind zum einen die Produkte $w_i^a \cdot s_i$ im Teildetektor D^j und $w_{i+1}^a \cdot s_i$ im Teildetektor $D^{(j+x)}$ identisch. Da die Eingangsdaten in jedem Taktschritt um G Stellen geschoben werden, muss im nächsten Takt $w_{i+x}^a \cdot s_{i+x}$ des Detektors $D^{(j+G-x)}$ im Detektor D^j nochmals berechnet werden. Dem lässt sich durch Speicherung des Zwischenergebnisses Rechnung tragen. Auch diese Vereinfachung wird in Abb. 4.5 illustriert. Hierzu sei die jeweils „obere“ Ebene mit E^0 und die jeweils „untere“ Ebene als E^1 bezeichnet, so wird davon ausgegangen, dass folgende Koeffizienten gleich sind: $w_3^0 = w_2^0 = w_3^1$. Nun wird $s_3^j \cdot w_3^0$ (links oben im grün hinterlegten Detektor D^1) berechnet, und dieses Produkt wird, um einen Takt verzögert (rot hinterlegter Detektor D^0, Mitte), wieder verwendet, da zu diesem Zeitpunkt die gleiche Eingangsgröße $\langle s_1^j \rangle_t = \langle s_3^j \rangle_{(t-1)}$ anliegt.

Als Fragestellung ergibt sich, bis zu welcher Abweichung es sinnvoll ist, die Gewichtungsfaktoren zusammen zu fassen. Hierzu ist der in Abschnitt 2.1.2 untersuchte Einfluss der Parametervariation auf die Performance bzw. Fehlerrate des Klassifizierers maßgeblich. Wie in Abbildung 2.2 (Seite 39) dargestellt wurde, geht für den Detektor im magnetischen Lesekanal einer Festplatte bei Parametervariation mit $\sigma^2 = 0,03$ diese Störung im Kanalrauschen unter. Als Schlussfolgerung können bei einer Abweichung von unter drei Prozent zwei Koeffizienten als „ähnlich“ betrachtet und zusammen gefasst werden.

4.2.3 Ausführungsbeispiel

Um die bisherigen Ausführungen zu verdeutlichen, wird im Folgenden aus dem so genannten WSSD für eine Pulsweite $PW_{50} = 3.5T$ gemäß [Str00] ein MSD konstruiert. Die für den WSSD optimierten Ebenengleichungen lauten gemäß [SHS00]:

$$\begin{pmatrix} -1.804 & 0.000 & 1.804 \\ -2.565 & -1.535 & -0.761 \\ -2.565 & -1.535 & -0.761 \\ -2.565 & -1.535 & -0.761 \\ -2.603 & -0.737 & 1.006 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \end{pmatrix}^T + \begin{pmatrix} 1.804 \\ 3.056 \\ 1.535 \\ -3.056 \\ -0.269 \end{pmatrix} = 0 \qquad (4.2)$$

Im ersten Schritt werden diese Ebenengleichungen normalisiert, indem mit dem Faktor gekürzt wird, der s_0 wichtet. Die resultierende Ebenenkonstellation lautet dann:

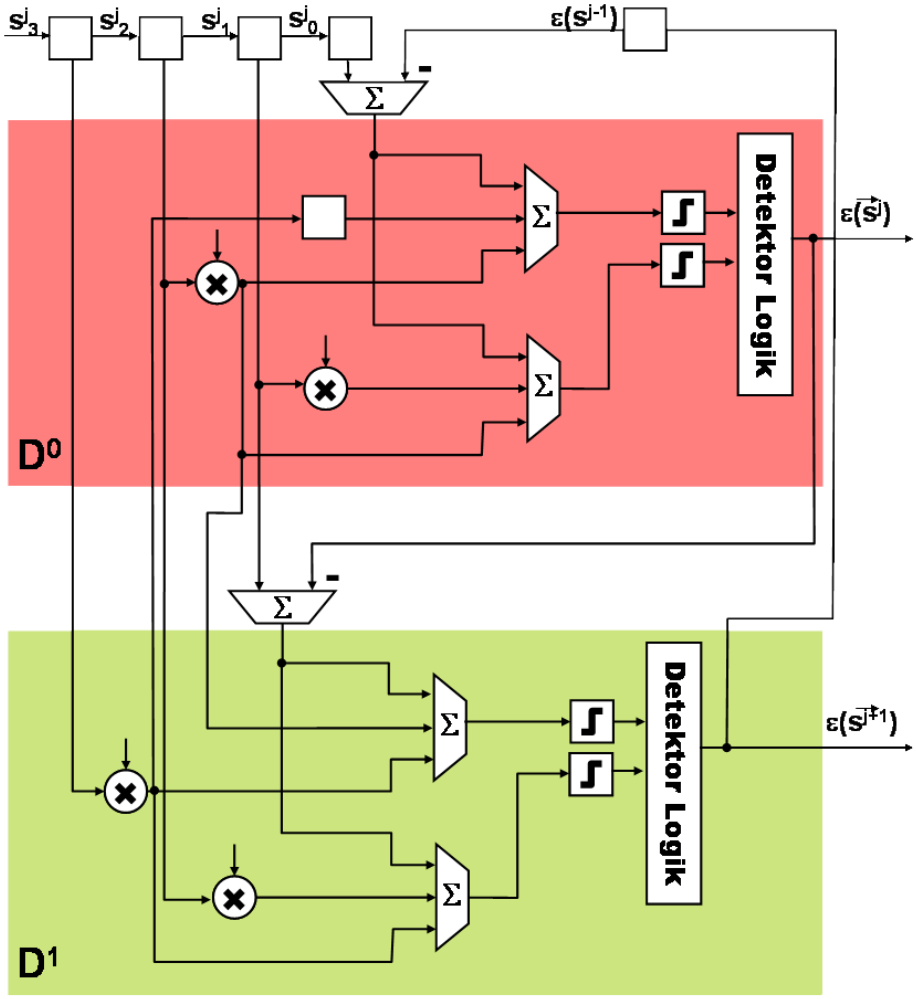

Abbildung 4.5: Multi-Symbol-Detektor mit optimierten Ebenengleichungen und Logik zur gemeinsamen Weiterverarbeitung gleicher Zwischenergebnisse.

$$\begin{pmatrix} 1.000 & 0.000 & -1.000 \\ 1.000 & 0.599 & 0.297 \\ 1.000 & 0.599 & 0.297 \\ 1.000 & 0.599 & 0.297 \\ 1.000 & 0.283 & -0.387 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \end{pmatrix}^T + \begin{pmatrix} -1.000 \\ -1.192 \\ -0.599 \\ 1.192 \\ 0.103 \end{pmatrix} = 0 \tag{4.3}$$

Im zweiten Schritt wird eine Komplexitätsreduktion durch Zusammenfassung ähnlicher Faktoren vorgenommen. Wie bereits hergeleitet, werden ähnliche Koeffizienten zusammengefasst. Im vorliegenden Fall folgt, dass zum einen alle Multiplizierer mit den Faktoren $w_2^i = 0.599$ zusammengefasst werden. Außerdem sind $w_2^i = 0.297$ für $i \in \{2, 3, 4\}$ und $w_1^5 = 0.283$ zu einem neuen Wert $w^* = 0.290$ zusammen zu fassen.

$$\begin{pmatrix} 1.000 & 0.000 & -1.000 \\ 1.000 & 0.599 & 0.290 \\ 1.000 & 0.599 & 0.290 \\ 1.000 & 0.599 & 0.290 \\ 1.000 & 0.290 & -0.387 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \end{pmatrix}^T + \begin{pmatrix} -1.000 \\ -1.192 \\ -0.599 \\ 1.192 \\ 0.103 \end{pmatrix} = 0 \tag{4.4}$$

Der den Ebenengleichungen zugehörige MSD profitiert nun von der Wiederverwendung bereits errechneter Zwischenergebnisse. Das Blockschaltbild einer beispielhaften Umsetzung ist in Abbildung 4.6 dargestellt. Die in der Abbildung der Teildetektoren jeweils links unten befindlichen Multiplizierer mit den Koeffizienten $w_3^j = -1$ lassen sich mittels „strength reduction“ durch Inverter ersetzen. Bei einer besonders vorteilhaften Implementierung in Analogtechnik müssten dazu nur die Leitungen gegeneinander vertauscht werden.

Durch die beschriebene Performanceoptimierung kann bei einem SSD Detektor mit $G = 2$ ein signifikanter Teil der nötigen Hardware eingespart werden. Die neu hinzu gekommenen Verzögerungsglieder sind schaltungstechnisch einfach zu realisieren und zeichnen sich, im Vergleich zu Multiplizierern, durch geringe Leistungsaufnahme und reduzierten Flächenbedarf aus [Str02].

Die Laufzeit des MSD zur Erkennung von *zwei* Symbolen hat sich gegenüber der Laufzeit des WSSD zur Erkennung von *einem* Symbol nur geringfügig erhöht: Lediglich die Gatterlaufzeit eines Addierers und der Booleschen Logik im abhängigen Schaltungsteil sowie die Ausführungszeit des Multiplexers, der die Ergebnisse beider Teildetektoren verschränkt, ist hinzugekommen. Die realisierbare Taktfrequenz hängt von weiteren Design-Entscheidungen ab.

Hierzu wird zunächst abgeschätzt, dass die Berechnung der Booleschen Algebra β in der gleichen Zeit t^+ wie eine Addition erfolgen kann. Weiterhin wird davon ausgegangen, dass eine Quantisierung mittels Schwellwertentscheidern genauso schnell erfolgt, wie eine Addition. Schließlich entspricht diese Operation

der Subtraktion von Schwellwert und Eingangsgröße, wobei nicht die Differenz, sondern nur das Vorzeichen benötigt wird. Die Zeitdauer der Multiplikation wird mit t° bezeichnet. Unter Berücksichtigung der Implementierungsvorschläge laut Abbildungen 4.1, 4.4 und 4.6 ergeben sich folgende Laufzeitabschätzungen:

- **Konventioneller WSSD:** Im Fall eines herkömmlichen WSSD ist die Länge des kritischen Pfades gegeben durch die Zeitdauer folgender aufeinander aufbauender Operationen: Addition, Multiplikation, Addition, Schwellwertbildung, Boolesche Algebra. Dieser Pfad lässt sich in Bild 4.1 durch Verfolgung des Signalweges von s_0^j nach $\varepsilon(\vec{s}^j)$ nachvollziehen. Die Laufzeit lässt sich somit abschätzen als

$$t_{ges} = 4 \cdot t^+ + \cdot t^\circ$$

- **Multi-Symbol-Detektor *ohne* spekulative Ausführung:** Wird ein MSD ohne spekulative Ausführungseinheiten verwendet, so lässt sich der kritische Signallaufpfad aus Abbildung 4.6 wie folgt ableiten: Multiplikation, Addition, Schwellwertbildung, Boolesche Algebra, Addition, Schwellwertbildung, Boolesche Algebra (Weg von s_2^j über D^0 nach $\varepsilon(\vec{s}^{(j+1)})$). Die Gesamtlaufzeit kann infolgedessen mit

$$t_{ges} = \underbrace{t^\circ + t^+ + t^+ + t^+}_{t_G} + (G-1) \cdot (\underbrace{t^+ + t^+ + t^+}_{t_L})$$

  abgeschätzt werden. Setzt man diese ins Verhältnis zu der Zeit, die ein WSSD benötigt, um G Symbole zu klassifizieren, so ergibt sich folgendes Verhältnis der Klassifizierungszeit pro Symbol:

$$\frac{\cdot t^\circ + 3 \cdot t^+ + 2 \cdot (G-1) \cdot t^+}{G \cdot (4 \cdot t^+ + t^\circ)} = \frac{1}{G} + \frac{3 - \frac{4}{G}}{4 + \frac{t^\circ}{t^+}}$$

  Für die Grenzwerte $t^\circ/t^+ \to 1$ und $t^\circ/t^+ \to \infty$ ergeben sich die obere und untere Grenze des Laufzeitverhältnisses als $1/G$ resp. $(3G+1)/(5G)$. Unter Zuhilfenahme des Grenzwertes $G \to \infty$ ergibt sich für den MSD eine Durchsatzrate zwischen $5/3$ und G bezogen auf den konventionellen WSSD.

