Entwicklung eines Verfahrens zur statischen Stabilitätsanalyse beim Roboterpalettieren

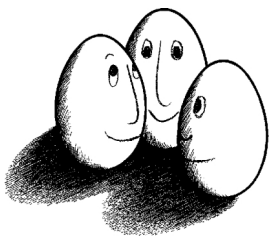
Diplomarbeit vom
Mykhaylo Ruderman

Universität Dortmund, Juni 2005

unter Betreuung von:

Dr. rer. nat. Frank Hoffmann
Lehrstuhl für elektrische
Steuerung und Regelung

Prof. Dr. Katharina Morik
Lehrstuhl für künstliche
Intelligenz



Danksagung

An dieser Stelle möchte ich Herrn Dr. Frank Hoffmann für seine Betreuung, konstruktive Kritik und Einfluss auf mein Interesse zum Wissensgebiet "Pattern Classification" sowie Frau Prof. Dr. Katharina Morik für die Übernahme der Zweitbetreuung danken.

Besonderer Dank gilt allen KUKA SysTec GmbH Mitarbeitern, mit denen ich während eines halben Jahres in einer durchaus freundlichen und kollegialen Atmosphäre arbeiten durfte. Vor allem danke ich Frau Dr. Gerda Ruge für ihre Betreuung vor Ort und die interessante, erfahrugsreiche Zusammenarbeit am Forschungsprojekt, aus dem diese Diplomarbeit entstanden ist.

Weiterhin danke ich Christof Krywka für fachliche Unterstützung in Physik und Matthias Güdemann für zahlreiche Diskussionen und Tipps während der alltäglichen Entwicklungsarbeit.

Meinen Eltern danke ich für ihre Unterstützung während meines Studiums.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

Motivation

Heutzutage versucht man weltweit den Bereich **Waren-Management** und **Logistik** teilweise oder vollständig zu automatisieren. Dieser Tendenz liegen mehrere Ursachen zu Grunde. Zum einen sind das die wirtschaftlichen Aspekte, unter denen die Senkung des Zeitaufwands und der Kosten eine ständig zu wachsende Rolle spielt. Zum anderen werden aus der Sicht der Ergonomie und des Arbeitsschutzes Versuche unternommen, die monotonen, körperlich anstrengenden und gesundheitsschädigenden Tätigkeiten bei Lagerung, Sortierung und Verpackung von Waren mit Hilfe von automatisierten Anlagen zu erledigen.

Eine der anspruchsvollsten Disziplinen im Bereich Lagerlogistik [Bau 04] ist die **Kommissionierung** von Gütern. Nach VDI[1] 3590 wird Kommissionieren wie folgt definiert: "Kommissionieren ist das Zusammenstellen von bestimmten Teilmengen (Artikel) aus einer bereitgestellten Gesamtmenge (Sortiment) aufgrund von Bedarfsinformationen (Auftrag). Dabei findet eine Umformung eines lagerspezifischen in einen verbraucherspezifischen Zustand statt." Jede Kommissionierung setzt sich nach VDI 3590 aus einem Informationsfluss, einem Materialfluss sowie der Organisation zusammen.

Sowohl bei einem manuellen als auch bei einem automatisierten Kommissionieren tauchen Packprobleme verschiedenster Art auf, unter denen das **dreidimensionale Packproblem** eine der wichtigsten Rollen spielt. Allgemein bezeichnet man mit "Packproblemen" solche Probleme, bei denen kleinere Einheiten (Güter) in größere Einheiten (Behälter) gepackt werden sollen. Ein wesentliches Unterteilungskriterium ist die Dimension des Packproblems, d.h. die Dimension der Einheit und Behälter, wobei zwei- und dreidimensionalen Packprobleme von praktischen Relevanz sind [Exeler 88].

Eine typische Anwendung, bei der ein dreidimensionales Packproblem nach einer Lösung verlangt, ist das **gemischte Palettieren** von verpackten Gütern, wobei die Einheiten durch die quaderförmigen **Pakete** mit Waren und Behälter durch die **Holzbrett-Paletten** vertreten sind. Der $3D$-Raum ist durch geometrische Größen der Palette begrenzt, wobei die dritte (i.A. in die vertikale Z-Richtung ausgelegte)

[1] Verein Deutscher Ingenieure

Dimension keiner physikalischen, sondern einer zu vereinbarenden Restriktion unterliegt.

Neben der allgemein weit untersuchten räumlichen Sicht vom dreidimensionalen Packproblem gibt es eine physikalische bzw. **statische** Sicht, der bis jetzt keine ausgearbeiteten Modelle und Verfahren zugrunde liegen.

Aktueller Stand

Das dreidimensionale Packproblem ist aufgrund seiner Komplexität bisher nicht vollständig untersucht worden. Wegen der dritten Dimension müssen für fast alle praktischen Anwendungen außer der Kapazität- auch die **Stabilitätsrestriktionen** berücksichtigt werden, da die Anordnung einer Einheit auf einen Punkt oberhalb der Grundfläche nur möglich ist, wenn diese durch darunterliegende Einheiten abgestützt wird [Exeler 88].

Die algorithmisch orientierte Literatur über Packprobleme hat sich auf Methoden zur Erzielung einer effizienten Auslastung der betreffenden Transport- oder Lagermedien konzentriert. Viele Probleme, die sich in der Praxis stellen, beinhalten jedoch auch andere, über eine Minimierung des unbenutzten Stauraums hinaus gehende, Zielsetzungen. Die Stabilität der Ladung ist ein offensichtliches Beispiel dafür[Bisch 91]. In der Tat nehmen nur sehr wenige Containerbeladungs-Algorithmen in ihrer Logik Bezug auf das Gewicht oder die Tragfähigkeit der zu verladenden Güter [Ratc 97].

Eine Marktstudie der kommerziellen Palettiersoftware sowie die Internetrecherche zeigten, dass es noch **keine Ansätze** zur Lösung des Stabilitätsproblems für das gemischte Palettieren gibt, obwohl dafür ein erheblicher Bedarf besteht und die Stabilität der Ladung in mehreren Anwendungen eine ausschlaggebende Rolle spielt.

Problemstellung

Das Ziel der vorliegenden Diplomarbeit setzt sich aus zwei gut separierbaren Teilzielen zusammen, die in Kooperation miteinander der **statischen Stabilitätsanalyse** beim gemischten Palettieren von verpackten Gütern dienen sollen.

Zum einen soll ein **Statikmodell** für die zu untersuchenden Packeinheiten und Packbehälter, d.h. für die quaderformigen Warenpakete und die daraus entstehenden Paketstapel ausgearbeitet und implementiert werden. Aus dem aufzustellenden Statikmodell sollen die **Stabilitätsfeatures** hervorgehen, die zur eindeutigen Beschreibung der Stabilität eines Paketstapels geeignet werden sollen. Die eigentliche Stabilität der einzelnen Pakete, d.h. ihre Festigkeit gegenüber verschiedener Deformationen, liegt nicht im Problembereich der folgenden Arbeit.

Zum anderen sollen nach der Durchführung einer Reihe von **Palettierexperimenten** und der Gewinnung von empirischen Daten verschiedene Klassifikationsmodelle

mit Hilfe von Methoden der **datenbasierten Mustererkennung** untersucht werden, worauf folgend ein erlerntes (trainiertes) Modell zur Vorhersage der Stabilität eines Paketstapels im Rahmen des Palettiervorgangs vorgestellt werden soll.

Ablauf und Gliederung

Diese Arbeit entstand aus einem Entwicklungsprojekt bei der **Robogistics** Group von der **KUKA SysTec GmbH** und wurde vor Ort durchgeführt unter paralleler Betreuung seitens der SW-Entwicklungsabteilung sowie des Lehrstuhls für elektrische Steuerung und Regelung der Universität Dortmund.

Die Arbeit beinhaltete eine relativ lange und komplexe experimentelle Phase, die allerdings in den zeitlichen Ablauf eingeplant war und zusätzlich zu den empirischen Daten mehrere Erkenntnisse aus der Praxis des gemischten Roboterpalettierens lieferte.

Die im Rahmen der Diplomarbeit vorgesehene SW-Implementierung der **Statikkomponente** sowie der **Klassifikatoren** nahm einen großen Teil des Arbeits- und Zeitaufwandes in Anspruch und wurde nach den gegebenen Spezifikationsvorschriften vollständig durchgeführt. Aufgrund der Geheimhaltungsvereinbarung und des großen Umfangs wurde der Implementierungsteil in der vorliegenden Ausarbeitung weggelassen, wobei die wesentlichen Vorgänge, Strukturen und Algorithmen teilweise oder vollständig enthalten sind.

Die vorliegende Ausarbeitung ist in einer chronologischen und zielbezogenen Reihenfolge gegliedert, wobei die theoretischen Teile zum Statikmodell und zur datenbasierten Mustererkennung in den Kapiteln 1. und 3. beinhaltet sind. Das 2. Kapitel ist der Durchführung der Experimente und Gewinnung der empirischen Daten gewidmet. Das 4. Kapitel beschreibt den Einsatz von Klassifikatoren und die Endergebnisse. Die Ausarbeitung wird mit der Zusammenfassung und einem Ausblick auf weitere Entwicklungen der Stabilitätsanalyse abgeschlossen.

Kapitel 1

Statik des Paketstapels

1.1 Allgemeines

Bei einem Paketstapel handelt es sich offensichtlich um eine bauähnliche Struktur, die allerdings keine Befestigung zwischen einzelnen Elementen enthält. Beim Ausschließen von Schrägstellungen und seitlichen Neigungen der Pakete bleibt ein Paketstapel solange statisch stabil, bis einer der Schwerpunkte der Pakete nicht mehr unterstützt wird.

Bei einem Palletiervorgang können aufeinander gestellte Pakete ihre bisher stabile Lage gegenseitig positiv bzw. negativ beeinflussen, wodurch im ungünstigen Fall der Gesamtstapel einstürzen kann.

Um eine Aussage über die statische Stabilität eines Paketstapels zu ermöglichen, benötigt man ein Statikmodell, mit dessen Hilfe die Konfiguration eines Paketstapels zu jedem Palettierschritt quantitativ beschrieben werden könnte. Basierend auf einem solchen Modell können anschließend diverse Stabilitätsfeatures hergeleitet werden, deren Realisierungen die statische Stabilität eines Paketstapels eindeutig beschreiben würden.

Weitergehend ist es wünschenswert, nicht nur Informationen über die statische Stabilität des Paketstapels zu gewinnen, sondern auch eine Aussage über die statische Stabilität beim Transport zu ermöglichen. Dies vor allem um zu bestimmen, wie schnell ein Paketstapel in alle Richtungen der X, Y-Ebene beschleunigt bzw. gebremst werden darf.

Wegen der unvollständigen bzw. abweichenden Kenntnisse über geometrische und physikalische Eigenschaften einzelner Pakete und deren Lage im Paketstapel, muss zuerst eine Reihe von Annahmen getroffen werden, um die Aufstellung des Statikmodells bis zu einem gewissen Grad zu vereinfachen und dabei seine Richtigkeit nicht zu verletzen:

1. Aufgrund der unbekannten Reibungskoeffizienten der einzelnen Pakete werden

die Reibungskräfte aus dem Modell vollständig ausgeschlossen, wodurch lediglich das Kippen und nicht das Rutschen der Pakete als Folge der instabilen Lage in Betracht gezogen wird.