- **Multi-Symbol-Detektor *mit* spekulativer Ausführung:** Wird der MSD um eine Einheit zur spekulativen Ausführung (in Anlehnung an das in Abb. 4.4 illustrierte Blockschaltbild) erweitert, so können mehr zeitkritische Operationen parallel und unabhängig voneinander verarbeitet werden. Der

kritische Pfad verkürzt sich auf: Multiplikation, Addition, Schwellwertbildung, Boolesche Algebra, Dekoder-Entscheidung, Boolesche Algebra (Weg von s_2^j über D^0 nach $\varepsilon(\vec{s}^{(j+1)})$). Somit beträgt die Laufzeitabschätzung:

$$t_{ges} = \underbrace{t^\circ + t^+ + t^+ + t^+}_{t_G} + (G-1) \cdot (\underbrace{t^+ + t^+}_{t_L})$$

Setzt man t_{ges} ins Verhältnis zur Dauer der Klassifizierung von G Symbolen durch den WSSD, dann erhält man das folgende Verhältnis der Klassifizierungszeit pro Symbol:

$$\frac{t^\circ + 3 \cdot t^+ + 2 \cdot (G-1) \cdot t^+}{G \cdot (4 \cdot t^+ + t^\circ)} = \frac{1}{G} + \frac{1 - \frac{1}{G}}{2 + \frac{t^\circ}{2 \cdot t^+}}$$

Obere und untere Grenze des Laufzeitverhältnisses resultieren abermals aus den Grenzwertbetrachtungen $t^\circ/t^+ \to 1$ und $t^\circ/t^+ \to \infty$ mit $1/G$ bzw. $(2G+3)/(5G)$. Die Abschätzung der Durchsatzrate ergibt sich für den Grenzfall $G \to \infty$ zwischen $2,5$ und G bezogen auf den WSSD.

4.3 Simulationsergebnisse

Durch die in Abschnitt 4.2 ausgeführten Hardware-Optimierungen wurden die Ebenengleichungen des Multi-Symbol-Detektors verändert. Mittels Simulation soll nun eruiert werden, wie stark sich diese Veränderung auf die Qualität der Detektionsergebnisse auswirkt und wie stark sich somit das BER/SNR-Verhältnis vom konventionellen und vom Multi-Symbol-Detekor unterscheidet.

Hierzu wurden Simulationen für den MSD mit $G = 2$ bei einer Pulsweite von $PW_{50} = 3,50T$ im SNR-Bereich von 27 bis 32 dB durchgeführt. Abbildung 4.7 zeigt das zugehörige BER-SNR-Diagramm, in dem die Bitfehlerrate in Abhängigkeit vom Signal-Rausch-Abstand aufgetragen ist. Um mögliche Unterschiede zwischen den beiden Detektoren sichtbar zu machen, sind in dem Diagramm sowohl die Kennlinie für den WSSD als auch für den MSD dargestellt. Der Einfluss des für magnetische Lesekanäle charakteristischen Transitionsrauschens bleibt in diesem Fall unberücksichtigt [AT86, Hug98].

Beide Detektoren verhalten sich, wie Abbildung 4.7 zeigt, nahezu identisch. Kleine Abweichungen sind ursächlich im angewandten Simulationsverfahren: Transitionsrauschen und AWGN werden durch Zufallszahlen simuliert, so dass bereits die EIngangsdaten leicht voneinander abweichen, statistisch jedoch gleich charakterisiert sind [DPS01]. Wie bereits prognostiziert, geht die Modifikation der Ebenenparameter im Kanalrauschen unter.

Schließlich zeigt Abbildung 4.8 die Detektionsergebnisse von MSD und WSSD unter Berücksichtigung des für Festplattenlesekanäle charakteristischen Jitterrauschens. Es ist erkennbar, dass sich beide Detektoren auch unter Einfluss von Jitter nahezu identisch verhalten.

Insgesamt lässt sich feststellen, dass die Performance des konzipierten MSD nahezu gleich ist mit der des konventionellen WSSD, und das bei bis zu G-fach höherer Datendurchsatzrate. Somit wurde ein effizientes Klassifikationskonzept entwickelt, das in der Lage ist, eine deutlich höhere Durchsatzrate bei gleich bleibend guter Performance zu erzielen.

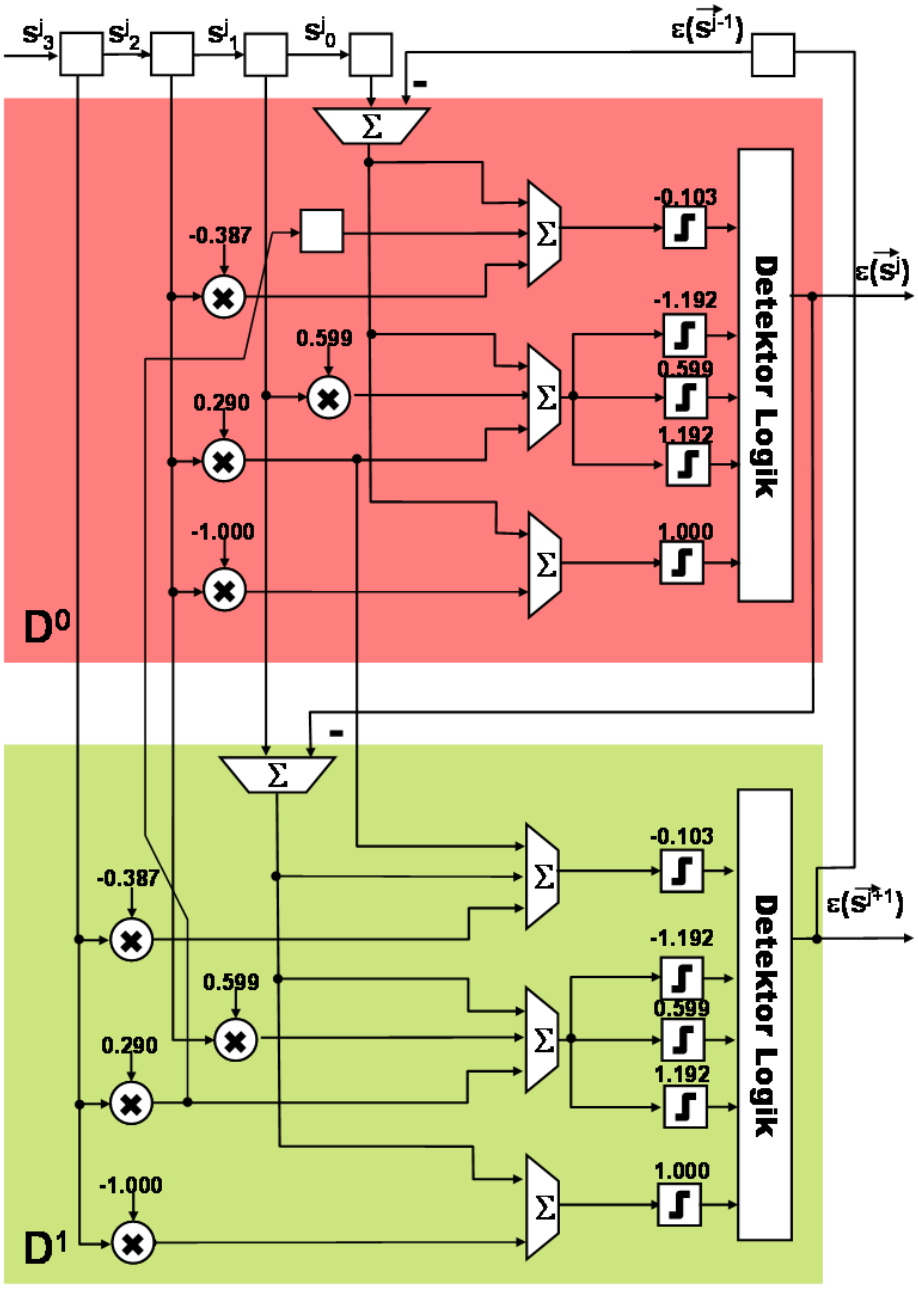

Abbildung 4.6: Implementierungsbeispiel eines komplexitätsreduzierten MSD mit $M = 3, G = 2$ für den Einsatz im Festplattenlesekanal mit $PW_{50} = 3,5T$.

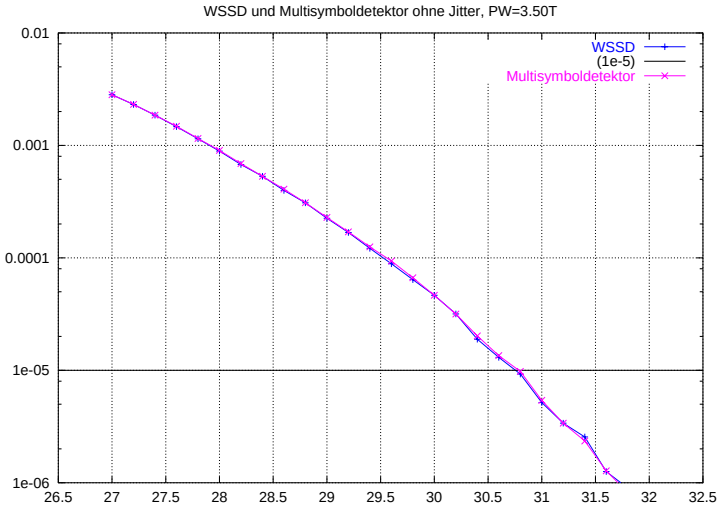

Abbildung 4.7: Fehlerrate-SNR-Diagramm für WSSD und MSD ohne Berücksichtigung von Transitionsrauschen.