2. Aufgrund der begrenzten Positioniergenauigkeit der Palettieranlage und der Abweichung von den vermessenen Paketgrößen sind die seitlichen Kontaktflächen nicht bzw. nur teilweise bekannt und können deshalb bei der Aufteilung der Gewichtskräfte innerhalb des Paketstapels nicht mit berücksichtigt werden. Demzufolge wird angenommen, dass die Pakete keine vertikalen unterstützenden Kontaktflächen besitzen, wodurch die Aufteilung der Gewichtskräfte lediglich über die horizontalen Kontaktflächen erfolgt.

3. Bis zu einer Höhentoleranz, die üblicherweise bei Palettieralgorithmen als Parameter eingestellt werden kann, werden die Pakete bzw. Pakettürme als gleich hoch betrachtet, wodurch die geringfügigen Schrägstellungen als Geradestellungen behandelt werden.

4. Beim Ablegen eines Paketes auf den Paketstapel wird kein Spielraum bezüglich der Positionier- und Größengenauigkeit behandelt und die Sollposition eines Paketes wird als seine Istposition betrachtet. In der Tat können die Kontaktflächen von ihren Sollgrößen, die durch Sollgrößen und Sollpositionen der Pakete bestimmt werden, entweder positiv oder negativ abweichen. Während eine positive Abweichung eher für die zusätzliche Stabilisierung sorgt, können die negativen Abweichungen dazu führen, dass mit einer tatsächlich nicht entstehende Stützfläche gerechnet wird. Um dies teilweise vermeiden zu können, werden die Stützflächen, deren Breite einer als Parameter einstellbaren Positioniergenauigkeit unterliegt, nicht als Stützflächen angesehen.

5. Jedes Paket im Stapel wird als ein starrer Körper betrachtet, der keinerlei Deformation ausgesetzt ist und dessen Schwerpunkt in seiner geometrischen Mitte liegt.

Unter den getroffenen Annahmen ist es nun möglich ein Statikmodell des Paketstapels aufzustellen und darauf basierend eine Reihe von Stabilitätsfeatures herzuleiten.

Sowohl der Aufstellung des Modells als auch der Herleitung einzelner Features unterliegt der Gedanke, dass die statische Stabilitätsanalyse in Form einer performanten Software realisierbar sein sollte, um die Arbeit der Palettieralgorithmen sowohl offline als auch online beurteilen bzw. unterstützen zu können.

1.2 Statikmodell

1.2.1 Ruhelage und Kippkraft eines Paketes

Wir betrachten zuerst ein Paket mit Masse m, Schwerpunkt s und seiner Gewichtskraft $\vec{G} = m \cdot \vec{g}$. Unter der getroffenen Annahme, dass ein Paket durch das Kippen und nicht Rutschen aus der Ruhelage gebracht werden kann, kann jede **Kippkante** k des Pakets als eine feste Drehachse angesehen werden, wie es in der Abbildung 1.1 gezeigt wird. Eine Kraft $\vec{F_k}$, die das Paket aus der Ruhelage bringen kann, wird als **Kippkraft** bezeichnet und kann beispielsweise beim Transport des Paketstapels entstehen. Die Entstehung einer solchen Kraft basiert darauf, dass jeder Körper seiner Beschleunigung Widerstand durch seine **träge Masse** entgegen setzt, woraus sich das Newtonsche Kraftgesetz ergibt [Hütte 00]:

$$\vec{F} = m \cdot \vec{a} = m \cdot \frac{d\vec{v}}{dt} \quad , \tag{1.1}$$

wobei $\vec{v}$ für die Geschwindigkeit und $\vec{a}$ für die Beschleunigung des Körpers stehen. Die in der Abbildung 1.1 dargestellte Situation stellt den Fall des zweiseitigen Hebels dar, wobei die Kippkante k als eine Drehachse auftritt und die Kräfte $\vec{F_k}$ und $\vec{G}$ zwei gegeneinander wirkende Drehmomente M_k und M_s erzeugen, wobei M_k als Kipp- und M_s als Standmoment bezeichnet werden.

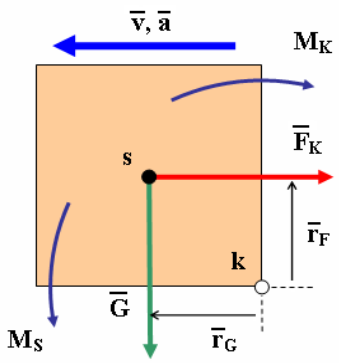


Abbildung 1.1: Instabile Ruhelage des Paketes bezüglich einer Kippkante

An einem zweiseitigen Hebel herrscht das Gleichgewicht, wenn die Summe aller rechtsdrehenden Momente gleich der Summe aller linksdrehenden Momente ist. Dies lässt sich folgendermaßen aufschreiben [Schulze 91]:

$$\sum M = 0 \quad . \tag{1.2}$$

Daraus folgend kann für das Paket aus Abbildung 1.1 aufgeschrieben werden:

$$\vec{F_k} \cdot \vec{r}_F = \vec{G} \cdot \vec{r}_G \quad , \tag{1.3}$$

wobei $\vec{r}_F$ und $\vec{r}_G$ für die entsprechenden Hebelarme im Bezug auf die Kippkante k stehen.

Die in (1.3) stehende Gleichung beschreibt eine sogenannte instabile Ruhelage. Diese bedeutet, dass bereits ein beliebig kleiner Anstieg der Kippkraft $\vec{F}_k$ das Paket aus der Ruhelage bringen würde. Um eine stabile Ruhelage eines Paketes zu beschreiben, muss der linke Teil der Gleichung (1.3) kleiner als rechter sein, woraus folgend eine Ungleichung folgender Form entsteht:

$$\vec{F}_k \cdot \vec{r}_F < \vec{G} \cdot \vec{r}_G \,, \qquad \textit{oder} \qquad |\vec{F}_k| < \frac{\vec{G} \cdot \vec{r}_G}{|\vec{r}_F|} \quad . \tag{1.4}$$

Ist die Bedingung für die Kippkraft $\vec{F}_k$ erfüllt, bleibt die Lage des Paketes stabil. Wird allerdings $|\vec{F}_k| > \vec{G} \cdot \vec{r}_G/|\vec{r}_F|$ kippt das Paket um die entsprechende Kippkante um. Die in dieser Form definierte Bedingung für eine stabile Ruhelage bezüglich einer Kippkraft $\vec{F}_k$ gilt für alle Kippkanten des Paketes, die durch die gesamte **konvexe Stützfläche** des Paketes bestimmt sind.

Es ist offensichtlich, dass je weiter der Schwerpunkt eines Paketes von der jeweiligen Kippkante entfernt liegt, eine umso größere Kippkraft benötigt wird, um das Paket aus der Ruhelage zu bringen.

1.2.2 Ausbreitung der Gewichtskräfte im Paketstapel

Die in einem Paketstapel stehenden Pakete sind nicht von einander unabhängig, sondern beeinflussen ihre Lage gegenseitig, indem sie ihre Gewichtskräfte auf darunter stehende Pakete aufteilen, bzw. die Gewichtskräfte von den oberen Paketen aufnehmen. Während des Palettiervorgangs kann diese gegenseitige Wirkung die Stabilität des Paketstapels erhöhen bzw. verringern und muss deswegen zu jedem Palettierschritt genauer untersucht werden.

Ein Beispiel in Abbildung 1.2 zeigt die Wirkung eines neu aufgelegten Paketes auf den bereits aufgebauten Paketstapel. Dabei zeigen die Pfeile lediglich den Wirkungspfad und nicht die Kraftsvektoren, obwohl die Aufteilung der Gewichtskräfte offensichtlich den gleichen Pfad verfolgt.

Aus Abbildung 1.2 ist nun zu erkennen, dass nicht alle Pakete von der Wirkung eines neu aufgelegten Paketes beeinflusst werden, wie es bei den Paketen **P5**, **P8** und **P11**, die offensichtlich nicht auf dem Wirkungspfad liegen, der Fall ist. Alle anderen Pakete nehmen, wenn auch indirekt, die Teile der Gewichtskraft eines neu aufgelegten Paketes **P14** auf, wodurch die Werte ihrer Kippkräfte verändert werden.

Daraus folgt, dass ein Verfahren notwendig ist, mit dem die Aufteilung der Gewichtskräfte innerhalb eines Paketstapels von oben nach unten verfolgt werden kann.

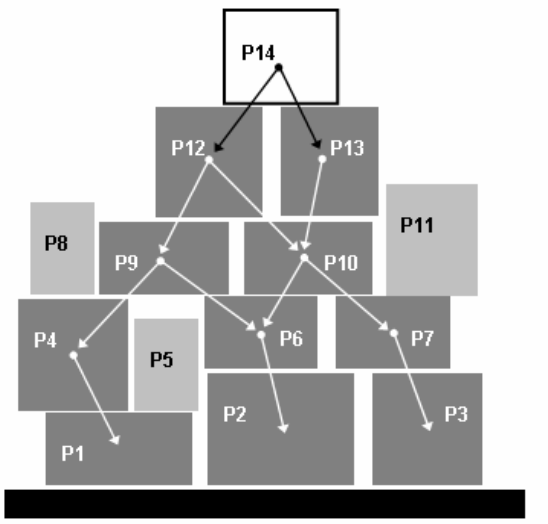


Abbildung 1.2: Wirkung eines neu aufgelegten Paketes auf den Paketstapel

1.2.3 Aufteilung der Gewichtskräfte

Im Folgenden betrachte man die Aufteilung der Gewichtskraft eines Paketes auf die anderen, darunter stehenden Pakete. Um dies zu ermöglichen müssen allerdings einige Annahmen bezüglich der Verteilung der Last getroffen werden:

- Es wird angenommen, dass bei jeder Stützfläche eine gleichmäßig verteilte Flächenlast gemäß [Erlhof 94] vorliegt.
- Ausgehend von erster Annahme wird an der Stelle der Flächenlast eine Punktlast betrachtet, wobei der Angriffspunkt der Punktlast [Erlhof 94] in die Mitte der entsprechenden Stützfläche gesetzt wird.

Unter den getroffenen Annahmen reduziert sich das Problem der Ermittlung der Teilgewichtskräfte und ihre Angriffspunkte, wie es in der Abbildung 1.3 gezeigt wird.

Laut der **ersten notwendigen Bedingung** [Tipler 00] für das statische Gleichgewicht eines Körpers, muss die resultierende äußere Kraft, die auf einen Körper wirkt, gleich 0 sein. Für die Aufteilung einer Gewichtskraft lässt sich daraus folgend aufschreiben:

$$\vec{G} = \sum_{i=1}^{j} \vec{F}_i \quad , \tag{1.5}$$

wobei $\vec{G}$ für Gewichtskraft des oberen Paketes und $\vec{F}_i$ für eine in Gegenrichtung zu $\vec{G}$ gesetzte **Stützkraft** des i-ten unteren Paketes stehen. Die Kraft $\vec{F}_i$ ist im Betrag gleich der Teilgewichtskraft, die vom i-ten Paket aufgenommen wird und greift am selben Mittelpunkt der entsprechenden Stützfläche an.