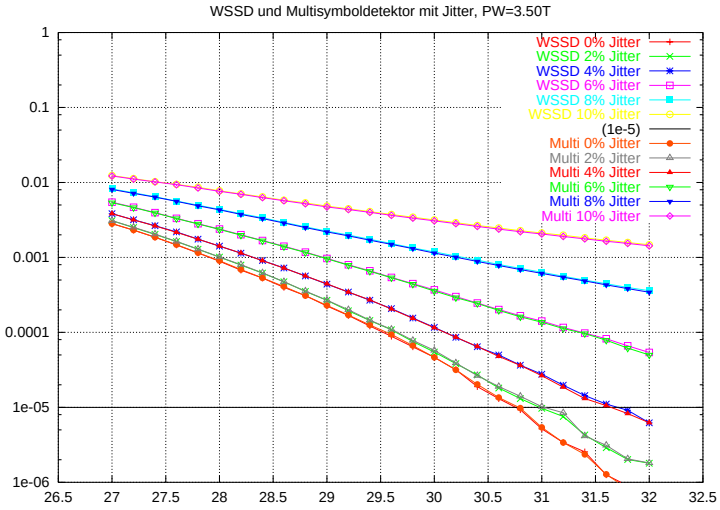

Abbildung 4.8: Fehlerrate-SNR-Diagramm für WSSD und ASSD-Detektor (Multi) unter Einfluss unterschiedlicher Jitterstärken.

Kapitel 5

Kombination mehrerer Verfahren

Besonders effizient ist die Kombination mehrerer der vorgestellten Verfahren in einem komponentenübergreifenden System-Design. Auf diese Weise kann eine besonders präzise und gleichzeitig schnelle Klassifizierung gewährleistet werden. Zum Beispiel ist es möglich, dass die durch adaptive Komponenten eingebüßte Laufzeit durch den Einsatz der Multi-Symbol-Detektion kompensiert wird. Die Leistungsfähigkeit von miteinander kombinierten Verfahren und Vorgehensweisen wird an dem folgenden praktischen Beispiel vorgestellt und analysiert.

5.1 Klassifizierung von Laufmustern

Ausgangspunkt sind vierbeinige Laufroboter vom Typ Sony Aibo ERS 210, wie sie in Abbildung 5.1 dargestellt sind. Diese Roboter verfügen über drei Motoren je Bein, drei weitere Motoren im Kopf, zwei Motoren im Schwanz, einen pro Ohr und einen am Maul. Somit besitzen diese Roboter insgesamt 20 Freiheitsgrade, über die jede vom Roboter eingenommene (Körper)-Haltung exakt beschrieben werden kann [DZ02b].

Damit sich der Roboter fortbewegen kann, ist eine Software zu implementieren, die die 20 Motoren genau so ansteuert, dass die resultierende Sequenz von angesteuerten Gelenkwinkeln ein funktionierendes Laufmuster ergibt. Laufmuster können durch verschiedene Merkmale, z.B. Vorwärtsgeschwindigkeit, Eleganz und Energiebedarf, charakterisiert werden. Laufmuster werden als – typischerweise periodische – Folge von Gelenkwinkelkombinationen erzeugt [Cru91].

Ist ein Lauf periodisch mit der Periodendauer t, so kann er dadurch beschrieben werden, dass mit jedem Bein innerhalb t eine vorgeschriebene Bahnkurve abgefahren wird. Eine einfache Form, eine Bahnkurve zu beschreiben, besteht darin, Stützpunkte zu definieren, die zu bestimmten Zeitpunkten angesteuert werden. Werden beispielsweise fünf Stützpunkte für jedes der vier Beine verwendet, welche sich durch jeweils drei Motoren ansteuern lassen, so lässt sich ohne

Abbildung 5.1: Der vierbeinige Laufroboter Sony Aibo ERS 210.

Berücksichtigung von Symmetrien der resultierende Lauf durch $5 \cdot 4 \cdot 3 = 60$ Parameter charakterisieren.

Aufgabenstellung bei der Laufmustergenerierung ist es nun, diesen 60-dimensionalen Vektor derart mit sinnvollen Werten zu besetzen, dass ein Laufmuster mit ausgezeichneten Eigenschaften entsteht, z.B. einer besonders hohen Vorwärtsgeschwindigkeit. Ein wichtiger Schritt hierzu ist die Reduktion der Dimension und des Wertebereiches der einzelnen Merkmale im $\underline{\underline{R}}^{60}$. Dies kann zum Beispiel dadurch geschehen, dass Abhängigkeiten zwischen einzelnen Parametern berücksichtigt werden. So beschreiben die jeweils linken und rechten Pfoten von Vierbeinern beim Laufen typischerweise symmetrische Bahnkurven. Gleichfalls sind aufgrund der physikalischen Abmessungen die Schrittlängen von Vorder- und Hinterbein beschränkt und miteinander korreliert.

Derartige Abhängigkeiten sind in einer so genannte Walking-Engine definiert. Somit ist die Walking Engine in der Lage, durch Eingabe eines Merkmalsvektors niedriger Dimension ein Laufmuster höherer Dimension zu generieren, welches den vorgegebenen Abhängigkeiten Rechnung trägt. Ein Beispiel für eine solche

Anz.	Σ	Parameter	Typ
1	1	footmode	binary
3	4	foreHeight, foreWidth, foreCenter	float
3	7	hindHeight, hindWidth, hindCenter	float
2	9	foreFootTilt, hindFootTilt	float
2	11	foreFootLift, hindFootLift	float
3	14	legSpeedFactorX legSpeedFactorY legSpeedFactorR	float
2	16	maxStepSizeX, maxStepSizeY	float
3	19	maxSpeedXChange maxSpeedYChange maxRotationChange	float
1	20	counterRotation	float
1	21	stepLen	int
1	22	groundPhase	float
1	23	liftPhase	
4	27	headTilt, headPan, headRoll, mouth	float

Tabelle 5.1: Parametersatz der so genannten „Inverse Kinematic Walking Engine“ [Ris02].

Walking-Engine ist die vom GermanTeam benutzte „Inverse Kinematik Walking Engine“ (IKWE) [JLB+03, Ris02]. Unter Verwendung der IKWE kann ein 27-dimensionaler Parametervektor verwendet werden, um ein Laufmuster eindeutig zu charakterisieren. Einen Überblick über die einzelnen Parameter der IKWE gibt Tabelle 5.1.

Auch unter Verwendung einer Walking-Engine hat der den Lauf beschreibende Merkmalsvektor eine sehr hohe Dimension. Daher ist es sehr schwierig, Merkmalsvektoren resp. Parametersätze zu bestimmen, die ausgezeichnete Laufmuster beschreiben [CDH03].

5.2 Parameteroptimierung mit evolutionären Algorithmen (EA)

Wie bereits in Abschnitt 2.2 (Seite 43f.) vorgestellt wurde, ist durch Anwendung von EA ein geeignetes Verfahren gegeben, Parametersätze hoher Dimension so zu optimieren, dass der Algorithmus, welcher von ihnen parametrisiert wird, eine gegebene Fitnessfunktion besonders gut erfüllt. Es liegt also nahe, Laufmuster durch die Anwendung von Evolutionsstrategien zu finden bzw. zu optimieren. Die

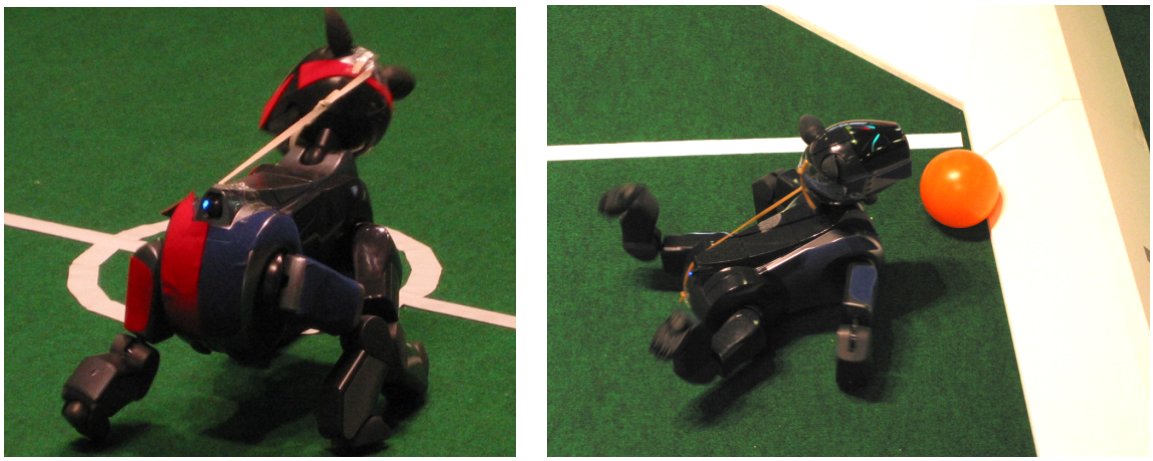

Abbildung 5.2: Sony ERS 210 mit zusätzlichem Dämpfer in Form eines zwischen Rumpf und Kopf montierten Gummibandes.

Verwendung von EA zum Finden derartiger Laufmuster hat sich in der Praxis bereits vielfach bewährt [BZA+02, Spe94, HTY+99]. Darüber hinaus werden EA auch zur komplexen Robotersteuerung verwendet [DBB98, OBN96].

Auch für den Sony Aibo wurden bereits stabile und schnelle Gangarten mit evolutionären Ansätzen entwickelt [DZ02a, DJL+02, DZ03]. Dabei ist eine hohe Geschwindigkeit eine besonders wichtige Anforderungen an das zu erzeugende Laufmuster [LFS92]. Für diesen Fall ist die Implementierung einer Fitnessfunktion sehr einfach. Unter der Prämisse, einen schnellen Geradeauslauf zu entwickeln, kann die Fitness eines Laufmusters durch das physikalische Experiment eines Wettlaufes bestimmt werden, bei dem die vom Roboter benötigte Zeit gemessen wird, um eine vorgegebene Strecke zurück zu legen. Die Fitness kann dabei absolut (Geschwindigkeit in cm/s) oder relativ zu einem Vergleichsmuster bestimmt werden. In letzterem Fall wird das schnellste Individuum einfach durch das zyklische Ausführen von Wettläufen gefunden, z.B. in Form eines Turniers.Das schnellere von je zwei Individuen hat dann die höhere Fitness[1].

Unglücklicherweise verursachen solche Wettläufe und Turniere einen starken Verschleiß der Gelenke und Motoren der Roboter [DZ02b, ZBBB02]. Ursächlich sind die während der Evolution entstehenden, zum Teil ungeeigneten Parametervektoren, die in hakeligen, schleifenden und abrupten Bewegungen resultieren. Nicht modellierte Einschränkungen der Bewegungsfreiheit können sich ebenfalls negativ auf die Lebensdauer von Getriebe, Lager und Motoren auswirken. Beispielsweise ist in EA-Experimenten mit der IKWE zu beobachten, dass die Knie von Vorder- und Hinterbeinen während der Laufevolution aneinander schlagen. Ein weiteres Problem bei der Evolution von Laufmustern mit dem Sony Aibo

[1]Das akademische Problem, dass zwei Parametervektoren exakt gleich schnelle Läufe hervorrufen, wird dadurch gelöst, dass das Los über den Sieger entscheidet.

stellt der vergleichsweise schwere Kopf dar, der nicht nur ungeeignet ist, Stöße abzufangen, sondern diese sogar über den Hebel des Halsgelenkes noch verstärkt auf die Lager überträgt.