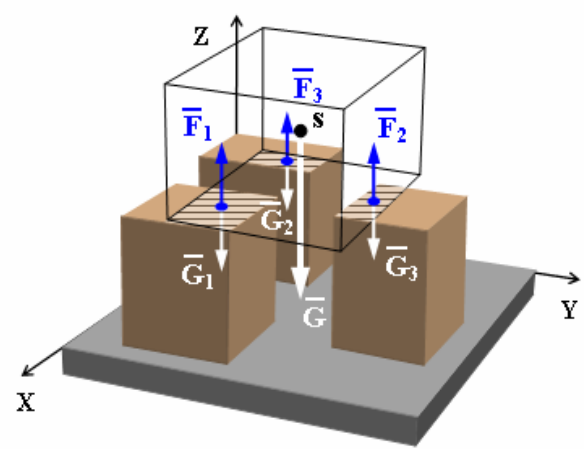


Abbildung 1.3: Aufteilung der Gewichtskraft eines Paketes

Bei der Anzahl der Stützflächen gleich j gibt es genau j Stützkräfte, deren Betrag und Angriffspunkt ermittelt werden müssen. Während die Angriffspunkte direkt aus der Kenntnis der Stützflächen bestimmt werden können, benötigt die Ermittlung der Beträge von den Kräften zuerst eine Fallunterscheidung, die sich auf die Anzahl der Stützflächen bezieht.

Bei der Anzahl der Stützkräfte gleich j werden j linear unabhängige Gleichungen benötigt, um ein Gleichungssystem mit j Unbekannten lösen zu können, wobei die erste Gleichung direkt aus (1.5) übernommen werden kann. Da die Pakete als starre Körper angesehen werden und ihre tatsächliche Deformation nicht mit berücksichtigt wird, können die restlichen $j-1$ Gleichungen lediglich mit Hilfe von Momenten gewonnen werden. Allerdings befinden sich alle Stützflächen in der X,Y-Ebene, wodurch nur zwei unabhängige Momentengleichungen zur Verfügung stehen. Daraus folgt, dass die maximale Anzahl der mit dem Verfahren zu ermittelnden Stützkräfte 3 beträgt.

Basierend auf den Kraft- und Momentgleichungen aus erster und zweiter notwendiger Bedingung für das statische Gleichgewicht eines Körpers [Tipler 00], kann für ein Beispiel mit der Gewichtskraft G und den drei Stützkräften F_1, F_2 und F_3 das folgende Gleichungssystem aufgestellt werden:

$$\begin{pmatrix} 1 & 1 & 1 \\ x_{F1} & x_{F2} & x_{F3} \\ y_{F1} & y_{F2} & y_{F3} \end{pmatrix} \cdot \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} G \\ G \cdot x_G \\ G \cdot y_G \end{pmatrix} \quad , \tag{1.6}$$

wobei x und y für Hebelarme der entsprechenden Kräfte stehen, die durch Koordinaten der Angriffspunkte dieser Kräfte repräsentiert sind.

Für den Fall, dass die Anzahl der Stützflächen größer als 3 ist, muss die Annahme getroffen werden, dass der Betrag der von einem Paket aufgenommenen Teilgewichtskraft proportional zur Stützfläche ist. Eine solche Annahme beruht auf der Vermutung, dass bei mehr als drei Stützflächen der größte Teil der Bodenfläche des Paketes

unterstützt wird. Unter dieser Annahme können nun die einzelnen Teilgewichtskräfte folgendermaßen ermittelt werden:

$$\vec{F}_i = \lambda_i \cdot \vec{G} \qquad mit \qquad \lambda_i = \frac{A_i}{\sum_{i=1}^{j} A_i} \quad . \tag{1.7}$$

Dabei steht λ_i für den Gewichtsaufteilungskoeffizient und A_i für die Stützfläche des i-ten unteren Paketes, wobei die Anzahl der unteren Pakete j beträgt.

1.2.4 Aufnahme der Gewichtskräfte

Als nächstes untersuchen wir die Aufnahme der Gewichtskräfte von einem unterstehenden Paket. Dies ist der Fall, wenn ein Paket ein oder mehrere andere Pakete teilweise oder vollständig unterstützt, wie in Abbildung 1.4 gezeigt wird.

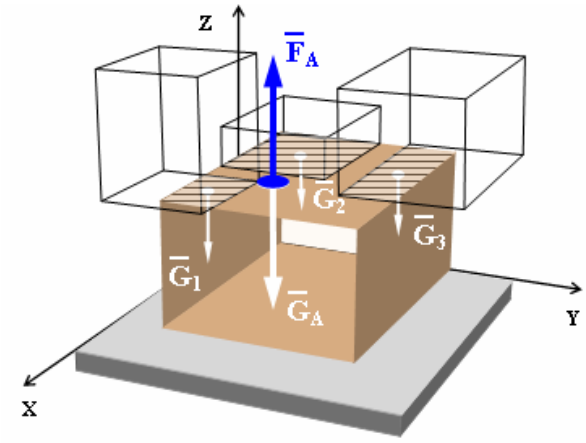


Abbildung 1.4: Aufnahme der Gewichtskräfte von einem Paket

Wenn zwei oder mehrere parallele Kräfte auf einen Körper wirken, dann können sie durch eine einzelne äquivalente Kraft ersetzt werden. Diese Ersatzkraft ist gleich der Summe der Kräfte und greift an einem solchen Punkt an, so dass das von ihr bewirkte Drehmoment gleich dem resultierenden Drehmoment der ursprünglichen Kräfte ist [Tipler 00]. Daraus folgend kann für die Aufnahme der Gewichtskräfte aufgeschrieben werden:

$$\vec{G}_A = \sum_{i=1}^{j} \vec{G}_i \quad , \tag{1.8}$$

wobei $\vec{G}_A$ für die resultierende aufgenommene Gewichtskraft, $\vec{G}_i$ für die aufgeteilte Gewichtskraft des i-ten oberen Paketes und j für die Anzahl der oberen Pakete stehen.