Für eine sinnvolle Evolution sind viele Wettläufe erforderlich, da in jedem Evolutionszyklus alle Individuen der Population mindestens einen Lauf ausführen müssen. Beim ERS 210 leidet daher insbesondere das ohnehin schwach ausgelegte Halsgelenk unter den starken Erschütterungen, was schließlich dazu führt, dass das Halsgelenk bricht. Selbst Maßnahmen, die die Stöße dämpfen, wie die (in Abbildung 5.2 dargestellten) zwischen Kopf und Rumpf befestigten Gummibänder, mindern den Verschleiß nur unwesentlich.

Es ist also eine Herausforderung, für den Sony Aibo geeignete Laufmuster mit möglichst wenig physikalischen Experimenten zu finden, um die empfindlichen Halsgelenke der Roboter zu schonen.

Eine effiziente, mehrfach diskutierte Methode, um physikalische Experimente zu ersetzen, ist der Gebrauch einer physikalischen Simulation. Diese konnte in der Vergangenheit – insbesondere für andere Robotertypen – erfolgreich eingesetzt werden [ZBBB02]. Allerdings ist die Implementierung einer derartigen Simulation komplex, aufwändig und daher teuer. Außerdem sind speziell beim Sony ERS 210 die Charakteristika von Motoren, Gelenken und Antrieben weitgehend unbekannt. Daher scheidet diese Variante für den im Experiment eingesetzten Roboter aus.

5.3 Klassifizierung mittels Signalraumdetektor

Eine Alternative zur Simulation stellt die Bereitstellung eines Klassifizierers dar, der aufgrund der Lage der Parametervektoren im Merkmalsraum eine Vorauswahl trifft, welche Laufmuster für die Wettläufe in Betracht kommen und welche nicht [KM98a]. Zu diesem Zweck wird die Verwendung des adaptiven Signalraumdetektors vorgeschlagen, der darüber hinaus die Möglichkeit der Extraktion von Zuverlässigkeitsinformation bieten soll.

Ziel ist es dabei, den Klassifizierer zunächst die manuelle Entscheidung über den Sieger eines Wettlaufs erlernen zu lassen, um diese Entscheidung für künftige Laufmuster automatisch zu generieren, ohne jeden Wettlauf tatsächlich ausführen zu müssen. Dabei wird die Geschwindigkeit von je zwei Laufmustern relativ zueinander erfasst. Die dabei untersuchten Parametervektoren lassen sich in zwei Klassen einteilen:

1. überlegene Individuen $\hat{\varepsilon}(\vec{s^i}) = k^{sup}$ und
2. unterlegene Individuen $\hat{\varepsilon}(\vec{s^i}) = k^{inf}$.

Während alle $s^i \in k^{inf}$ aussortiert werden, bildet k^{sup} die Grundlage für den nächsten Evolutionszyklus. Diese Klassifizierung soll so weit wie möglich automatisch vom AS3D-Detektor vorgenommen werden können. Im späteren Verlauf der

Evolution wird das aktuell „fitteste“ Laufmuster als Vergleichskriterium genommen und mit jedem neu erzeugten Individuum auf Basis der 1+1 Evolutionsstrategie verglichen. Ansonsten werden jeweils zwei zu vergleichende Laufmuster als $M = 27$ dimensionale Merkmalsvektoren betrachtet und in den AS3D-Detektor gespeist (linker Bereich von Abbildung 5.4, Seite 95).

Es wird davon ausgegangen, dass von ähnlichen Laufmustern Cluster im Merkmalsraum gebildet werden, die mittels Signalraumdetektion in c^{sup} und c^{inf} zu klassifizieren sind. Diese Situation ist in Abbildung 5.3 beispielhaft illustriert: Die überlegenen Individuen (rot) und die unterlegenen Individuen (blau) sind durch Hyperebenen voneinander separiert. Der Subraum, in dem sich die überlegenen Individuen befinden ($\vec{s}_i \in c^{sup}$), ist rötlich schattiert.

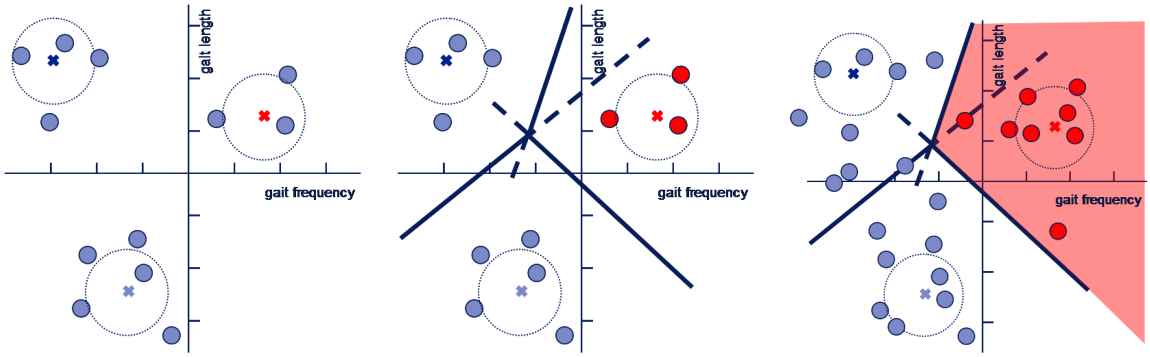

Abbildung 5.3: Merkmalsraum nach manueller Klassifizierung (links), Entscheidungsebenen (Mitte), Klassifizierung unbekannter Parametersätze (rechts).

5.4 Zuverlässigkeitsbestimmung mit AS3D

Wie in Abbildung 5.3 veranschaulicht, liegen einige Individuen in der Nähe einer Entscheidungsebene. Andere Individuen befinden sich in der Nähe des Zentrums eines Clusters und haben einen großen Abstand zu den angrenzenden Hyperebenen. Somit kann (analog zu Abschnitt 3.3.1, Seite 3.3.1ff.) die Zuverlässigkeitsinformation jeder Entscheidung extrahiert werden. Insbesondere kann sie auch dann (als niedrig) bestimmt werden, wenn ein völlig neues Laufmuster entwickelt wurde, das sich abseits bestehender Cluster in $\underline{\underline{R}}^M$ befindet.

Somit erscheint es plausibel, dass eine geeignete Nachbearbeitung der Daten, basierend auf den extrahierten Informationen, durchgeführt werden kann. Hierzu wird das in Abbildung 5.4 illustrierte Vorgehen vorgeschlagen:

1. Wettläufe werden mit verschiedenen Laufmustern durchgeführt.
2. Die Laufmuster werden *manuell* in k^{sup} und k^{inf} klassifiziert.

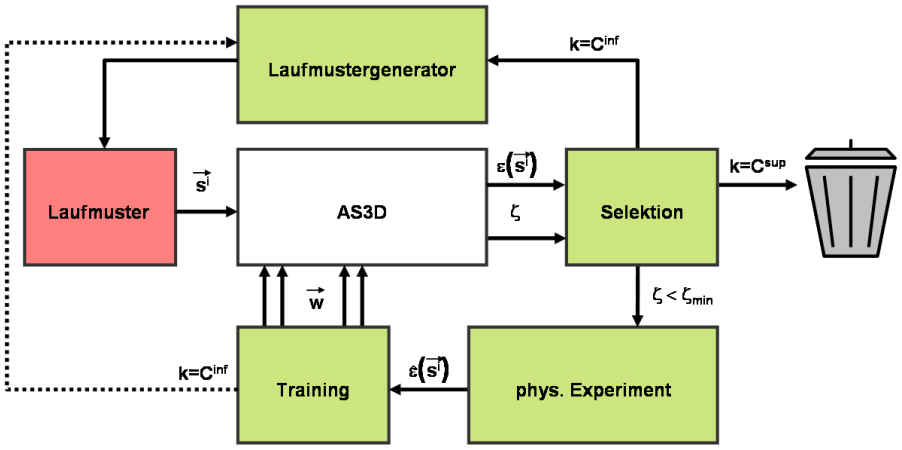

Abbildung 5.4: Prozessskizze des Evolutionsprozesses unter Verwendung des AS3D-Detektors.

3. Ein AS3D-Detektor wird so trainiert, dass er die Trainingsmenge korrekt klassifiziert.

4. Neue Laufmuster werden automatisiert erzeugt.

5. Der AS3D-Detektor klassifiziert die Laufmuster in k^{sup} und k^{inf} und bewertet die Zuverlässigkeit ζ der Entscheidung.

6. Entscheidungen $\varepsilon(\vec{s}^{\,i}) = k^{inf}$ mit einer hohen Zuverlässigkeit $\zeta > \zeta_{min}$ führen zum Löschen des betreffenden Laufmusters (weiter bei Schritt 4).

7. Individuen $\varepsilon(\vec{s}^{\,i}) = C^{sup}$, deren Klassifizierung mit einer hohen Zuverlässigkeit $\zeta > \zeta_{min}$ gekennzeichnet ist, werden direkt zum nächsten Evolutionszyklus übernommen (weiter bei Schritt 4).

8. Unzuverlässige Entscheidungen $\zeta \leq \zeta_{min}$ werden unabhängig vom Ergebnis der Klassifizierung experimentell überprüft und dienen der weiteren Adaption der Entscheidungsebenen (weiter bei Punkt 1).

Werden hinreichend viele Laufmuster korrekt klassifiziert, so kann auf diese Weise eine Vielzahl von physikalischen Experimenten (Wettläufen) entfallen und somit der Verschleiß der Roboter reduziert werden. Da die manuell durchzuführenden Experimente auch relativ viel Zeit in Anspruch nehmen, kann – eine genügend geringe Fehlerrate vorausgesetzt – der Evolutionsprozess beschleunigt werden. Alternativ könnte ebenso die Populationsgröße ohne Zeitverlust erhöht werden. Dies kann gegebenenfalls zu einer weiteren Verbesserung der Qualität der so erzeugten Individuen (hier: Laufmuster) führen [Koz92].

5.5 Simulationsergebnisse

Der vorgeschlagene komplexe Systemaufbau aus evolutionärem Algorithmus, Klassifizierer und Modul zur Verarbeitung von Zuverlässigkeitsinformation wurde zur Findung von schnellen Laufmustern für den Sony Aibo erfolgreich eingesetzt [DZ02c].

Hierzu wurde eine Populationsgröße von 20 Individuen und 100 Generationen verwendet. Die während der Evolution erstellten 2.000 Laufmuster wurden in 1.000 Wettläufen bewertet [DZ02b]. Dabei kam der beschriebene Ansatz zur „Fitnessbestimmung mittels Durchführung von Wettläufen“ zur Anwendung. Die Ergebnisse der Wettläufe wurden als Trainingsdaten für das neuronales Netz eines AS3D-Detektors verwendet. Da die Referenzpunkte a priori unbekannt sind, müssen sie geschätzt werden. Als Schätzwert für den Referenzpunkt eines Unterraumes wird der Schwerpunkt aller Vektoren innerhalb dieses Unterraumes verwendet. Die einzelnen Unterräume ergeben sich dabei beispielsweise durch die Partitionierung mittels OFF-ASSD.

Weiterhin wird die zur Bestimmung der Zuverlässigkeitsinformation erforderliche Varianz des Rauschens dadurch geschätzt, dass diejenigen $\vec{s}^i$, die sich im gleichen Unterraum wie ein Referenzpunkt $\vec{r}^j$ befinden, als $\vec{r}^j + \vec{\eta}$ modelliert werden und somit die Varianz σ_η^2 rechnerisch bestimmt werden kann.

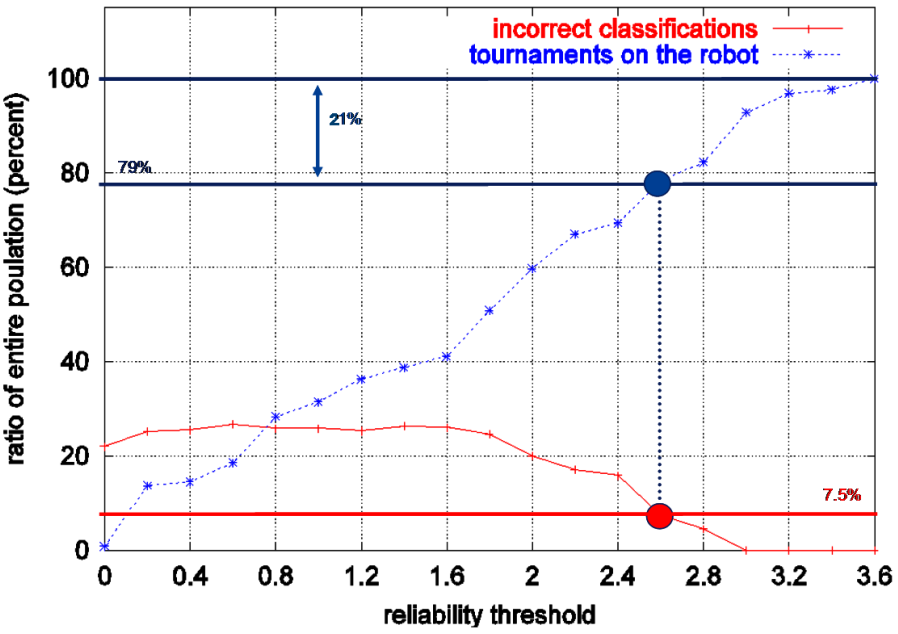

Abbildung 5.5: Anteil fehlerhafter Klassifizierungen (rote Linie) und Anteil der auf dem Roboter durchzuführenden Experimente (blaue Linie) in Abhängigkeit von der Zuverlässigkeitsinformation ζ.

In Abbildung 5.5 ist als rote Linie der Anteil fehlerhafter Klassifizierungen in Abhängigkeit von der Zuverlässigkeitsinformation ζ angegeben. Es wird deutlich, dass ab einer Zuverlässigkeitsinformation $\zeta_{min} > 1,8$ der Anteil der fehlerhaft klassifizierten Laufmuster abnimmt. Dies beweist die Plausibilität der extrahierten Zuverlässigkeit: Je höher der extrahierte Wert ζ, desto geringer ist die Wahrscheinlichkeit, dass eine fehlerhafte Klassifizierung vorliegt. Für $\zeta < 0,6$ ist die Zuverlässigkeitsinformation jedoch unbrauchbar, denn der statistische Anteil fehlerhafter Klassifizierungen überwiegt offenbar die durch niedrige Zuverlässigkeitsinformationen markierten Klassifikationsfehler.

Weiterhin wird im Diagramm 5.5 als blaue Linie der Anteil der auf dem Roboter durchzuführenden Turniere in Bezug auf alle dem Klassifizierer vorgelegten Laufmuster dargestellt. Es kann festgestellt werden, dass bei einem hohen Wert ζ_{min} vergleichsweise viele Experimente auf dem Roboter ausgeführt werden müssen. Dies ist als Konsequenz daraus zu verstehen, dass nur vergleichsweise wenige Parametervektoren mit einer besonders hohen Zuverlässigkeit klassifiziert werden können.

Weiterhin gilt Monotonie: Je höher der Grenzwert ζ_{min} ist, der zwischen „zuverlässigen“ und „unzuverlässigen“ Entscheidungen trennen soll, desto mehr Wettläufe sind auf dem Roboter auszuführen. Wird beispielsweise – wie in Abbildung 5.5 illustriert – der Schwellwert $\zeta_{min} = 2,6$ gewählt, so sind weniger als $7,5\%$ derjenigen vom AS3D-Detektor getroffenen Entscheidungen, für die $\zeta > \zeta_{min}$ gilt, fehlerhaft. Eine so hohe Zuverlässigkeit haben dabei 21% aller Entscheidungen, so dass eben diese Anzahl an Wettläufe eingespart werden kann.

Der Zusammenhang zwischen dem Anteil von fehlerhaften Klassifizierungen und dem Anteil eingesparter Experimente ist in Abbildung 5.6 dargestellt. Hierbei ist die bereits getroffene Feststellung zu berücksichtigen, dass für sehr kleine ζ die Zuverlässigkeitsinformation unbrauchbar ist (ganz linker Teil der Abbildung). Man kann erkennen, dass um so weniger Experimente eingespart werden können, je niedriger die vom Nutzer tolerierte Fehlerrate der Klassifizierung ist. Weiterhin wird ersichtlich, dass eine Fehlerrate von (nahe) Null nur unter der Voraussetzung erzielbar ist, dass (fast) keine Experimente eingespart werden können. Demnach muss der Nutzer eine Fehlerrate ($0 < \rho^{ER} < 0,25$) akzeptieren, wenn ein signifikanter Anteil Experimente eingespart werden soll.

Somit ergibt sich die Frage, wie sich diese dann teilweise fehlerhafte Klassifizierung auf die Fitness der evolvierten Laufmuster auswirkt. Insbesondere ist zu klären, ob trotz der teilweise fehlerhaften Klassifizierung die Laufmuster im Sinne einer schnellen Vorwärtsgeschwindigkeit optimiert werden.

Zur Untersuchung dieser Frage wurde ein Laufmuster mittels 1+1 Evolutionsstrategie bei manueller Fitnessbestimmung optimiert. In einem weiteren Versuch wurde die Fitness der Laufmuster durch das vorgeschlagene Verfahren nach Abbildung 5.4 angewendet. Dabei wurden 10% der als unzuverlässig gekennzeichneten Entscheidungen zufällig ausgewählt, um die zugehörigen Laufmuster auf den Robotern auszuführen. Auf diese Weise wurde ein Laufmuster entwickelt, das ca.

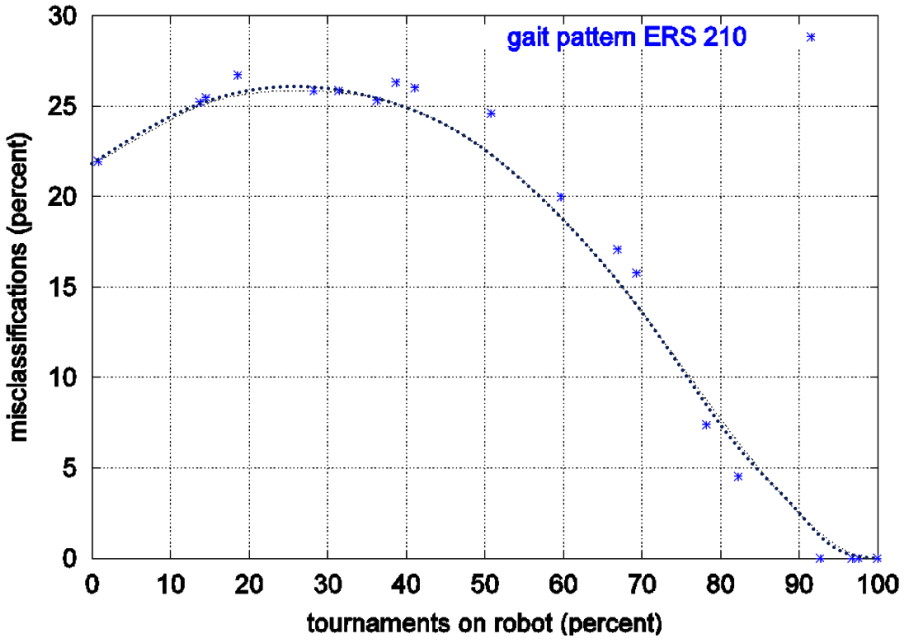

Abbildung 5.6: Zusammenhang zwischen dem Anteil fehlerhafter Klassifizierungen und dem Anteil der auf dem Roboter durchzuführenden Experimente bei jeweils gleichem Schwellwert für die Zuverlässigkeit

20% schneller ist als der Ausgangslauf (23,8 cm/s im Vergleich zu 20cm/s). Das von Hand optimierte Laufmuster war hingegen 28% schneller als der Ausgangslauf (25,6 cm/s).

Durch das vorgeschlagene Verfahren wurde der Verschleiß gegenüber der manuellen EA wesentlich reduziert. Während für die manuelle Evolution 285 Wettläufe durchzuführen waren, mussten unter Zuhilfenahme des vorgeschlagenen Verfahrens nur 34 Experimente manuell bewertet werden.

Somit konnte durch die Kombination der in den Kapiteln 2- 4 (Seite 35- 73) vorgeschlagenen Erweiterungen des Signalraumkonzeptes das eingangs vorgestellte Entwurfsziel, ein schnelles Laufmuster für den Sony Aibo zu entwickeln, erreicht werden, ohne viele Experimente durchführen zu müssen.

Kapitel 6

Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde die Signalraumdetektion als spezieller Klassifizierer untersucht, an verschiedene Anwendungsfelder angepasst und erweitert. Dies sind insbesondere:

- Die Erweiterung des Detektionskonzepts um eine geeignete adaptive Signalvorverarbeitung.
- Die Erweiterung des Detektionskonzepts um eine geeignete Strategie zur Signalraumtransformation zur Verringerung der Schaltungskomplexität.
- Die Erweiterung des Signalraumdetektors um ein Adaptionskonzept für a priori bekannte Kanäle.
- Die Erweiterung des Signalraumdetektors um ein Adaptionskonzept für a priori unbekannte Kanäle.
- Die Erweiterung des Signalraumdetektors zu einem Multi-Symbol-Detektor.

Durch diese Erweiterungen wird die Signalraumtransformation einsetzbar in einer Vielzahl von neuen Anwendungsfällen. Dies wurde an praktischen Beispielen bewiesen. Durch die adaptive Verarbeitung der zu klassifizierenden Daten ist es möglich, wechselnden Rahmenbedingungen Rechnung zu tragen. Dies kann – je nach Anwendungsfall – durchaus wirtschaftliche Vorteile bieten. Beispielsweise kann ein teures Re-Design der Detektorschaltung für den Lesekanal einer Festplatte bei veränderten Kanalparametern durch den Einsatz eines adaptiven Signalraumdetektors vermieden werden.

Die Einführung der Multi-Symbol-Detektion beschleunigt den Klassifizierungsvorgang, da mehrere Klassifizierungen gleichzeitig durchgeführt werden können. Es wurde gezeigt, dass, basierend auf dem vorgestellten Konzept, Schaltungskomponenten eingespart werden können, wodurch z.B. die Verlustleistung der implementierten Schaltung reduziert werden kann.