Von Interesse ist nicht nur der Betrag der Gewichtskraft $\vec{G}_A$ sondern auch ihr Angriffspunkt, der mit Hilfe von Momentengleichungen ohne weitere Schwierigkeiten ermittelt werden kann. Um allerdings zu sinnvollen Momentengleichungen zu gelangen, wird an Stelle von $\vec{G}_A$ eine gleich große aber in Gegenrichtung gesetzte Stützkraft $\vec{F}_A$ verwendet, wie in Abbildung 1.4 gezeigt.

Nun können die X, Y-Koordinaten des Angriffspunktes der Gewichtskraft $\vec{G}_A$ folgendermaßen ermittelt werden:

$$X_{GA} = \frac{\sum_{i=1}^{j} G_i \cdot X_{Gi}}{\sum_{i=1}^{j} G_i} \qquad und \qquad Y_{GA} = \frac{\sum_{i=1}^{j} G_i \cdot Y_{Gi}}{\sum_{i=1}^{j} G_i} \quad , \tag{1.9}$$

wobei X_{GA}, Y_{GA} und X_{Gi}, Y_{Gi} die Koordinaten der Angriffspunkte der entsprechenden Kräfte repräsentieren und die Anzahl der oberen Pakete j beträgt.

An diese Stelle ergibt sich eine sinnvolle Anmerkung, die auf die bereits in 1.2.1 definierte Kippkraft $\vec{F}_k$ zurückführt. Gemäß **zweiter notwendiger Bedingung** für das statische Gleichgewicht eines Körpers [Tipler 00], müssen alle an einem Körper wirkende Momente in der Gleichung (1.2) mit berücksichtigt werden. In Bezug auf die resultierende aufgenommene Gewichtskraft $\vec{G}_A$ bedeutet dies, dass die in Ungleichung (1.4) definierte Bedingung für eine Kippkraft $\vec{F}_k$ zu der folgenden Form ergänzt werden muss:

$$|\vec{F}_k| < \frac{\vec{G} \cdot \vec{r}_G + \vec{G}_A \cdot \vec{r}_{GA}}{|\vec{r}_F|} \quad , \tag{1.10}$$

wobei $\vec{r}_{GA}$ für den Hebelarm der aufgenommenen Gewichtskraft $\vec{G}_A$ steht. Dies wird in der Abbildung 1.5 veranschaulicht, wobei zwei prinzipiell unterschiedliche Fälle dargestellt werden.

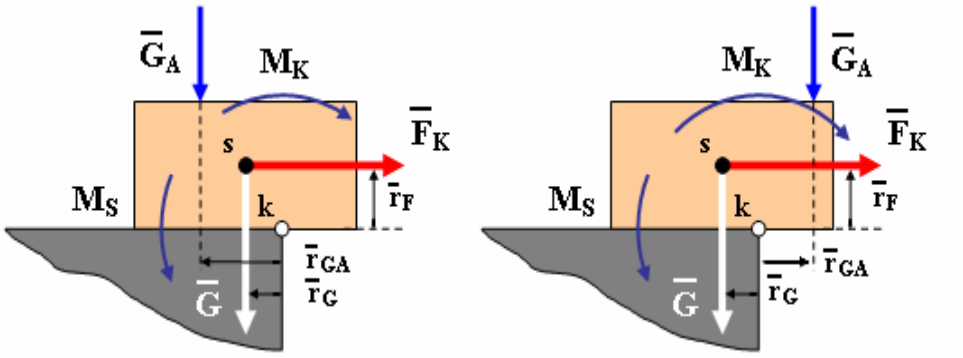


Abbildung 1.5: Wirkung der aufgenommenen Gewichtskraft

Es ist offensichtlich, dass der Hebelarm $\vec{r}_{GA}$, dessen Wert sich aus der Position des Angriffspunktes relativ zur Kippkante ergibt, den Beitrag von $\vec{G}_A$ zur Stabilität des Paketes bestimmt. Liegt der Angriffspunkt der Gewichtskraft $\vec{G}_A$ außerhalb der

Stützfläche des Paketes (siehe Abbildung 1.5 rechts), erhält $\vec{r}_{GA}$ in Ungleichung (1.10) ein negatives Vorzeichen, wodurch der Betrag der Kippkraft $\vec{F}_k$ kleiner wird und das Paket eine weniger stabile Lage erhält.

Die von einem Paket aufgenommene Gewichtskraft $\vec{G}_A$ muss bei der weiteren Aufteilung der Gewichtskräfte auf die darunter stehenden Pakete mit berücksichtigt werden. Nach dem bereits erwähnten Superpositionsprinzip der parallel wirkenden Kräfte können die Gewichtskräfte $\vec{G}_A$ und $\vec{G}$ zu einer resultierenden Gewichtskraft zusammengefasst werden, deren Betrag und Angriffspunkt bei der weiteren Aufteilung verwendet werden müssen. Dabei stellt der Angriffspunkt dieser resultierenden Kraft einen sogenannten **gemeinsamen Schwerpunkt** des Paketes dar.