Die dargelegten Modifikationen des Signalraumdetektors weisen mithin eine Reihe von Leistungsmerkmalen auf, die über die gewünschte Grundfunktionalität „Klassifizierung mit niedriger Fehlerrate“ hinaus gehen. Insbesondere wurden Vorteile gewonnen gegenüber konventionellen Klassifizierern bei der Extraktion von sinnvoll verwertbarer Zuverlässigkeitsinformation, der Geschwindigkeit des Verfahrens sowie der algorithmische Komplexität des Klassifizierers und bei den damit verbundenen Kosten. Diese Vorteile lassen sich, wie folgt, zusammenfassen:

1. Der zu Grunde liegende Algorithmus weist eine geringe Komplexität auf, die sich in einem geringen Leistungsbedarf der eingesetzten Elektronik niederschlägt, falls der Algorithmus als Hardwareschaltung implementiert wird.

2. Während der Klassifizierung können so genannte Zuverlässigkeitsinformationen extrahiert werden, die Rückschlüsse auf die Zuverlässigkeit der vom Detektor getroffenen Entscheidung ermöglichen.

3. Die SSD lässt sich mit geringem Zusatzaufwand so modifizieren, dass in einem Takt mehrere Symbole gleichzeitig klassifiziert werden. Daher ist die SSD geeignet, hohe Durchsatzraten zu gewährleisten.

4. Das Konzept der SSD kann um lernfähige Komponenten erweitert werden, so dass sich der Klassifizierer an wechselnde Rahmenbedingungen und Umwelteinflüsse automatisch anpasst.

Somit ist die Signalraumdetektion ein leistungsfähiges Werkzeug, um den vielfältigen Aufgaben im Bereich der Klassifizierung Rechnung zu tragen.

Künftige Untersuchungen sollten sich zunächst auf die Integration der vorgestellten Konzepte richten. Wie in Kapitel 5 dargestellt, führt gerade die Kombination mehrerer der vorgestellten Verfahren zu einer besonders vorteilhaften Ausprägung des resultierenden Signalraumdetektors. Dabei ist jedoch zu erwarten, dass die Komplexität der zu implementierenden Gesamtschaltung höher ist als die des ursprünglichen Signalraumdetektors. Ebenso ist nicht auszuschließen, dass der Leistungsbedarf mit dieser zusätzlichen Komplexität wächst. Daher ist es z.B. eine Herausforderung zu untersuchen, wie durch ein komponentenübergreifendes Design Verlustleistung eingespart werden kann.

Besonders viel versprechend erscheint dabei der Ansatz, die vorgestellten adaptiven Konzepte mit einem Multi-Symbol-Detektor zu kombinieren. Es ist vorstellbar, dass auf diese Weise der für die Adaption erforderliche zusätzliche Leistungsbedarf bei der Multi-Symbol-Detektion wieder eingespart werden kann. Ob sich dies praktisch erzielen lässt, sollte ebenfalls ermittelt werden. Schließlich ist die Integration einer effizienten Bestimmung der Zuverlässigkeitsinformation in den Multi-Symbol-Detektor ein weiteres nützliches Forschungsthema.

Literaturverzeichnis

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[And95] James A. Anderson. *An Introduction to Neural Networks*. MIT Press. Boston, 1995.

[Ang94] Peter J. Angeline. Genetic Programming and Emergent Intelligence. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 4, pages 75–98. MIT Press, 1994.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers*. Addison-Wesley Pub Co, January 1986.

[AT86] T. C. Arnoldussen and H. C. Tong. Zigzag Transition Profiles, Noise, and Correlation Statistics in Highly Oriented Longitudinal Film Media. *IEEE Transactions on Magnetics*, 22(5):889, September 1986.

[AVT+99] Minoru Asada, Manuela Veloso, Milind Tambe, Itsuki Noda, Hiroaki Kitano, and Gerhard K. Kraetzschmar. Overview of RoboCup 1998. *Lecture Notes in Computer Science*, 1604:1ff., 1999.

[BBV00] James Bruce, Tucker Balch, and Manuela Veloso. Fast and Inexpensive Color Image Segmentation for Interactive Robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061 – 2066, October 2000.

[BC93] H. N. Bertram and X. Che. General Analysis of Noise in Recorded Transitions in Thin Film Recording Media. *IEEE Transactions on Magnetics*, 29(1):201–208, January 1993.

[BC00] James Bruce and Krishnamurthy Chaudhuri. Automatic Thresholding for Realtime Color Vision: Proposal, 2000.

[Ber68] Elwyn R. Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, New York, 1968.

[Ber94] H. Neal Bertram. *Theory of Magnetic Recording*. Cambridge University Press, 1994.

[BGM85] Nathan R. Belk, Peter K. George, and Greg S. Mowry. Noise in High Performance Thin-film Longitudinal Magnetic Recording Media. *IEEE Transactions on Magnetics*, 21(5):1350–1355, September 1985.

[BGS94] David F. Bacon, Susan L. Graham, and Oliver J. Sharp. Compiler Transformations for High-Performance Computing. *ACM Computing Surveys*, 26(4):345–420, 1994.

[BM96] Barret Brickner and Jaekyun Moon. A High Dimensional Signal Space Implementation of FDTS/DF. *IEEE Transactions on Magnetics*, 32(5):3941–3943, September 1996.

[BM97] Barret Brickner and Jaekyun Moon. High Data Rate Detection for 3D-110 Channels. *IEEE Transactions on Magnetics*, 33(5):2806–2808, September 1997.

[BM98] Barret Brickner and Jaekyun Moon. Low Complexity Signal Space Detection for MTR Coded Channels. *IEEE Transactions on Magnetics*, 34(1):104–109, January 1998.

[BMN83] Richard A. Baugh, Edward S. Murdock, and Bangalore R. Natarajan. Measurement of Noise in Magnetic Media. *IEEE Transactions on Magnetics*, 19(5):1722–1724, September 1983.

[BNKF98] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag, 1998.

[Bre93] G. Larry Bretthorst. An Introduction to Model Selection Using Probability Theory as Logic. In *Maximum Entropy and Bayesian Methods*, 1993.

[BZA+02] Jens Busch, Jens Ziegler, Christian Aue, Andree Ross, Daniel Sawitzki, and Wolfgang Banzhaf. Automatic Generation of Control Programs for Walking Robots Using Genetic Programming. In J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi, editors, *Proceedings of the 5th European Conference on Genetic Programming*, volume 2278 of *Lecture Notes in Computer Science*, pages 258–268. Springer, New York, 2002.

[Cal99] Robert Calan. *The Essence of Neural Networks*. Prentice Hall, 1999.

[CDH03] Arthur Cesarz, Ingo Dahm, and Matthias Hebbel. Reduced Wearout By Automatic Fitness Estimation for the Evolution of Gait Patterns. Technical Report TR 0403, Universität Dortmund, Dortmund, September 2003.

[Com00] Richard Comedford. Magnetic Storage: The medium that wouldn't die. *IEEE Spectrum*, 12:36–39, December 2000.

[CPO00] Elio D. Di Claudio, Raffaele Parisi, and Gianni Orlandi. *Discriminative Learning for Neural Decision Feedback Equalizers*. D-Facto public., April 2000.

[Cru91] Holk Cruse. Coordination of Leg Movement in Walking Animals. In *From Animals to Animats. International Conference on Simulation of Adaptive Behavior*, pages 105–119. MIT Press, 1991.

[CS89] Kenneth L. Clarkson and Peter W. Shor. Applications of Random Sampling in Computational Geometry, II. *Discrete and Computational Geometry*, 4(1):387–421, 1989.

[CST04] Nello Cristianini and John Shawe-Taylor. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.

[Dah04] Ingo Dahm. Neural Networks with On-The-Fly Confidence-Estimation. In *IEEE International Conference on Signal Processing (ICSP)*, September 2004.

[DBB98] Peter Dittrich, Andreas Buergel, and Wolfgang Banzhaf. Learning to Control a Robot with Random Morphology. In P. Husbands and J.-A. Meyer, editors, *Proceedings First European Workshop on Evolutionary Robotics*, pages 165–178, Berlin, 1998. Springer, Berlin, 1998.

[DDHO03] Ingo Dahm, Sebastian Deutsch, Matthias Hebbel, and Andreas Osterhues. Robust Color Classification for Robot Soccer. In *International Robocup Symposium*, July 2003.

[DHN04] Ingo Dahm, Matthias Hebbel, and Walter Nistico. Virtueller Roboter: Automatische Situationsanalyse und Ressourcemanagement in einem Team von Fußballrobotern. Technical Report TR 0304, Universität Dortmund, Dortmund, September 2004.

[DIS99] DISK/TREND. The Rigid Disk Drives Report. *DISK/TREND News*, 1999. http://www.disktrend.com/.

[DJL+02] Uwe Dueffert, Matthias Juengel, Martin Loetzsch, Ronnie Brunn, Martin Kallnik, Nicolai Kuntze, Michael Kunz, Sebastian Petters, Max Risler, Nils Koschmieder, Tim Laue, Thomas Roefer, Kai Spiess, Arthur Cesarz, Ingo Dahm, Matthias Hebbel, and Jens Ziegler. German Team 2002. In *RoboCup 2002*. Lecture Notes in Artificial Intelligence. Springer, 2002.

[DP80] Didier Dubois and Henri Prade. *Fuzzy Sets and Systems: Theory and Application*. Academic Press, 1980.

[DPS01] Ingo Dahm, Jörg Platte, and Guido Stromberg. Hard Disk Simulation Environment, HdSim v2.0, November 2001. Computer Engineering Institute, Universität Dortmund, http://www-ds.e-technik.uni-dortmund.de/~ magrec/.

[DR87] Wilbur B. Davenport and William L. Root. *An Introduction to the Theory of Random Signals and Noise*. IEEE press, New York, 1987.

[DS03] Ingo Dahm and Stefan Schmermbeck. A Fully Adaptive Signal-space Detector with Soft-Information (AS3D). In *IEEE International Conference on Communications (ICC)*, May 2003.

[DSS00] Ingo Dahm, Guido Stromberg, and Uwe Schwiegelshohn. Leistungsarme Signalverarbeitung für Festplattenlesekanäle. In *Mikroelektronik für die Informationstechnik*, pages 269–274. ITG Workshop, VDE Verlag, 2000.

[DZ02a] Ingo Dahm and Jens Ziegler. Adaptive Methods to Improve Self-Localization in Robot Soccer. In *International Robocup Symposium*, July 2002.

[DZ02b] Ingo Dahm and Jens Ziegler. Enwurf und Realisierung einer modularen hierarchischen Kontrollarchitektur für fußballspielende AIBO Roboter. Technical Report TR 0902, Universität Dortmund, Dortmund, April 2002.

[DZ02c] Ingo Dahm and Jens Ziegler. Using Artificial Neural Networks to Construct a Meta-Model for the Evolution of Gait Patterns of Four-Legged Walking Robots. In *International Conference on Climbing And Walking Robots (CLAWAR)*, November 2002.

[DZ03] Ingo Dahm and Jens Ziegler. Entwicklung von verteilten Algorithmen zur effizienten Kontrolle von autonomen Fußballrobotern. Technical Report TR 0303, Universität Dortmund, Dortmund, November 2003.

[EPG94] Javan Erfanian, Subbarayan Pasupathy, and Glenn Gulak. Reduced Complexity Symbol Detectors with Parallel Structures for ISI Channels. *IEEE Transactions on Communication*, 42(4):1661–1671, April 1994.

[FHK+98] K. Fukahori, D. Hutchinson, R. Kuki, K. Saeki, H. Oshikubo, T. Hori, and M. Leung. An Analog EPR4 Viterbi Detector in Read Channel IC for Magnetic Hard Disks. In *IEEE International Solid-State Circuits Conference*, pages 380–381. Silicon Systems Inc., 1998.

[GASRG98] Uta Grothkopf, Heinz Andernach, Sarah Stevens-Rayburn, and Monique Gomez. Library and Information Services in Astronomy III. In H. E. Payne, editor, *ASP Conference Series*, volume 153, 1998.

[GH96] Ed Grochowski and Roger F. Hoyt. Future Trends in Hard Disk Drives. *IEEE Transactions on Magnetics*, 32(3):1850–1854, May 1996.

[Gro03] Ed Grochowski. *Hard Disk Drives - Roadmap*. Hitatchi Global Storage Technologies, 2003. Published at www.hitachigst.com/hdd/hddpdf/tech/hdd-technology2003.pdf.

[GT94] Ed Grochowski and David A. Thompson. Outlook for Maintaining Areal Density Growth in Magnetic Recording. *IEEE Transactions on Magnetics*, 30(6):3797–3800, November 1994.

[Hay94] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company Inc., 1994.

[Hee98] Chris Heegard. Turbo-Coding for Magnetic Recoding. In *IEEE Information Theory Workshop*, pages 18–19, San Diego, CA, 1998.

[HF92] Martin Hassner and Gerhard Fettweis. A Combined Reed-Solomon Encoder and Syndrome Generator with Small Hardware Complexity. In *Proceedings of the 1992 IEEE International Symposium on Circuits and Systems*, pages 1871–1874. IEEE, 1992.

[HHS+98] Martin Hassner, Nyles Heise, Guido Stromberg, Tetsuya Tamura, Barry Trager, and Samuel Winograd. EEPR4 Performance and Symbol ML Detector Implementation for Magnetic Storage Channels. White Paper, RJ 10128, IBM Corporation, August 1998.

[HOP96] Joachim Hagenauer, Elke Offer, and Lutz Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, IT-42:429–445, March 1996.

[HPG02] John L. Hennessy, David A. Patterson, and David Goldberg. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 3rd edition, May 2002.

[HTY+99] Gregory Hornby, Seiichi Takamura, Jun Yokono, Osamu Hanagata, Takashi Yamamoto, and Masahiro Fujita. Autonomous Evolution of Gaits with the Sony Quadruped Robot. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1297–1304, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.

[Hug98] Gordon H. Hughes. Bit errors due to channel modulation of media jitter. *IEEE Transactions on Magnetics*, 43(5):3799–3803, September 1998.

[JCB01] Pieter Jonker, Jurjen Caarls, and Wouter Bokhove. Fast and Accurate Robot Vision for Vision Based Motion. *Lecture Notes in Computer Science*, 2019:149ff., 2001.

[JLB+03] Matthias Juengel, Martin Loetzsch, Ronnie Brunn, Martin Kallnik, Nicolai Kuntze, Michael Kunz, Max Risler, Tim Laue, Thomas Roefer, Ingo Dahm, Matthias Hebbel, Michael Wachter, Andre Osterhues, and Jens Ziegler. German Team 2003. In *RoboCup 2003*. Lecture Notes in Artificial Intelligence. Springer, 2003.

[JM97] Taehyun Jeon and Jaekyun Moon. A Systematic Approach to Signal Space Detection. *IEEE Transactions on Magnetics*, 33(5):2737–2739, September 1997.

[JM98] Taehyun Jeon and Jaekyun Moon. High Speed Implementation of Signal Space Detectors. *IEEE Transactions on Magnetics*, 34(4):1925–1927, July 1998.

[KAK+97] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The robot world cup initiative. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press.

[KM97] Sapthotharan K.Nair and Jaekyun Moon. Data storage channel equalization using neural networks. *Transactions on Neural Networks*, 8(5):1037–1048, September 1997.

[KM98a] Monika Köhle and Dieter Merkl. Experiments in Gait Pattern Classification with Neural Networks of Adaptive Architecture. In *International Conference on Artificial Neural Networks*. Technische Universität Wien, September 1998.

[KM98b] Younggyun Kim and Jaekyun Moon. Low Complexity Signal Space Detector for (1,7)-Coded Partial Response Channels. *IEEE Transactions on Magnetics*, 34(4):1928–1930, July 1998.

[Kor91] Norman L. Koren. Matched Filter Limits and Code Performance in Digital Magnetic Recording. *IEEE Transactions on Magnetics*, 27(6):4594–4599, November 1991.

[Koz92] John R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[KTS+97] Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda, and Minoru Asada. The RoboCup Synthetic Agent Challenge. In *International Joint Conference on Artificial Intelligence (IJCAI97)*, 1997.

[LC83] Shu Lin and Daniel J. Costello. *Error Control Coding*, volume 3. Prentice-Hall, Englewood Cliffs, 1983.

[LFS92] M. Anthony Lewis, Andrew H. Fagg, and Alan Solidum. Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2618–2623, Nice, France, 12-14 1992. IEEE Computer Society Press, Los Alamitos, CA.

[LHC86] Kadaba R. Lakshmikumar, Robert A. Hadaway, and Miles A. Copeland. Characterization and Modeling of Mismatch in MOS Transistors for Precision Analog Design. *IEEE Journal of Solid-State Circuits*, 21(6):1057–1066, 1986.

[MI92] Christopher Michael and Mohammed Ismail. Statistical Modeling of Device Mismatch for Analog MOS Integrated Circuits. *IEEE Journal of Solid-State Circuits*, 27(2):154–166, 1992.

[MJ98] Jaekyun Moon and Taehyun Jeon. Sequence Detection for Binary ISI Channels Using Signal-Space Partitioning. *IEEE Transactions on Communications*, 46(7):891–901, July 1998.

[NAMM93] Ilja N.Bronstein, Konstantin A.Semendjajew, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik*, volume 1.0. Verlag Harri Deutsch, 1993.

[NM94] Sapthotharan K. Nair and Jaekyun Moon. Improved Equalization for Digital Recording Using Nonlinear Filtering and Error Confinement. In *IEEE Transactions on Magnetics*, volume 30, pages 4221–4223, November 1994.

[OBN96] Markus Olmer, Wolfgang Banzhaf, and Peter Nordin. Evolving Real-time Behavior Modules for a real Robot with Genetic Programming. In M. Jamshidi, F. Pin, and P. Dauchez, editors, *Proceedings of the International Symposium on Robotics and Manufacturing (ISRAM-96)*, Robotics and Manufacturing, pages 675 – 680, New York, 1996. Asme Press, New York, 1996.

[Pla99] John Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In Alex Smola, P. Bartlett, Bernhard Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1999.

[Poy96] Charles A. Poynton. *A Technical Introduction to Digital Video*. Prentice Hall, Toronto, 1996.

[PS93] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer Verlag, New York, August 1993.

[Que00] Francis Quek. An Algorithm for the Rapid Computation of Boundaries of Run-length Encoded Regions. *Pattern Recognition Journal*, 33:1637–1649, 2000.

[RDD+04] Thomas Röfer, Ingo Dahm, Uwe Düffert, Jan Hofmann, Matthias Jüngel, Martin Kallnik, Martin Lötzsch, Max Risler, Max Stelzer, and Jens Ziegler. Team Description Paper: GT2003. In *International Robocup Symposium*, 2004.

[Rec94] Ingo Rechenberg. *Evolutionsstrategie '94*. Frommann Holzboog Verlag, 1st edition, September 1994.

[Röf05] Thomas Röfer. *Sony Four Legged Robot Football League Rule Book*. Robocup Federation, 5th edition, Mai 2005.

[RHW86] David Rumelhart, Geoffey Hinton, and Ronald Williams. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing*, volume 1, pages 318–362, Cambridge, MA, 1986. MIT Press.

[Ris02] Max Risler. Inverse Kinematik Walking-Engine: Kurze Beschreibung der Parameter. Technical Note, Technische Universität Darmstadt, Darmstadt, May 2002.

[Ros61] Frank Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington D.C., 1961.

[Sch04] Stefan Schmermbeck. *Soft-Output Signal Space Detection with Application to Magnetic Recording*. PhD thesis, Universität Dortmund, July 2004.

[SCT02] Min C. Shin, Kyong I. Chang, and Leonid V. Tsap. Does Colorspace Transformation Make Any Difference on Skin Detection? In *IEEE Workshop on Applications of Computer Vision*, 2002.

[SD02] Stefan Schmermbeck and Ingo Dahm. Using Reliability Information in Magnetic Recording Systems. In *Kolloquium des Schwerpunktprogramms der Deutschen Forschungsgemeinschaft VIVA*, pages 113–120, March 2002.

[SDO+04] Uwe Schwiegelsohn, Ingo Dahm, Andreas Osterhues, Tim Kaiser, and Carsten Schumann. Progress report: Porting windows ce to the sony ers-210. Technical Report TR 0404, Universität Dortmund, Dortmund, Oktober 2004.

[SHS00] Guido Stromberg, Martin Hassner, and Uwe Schwiegelshohn. Signal Space Detection in Colored Noise. *IEEE Transactions on Magnetics*, 36(3):604–612, May 2000.

[SK94] Wladyslaw Skarbek and Andreas Koschan. Colour Image Segmentation – A Survey. Technical report, Institute for Technical Informatics, Technical University of Berlin, October 1994.

[SK98] Warren Smith and K.K.Thornber. The uniqueness of the maximum and minimum functions for AND and OR in fuzzy logic, Januar 1998.

[SMO01] Baldur Steingrimsson, Jaekyun Moon, and Travis Oenning. Signal Space Detection for DVD Optical Recording. *IEEE Transactions on Magnetics*, 37(2):670–675, March 2001.

[Son03] Sony Corp. *SONY ERS 210/210a. Produkthandbuch*, 2003.

[Spe94] Graham F. Spencer. Automatic Generation of Programs for Crawling and Walking. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 15, pages 335–353. MIT Press, 1994.

[SS02] Bernhard Schölkopf and Alex Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. MIT Press, Cambridge, MA, 2002.

[SSHS03] Stefan Schmermbeck, Guido Stromberg, Martin Hassner, and Uwe Schwiegelshohn. Low-Complexity Signal Processing for ISI Channels. In *IEEE Global Communication Conference*, December 2003.

[Ste02] Baldur Steingrimsson. Soft Signal Space Detection for the Lorentzian magnetic recording channel. *IEEE Transactions on Magnetics*, 38(5):2322–2324, September 2002.

[Str00] Guido Stromberg. *Signal Space Detection with Application to Magnetic Recording*. PhD thesis, Universität Dortmund, September 2000.