1.3 Stabilitätsfeatures

Die Aufstellung des Statikmodells hat zum Ziel die Herleitung der Stabilitätsfeatures, deren Realisierung die statische Stabilität eines Paketstapels beschreiben würde.

Einige Features können bei der späteren Evaluierung der experimentellen Daten kausale Abhängigkeiten verschiedenes Grades aufweisen. Die kausalen Abhängigkeiten, die sich in mathematischen Abhängigkeiten äußern, haben stets eine entsprechende Korrelation zur Folge [Litz 01] und verringern dadurch die Kenntnis über die Konfiguration eines Paketstapels in einem n-dimensionalen Zustandsraum, wobei n die Anzahl der betrachteten Stabilitätsfeatures repräsentiert. Aus diesem Grund wird versucht eine Reihe von Stabilitätsfeatures herzuleiten, aus denen nach Gewinnung der empirischen Daten der aussagekräftigste Teilraum der Features ausgewählt werden kann.

1.3.1 Kippsicherheit

Unter allen möglichen Stabilitätsfeatures eines Paketes bzw. des Gesamtstapels benötigt man zuerst ein Maß, welches in direkter Beziehung zu der in 1.2.1 definierten Kippkraft steht. Darüber hinaus sollte dieses Maß eine von der Größe des Paketes möglichst unabhängige Aussage über seine Stabilität liefern. Ein solches Stabilitätsmaß kann durch eine sogenannte **Kippsicherheit**[1] ausgedrückt werden. In der Baustatik wird die Kippsicherheit wie folgt definiert [Erlhof 94]:

$$Kippsicherheit = \frac{Standmoment}{Kippmoment} \quad . \tag{1.11}$$

Allerdings schließt die Definition aus, dass das Kippmoment bereits gleich 0 sein kann, sodass sich der Körper in einer bereits instabilen Lage befindet. Um solche, für die Pakete im Paketstapel möglichen Fälle abzudecken, nimmt man das umgekehrte Verhältnis und betrachten die Kippsicherheit als Kippmoment geteilt durch das Standmoment. Allerdings beinhalten beide Momente die auf die Paketgrößen bezogene Terme, nämlich Hebelarme, die den Begriff der Kippsicherheit für verschiedene Pakete schwer vergleichbar machen. Um dies zu vermeiden und den direkten Bezug auf die Kippkraft nehmen zu können, definiert man die Kippsicherheit als Verhältnis zwischen der Kippkraft und der Gewichtskraft des Paketes. Es gilt dann

$$K = \frac{|\vec{F}_k|}{|\vec{G}|} \quad , \tag{1.12}$$

wobei die Kippsicherheit K angibt, wie groß eine Kippkraft im Vergleich zur Gewichtskraft eines Paketes angelegt werden darf, bis dieses seine Ruhelage verlässt.

[1] Ab Kapitel 2 als *Steadiness* bezeichnet

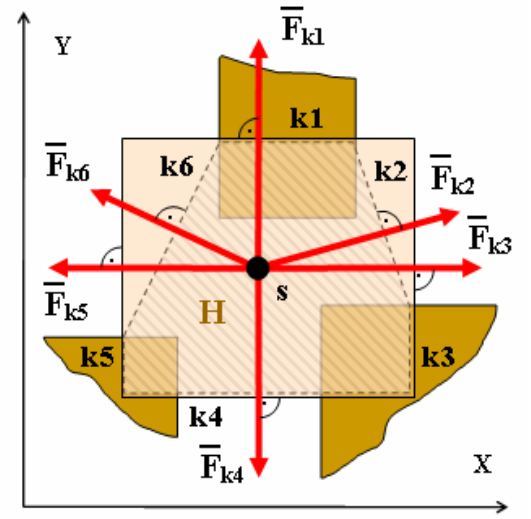


Abbildung 1.6: Kippkräfte im Bezug auf Kippkanten

Eine Kippkraft bezieht sich immer auf eine Kippkante des Paketes und verläuft in X, Y-Ebene senkrecht zu ihr. Die Abbildung 1.6 zeigt ein Beispiel mit der konvexen Stützfläche H, aus der sich sechs Kippkanten $k_1, \ldots, k_6$ ergeben und worauf sich entsprechende Kippkräften $\vec{F}_{k1}, \ldots, \vec{F}_{k6}$ beziehen.

Wie bereits erwähnt, wird der Wert der Kippsicherheit für jede Kippkante einzeln ermittelt. Da allerdings ein Paketstapel bei seinem Transport in allen Richtungen der X, Y-Ebene in Bewegung gesetzt werden kann, müssen alle Kippkräfte unabhängig von der Orientierung der entsprechenden Kippkanten gleich mit berücksichtigt werden. Daraus folgend kann der minimale Wert der Kippsicherheit aller Kippkanten als Stabilitätsmaß für das ganze Paket betrachtet werden. Dies lässt sich folgendermaßen aufschreiben:

$$K_p = \min(\, K_i \mid i = 1, \ldots, j \,) \quad , \tag{1.13}$$

wobei K_p für die Kippsicherheit des Paketes, K_i für die Kippsicherheit in Bezug zur i-ten Kippkante und j für die Anzahl der Kippkanten des Paketes stehen.

Übergehend vom einzelnen Paket zu dem Gesamtstapel kann der kleinste Wert der Kippsicherheit aller Pakete als Aussage für den ganzen Paketstapel betrachtet werden, woraus sich die folgende Definition der Kippsicherheit des Paketstapels ergibt:

$$K_s = \min(\, K_p \mid p = 1, \ldots, n \,) \quad . \tag{1.14}$$

Dabei ist K_s die Kippsicherheit des Paketstapels, K_p steht für die Kippsicherheit des p-ten Paketes und n ist die Anzahl der Pakete im Stapel.