[Str02] Alfred Strey. *Computer-Arithmetik*, volume 1. Universität Ulm, Oktober 2002.

[TA00] Jean Christophe Terrillon and Shigeru Akamatsu. Comparative Performance of Different Chrominance Spaces for Color Segmentation and Detection of Human Faces in Complex Scene Images, 2000.

[TB00] David A. Thompson and John S. Best. The Future of Magnetic Data Storage Technology. *IBM Journal on Research and Development*, 44(3):311ff., May 2000.

[TFAS00] Jean Christophe Terrillon, Hideo Fukamachi, Shigeru Akamatsu, and Mahdad N. Shirazi. Comparative Performance of Different Skin Chrominance Models and Chrominance Spaces for the Automatic Detection of Human Faces in Color Images. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, page 54ff., 2000.

[Vap98] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley & Sons Inc, Berlin, 1998.

[VC74] Vladimir Vapnik and Alexei Chervonenkis. *Theory of Pattern Recognition*. Akademie Verlag, Berlin, 1974.

[VR05] Jakob Voss and M.W. Richter. Klassifizierung. *Wikipedia*, Mai 2005.

[VUF+98] Manuela Veloso, William Uther, Masahiro Fujita, Minoru Asada, and Hiroaki Kitano. Playing Soccer with Legged Robots. In *International Conference on Intelligent Robots and Systems IEEE/RSJ*, pages 437–442, October 1998.

[WAD+00] Thilo Weigel, Willi Auerbach, Markus Dietl, Burkhard Dumler, Jens-Steffen Gutmann, Kornel Marko, Klaus Muller, Bernhard Nebel, Boris Szerbakowski, and Maximilian Thiel. CS Freiburg: Doing the Right Thing in a Group. In *RoboCup*, pages 52–63, 2000.

[Wid62] Bernard Widrow. *Self-Organizing Systems*, chapter Generalization and Information Storage in Networks of Adaline Neurons, page 435. Spartan Books, 1962.

[Zad65] Lotfi A. Zadeh. Fuzzy Sets. In *Information and Control*, August 1965.

[ZBBB02] Jens Ziegler, Jens Barnholt, Jens Busch, and Wolfgang Banzhaf. Automatic Evolution of Control Programs for a Small Humanoid Walking Robot. In *International Conference on Climbing and Walking CLAWAR*, 2002.

[Zel94] Andreas Zell. *Simulation Neuronaler Netze*. Addison Wesley, September 1994.

Anhang A

Formelzeichen und Nomenklatur

In dieser Arbeit werden Formelzeichen und Symbole, wie folgt, verwendet:

- Funktionen: kleine griechische Buchstaben (α, β, γ)
- Konstanten: große lateinische Buchstaben (A, B, C)
- Funktionsparameter: große griechische Buchstaben (Φ, Γ, Λ)
- Variablen: kleine lateinische Buchstaben (a, b, c)
- Bestandteile (z.B. von Vektoren): Fußnote (a_0, a_1, a_2)
- Variablen mit unterschiedlicher Belegung: Kopfnote (a^0, a^1, a^2)
- Soll-Wert: überdachtes Symbol (z.B. $\hat{a}, \hat{\alpha}, \hat{\vec{a}}$)

Daneben haben einige Symbolen eine Sonderbelegung: i und j werden generell als Index verwendet, x und y sind Platzhalter für beliebige Variablen, t ist ein Platzhalter für die Zeit, Δ kennzeichnet eine Differenz, die Varianz ist mit σ^2 gekennzeichnet, und η steht stets für eine Zufallsvariable.

Variiert ein Wert x über der Zeit, so wird sein Wert zum Zeitpunkt t als $\langle x \rangle_t$ angegeben. Bei zeitdiskreter Wertänderung und einem Takt Δt gilt $\langle x \rangle_{t+y}$ als Kurzschreibweise für $\langle x \rangle_{t+y \cdot \Delta t}$.

Ansonsten wird so oft wie möglich versucht, eine semantische Zuordnung der Art zu treffen, dass ein Großbuchstabe die Anzahl jener Objekte angibt, die mit dem zugehörigen Kleinbuchstaben gekennzeichnet sind. Indizierungen sind nullbasiert. Daher gilt der Wertebereich $0 \leq i < S$ für alle s^i und entsprechend $0 \leq j < R$ für alle r^j.

A.1 Funktionen und Abbildungen

α	Aktivierungsfunktion des Perzeptrons
β	Boolesche Logik eines Signalraumdetektors
δ	Diskriminanzfunktion
δ	Distanz zur Entscheidungsebene
ε	Abbildung: Punkt auf Klasse
ε_0	Abbildung: Punkt auf Unterraum
ε_1	Abbildung: Unterräume auf Klassen
γ	Funktion zur Bestimmung der Relevanz einer Entscheidungsebene
κ	Ausgabefunktion des Kernel-Perzeptrons
μ	Ausgabefunktion des Multilayer-Perzeptrons
ν	Ausgabefunktion des Adaline-Neurons
ω	Wahrscheinlichkeit
φ	Fehlerfunktion, Hilfsfunktion zur Bestimmung der Fehlerrate
ϕ	Kernel-Funktion des Kernel-Perzeptrons
π	Ausgabefunktion des Perzeptrons
ς	Fitnessfunktion
τ	Transformationsvorschrift zwischen zwei Signalräumen
θ	Nachbarschaftsfunktion einer selbstorganiserenden Karte
ξ	Quantisierungsfunktion
ζ	Zuverlässigkeitsfunktion

A.2 Konstanten

B	Anzahl der Taps des Feedbackfilters
D	Ordnungszahl des Teildetektors bei Multi-Symbol-Detektion
F	Anzahl der Parameter einer Signalraumtransformation
G	Anzahl gleichzeitig klassifizierter Symbole
K	Anzahl Klassen
M	Anzahl Merkmale (Dimension des Merkmalsraumes)
P	Anzahl paralleler Entscheidungsebenen
R	Anzahl Referenzobjekte
S	Anzahl zu klassifizierender Objekte
T	Größe des Bitintervalls bei Festplatten
U	Anzahl Unterräume
W	Anzahl der Taps des Forwardfilters

A.3 Variablen

c^i	Cluster im Merkmalsraum
$\vec{e^i}$	Einheitsvektor des Merkmalsraumes
k^i	Klasse
r^i	Referenzobjekt
s^i	Zu klassifizierendes Objekt
u^i	Unterraum
$\vec{w^i}$	Gewichtsvektor eines Neurons

A.4 Funktionsparameter

Γ_i	Erweiterungskoeffizient zur Komplexitätsreduktion bei MSD
Λ	Lernrate eines Neurons bzw. neuronalen Netzes
Φ_i	Parameter einer Signalraumtransformation
Ψ_i	Funktionsparameter der Aktivierungsfunktion α

Anhang B

Mathematische Notation

$\vec{x}$	Vektor, der aus M Elementen x_i besteht
$\underline{x}$	Matrix, die aus den Elementen $x_{i,j}$ besteht
$\vec{x}^T$	Transponieren des Vektors $\vec{x}$
$(\vec{x}, y)^T$	Vektor, um eine zusätzliche Dimension mit $x_M = y$ erweitert
$\vec{x} \cdot \vec{y}^T$	Skalarprodukt $\sum_i x_i \cdot y_i$
$\|\|\vec{x}\|\|$	Euklidische Norm $\sqrt{\vec{x} \cdot \vec{x}^T}$
$[x, y]$	Menge, die aus allen Elemente x bis y besteht
$\{x, y\}$	Menge, die aus den Elementen x und y besteht
$x \bigvee y$	Boolesche OR-Verknüpfung (x oder y)
$x \bigwedge y$	Boolesche AND-Verknüpfung (x und y)
$\overline{x}$	Boolesche Negation (NICHT x)
$x \oplus y$	Boolesche XOR-Verknüpfung $(x \bigwedge \overline{y}) \bigvee (\overline{x} \bigwedge y)$
$\max[x_i]_{i=0}^{M}$	Maximum aller x_i für $i \in [0, M]$
$\min[x_i]_{i=0}^{M}$	Maximum aller x_i für $i \in [0, M]$
$\omega(x)$	Wahrscheinlichkeit eines Events x
$\omega(x\|y)$	Verbundwahrscheinlichkeit von x bei gleichzeitigem y
$x\%y$	Modulo-Operation. Rest der Devision $x : y$

Anhang C

Abkürzungsverzeichnis

Die nachfolgende Liste enthält die in der Arbeit verwendeten Abkürzungen in alphabetischer Reihenfolge:

ASSD	Adaptiver Signalraumdetektor
AWGN	Average White Gaussian Noise - additive weißes Rauschen
BER	Bit Error Rate - Bitfehlerrate
dB	Dezibel
EA	Evolutionäre Algorithmen
ECC	Fehlerkorrektur
EQL	Equi-Probability-Line - verbindet Punkte gleicher Auftrittswahrscheinlichkeit
ER	Error Rate - Fehlerrate
FIR	Finit Impulse Response - Linearer Filter
ISI	Intersymbolinterferenz
IKWE	Inverse Kinematik Walking-Engine
MLP	Multi-Layer Perzeptronennetzwerk
MSD	Multi-Symbol-Signalraumdetektor
MUX	Multiplexer
OF-ASSD	Signalraumdetektor mit offline Lernverhalten
ON-ASSD	Signalraumdetektor mit online Lernverhalten
PCA	Primary Component Analysis - Hauptkomponentenanalyse
S3D	Signalraumdetektor mit Ausgabe von Zuverlässigkeitsinformation
SNR	Signal Noise Ratio - Signalrauschabstand
SSD	Signal Space Detektor - Signalraumdetektor auch: Signalraumdetektion
SVM	Support-Vektor-Maschine
WSSD	Signal Space Detektor with noise Whitening

Anhang D

Danksagungen

In erster Linie bedanke ich mich bei den Gutachtern dieser Arbeit: Prof. Dr.-Ing. Uwe Schwiegelshohn und Prof. Dr. sc. nat. Hans-Dieter Burkhard. Für die vielen hilfreichen Kommentare bedanke ich mich bei meinen Kollegen Matthias Hebbel, Walter Nistico, Lars Schley, Stefan Schmermbeck und Jens Ziegler. Und schließlich gilt mein Dank den Studierenden, die meine Arbeit mit Fleiß und Ideenreichtum begleiteten.

Der Deutschen Forschungsgemeinschaft (DFG) danke ich für die finanzielle Unterstützung der Projekte im Rahmen der Programme VIVA und SPP 1125. Ich danke der Microsoft Deutschland GmbH, insbesondere Wolfgang Krenz und Alexander Brändle für die Förderung unseres Roboter-Fußballteams „Microsoft Hellhounds“.

Zu guter Letzt gilt mein Dank gilt den außergewöhnlichsten und wichtigsten Menschen auf dieser Erde: meiner Familie. Danke, dass Ihr durch Eure Liebe, Eure immer wieder gespendete Motivation und nicht zuletzt Euren Fleiß bei der Rechtscheibkontrolle dafür gesorgt habt, dass ich diese Arbeit so schreiben konnte, wie sie nun vorliegt.