Weist ein Paket im Paketstapel den Wert der Kippsicherheit ≤ 0 auf, bedeutet dies, dass es ohne Anlegen einer Kippkraft umkippen und den Einsturz des gesamten

Paketstapels verursachen würde. Dadurch zeigen sich die Paketstapel mit $K_s \leq 0$ in jedem Fall als instabil.

1.3.2 Sicherheitsabstand

Das nächste Stabilitätsfeature wird mit der Position einer Kippkante relativ zum Schwerpunkt des Paketes verknüpft, wobei alle Kippkanten einzeln betrachtet werden müssen. Der Abstand vom Schwerpunkt zu einer Kippkante in X, Y-Ebene entspricht dem Hebelarm r_G, wie es in der Abbildung 1.5 gezeigt wurde und wird als **Sicherheitsabstand**[2] bezeichnet. Allerdings benötigt dies eine Normierung, die die Werte des Sicherheitsabstands für Pakete unterschiedlicher Größen vergleichbar machen würde.

Als die Normierungsgröße bei den parallel zu Seiten des Paketes verlaufenden Kippkanten eignet sich die Hälfte der Länge bzw. der Breite des Paketes, je nach Orientierung der Kippkante. Für den Fall, dass eine Kippkante nicht parallel zu den Seiten des Paketes verläuft, kann die Hälfte der Diagonale seiner Bodenfläche als Normierungsfaktor genommen werden.

Ein Beispiel für die Normierung des Sicherheitsabstandes bei verschiedenen Verläufen der Kippkanten k_1, k_2 und k_3 wird in Abbildung 1.7 gezeigt, wobei s für den Schwerpunkt, l für die Länge, b für die Breite und d für die Diagonale der Bodenfläche des Paketes stehen.

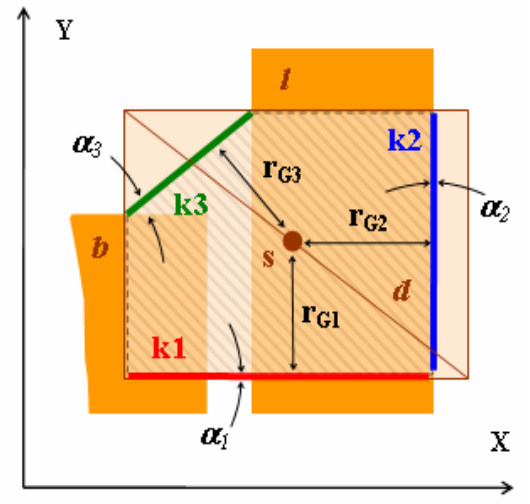


Abbildung 1.7: Normierung des Sicherheitsabstandes

Um die nicht parallel zu den Seiten des Paketes verlaufenden Kippkanten auf eine sinnvolle Weise behandeln zu können, kann eine Fallunterscheidung getroffen werden, sodass der Sicherheitsabstand sich folgendermaßen formulieren lässt:

[2] Ab Kapitel 2 als *Safety Margin* bezeichnet

$$S = \frac{r_G}{q} \tag{1.15}$$

mit

$$q = \begin{cases} l/2, & \text{für} \quad -\pi/6 + \pi n < \alpha < \pi/6 + \pi n, \quad n \in \mathbb{N} \\ b/2, & \text{für} \quad \pi/3 + \pi n < \alpha < 2\,\pi/3 + \pi n, \quad n \in \mathbb{N} \\ d/2, & \text{sonst} \quad , \end{cases}$$

wobei r_G für den Abstand zwischen dem Schwerpunkt und der entsprechenden Kippkante, l für die Länge, b für die Breite und d für die Diagonale der Bodenfläche des Paketes stehen. α repräsentiert den Polarwinkel der jeweiligen Kippkante.

Gleich wie bei der Definition der Kippsicherheit des Paketes bzw. des Paketstapels (siehe 1.3.1) wird der minimale Sicherheitsabstand aller Kippkanten als Stabilitätsmaß für das ganze Paket und der minimale Sicherheitsabstand aller Pakete als Stabilitätsmass für den ganzen Paketstapel betrachtet. Dies lässt sich folgendermaßen aufschreiben:

$$S_p = \min(\, S_i \mid i = 1, \ldots, j \,) \quad \textit{und} \quad S_s = \min(\, S_p \mid p = 1, \ldots, n \,) \quad , \tag{1.16}$$

wobei S_i für den Sicherheitsabstand der i-ten Kippkante, S_p für den Sicherheitsabstand des p-ten Paketes und S_s für den Sicherheitsabstand des ganzen Paketstapels stehen. Die Anzahl der Pakete im Paketstapel beträgt dabei n und die Anzahl der Kippkanten des jeweiligen Paketes j.

1.3.3 Stützbereich

Abgesehen von Betrachtung der Schwerpunkte und Gewichtskräfte, hängt die statische Stabilität eines Paketes im Stapel allgemein davon ab, welcher Anteil seiner Bodenfläche von den anderen Paketen unterstützt wird. Dies gibt an, wie fest ein Paket auf seiner Position steht und kann durch das Verhältnis der Summe aller Stützflächen des Paketes zu seiner Bodenfläche ausgedrückt werden [Hemm 98]. Dieses Stabilitätsfeature wird als **Stützbereich**[3] bezeichnet, deren Definition folgendermaßen aufgeschrieben werden kann:

$$R = \frac{\sum_{i=1}^{j} A_i}{A_{bottom}} \quad , \tag{1.17}$$

wobei A_i für die i-te Stützfläche und A_{bottom} für die Bodenfläche des Paketes stehen. Die Anzahl der Stützflächen beträgt j.

[3] Ab Kapitel 2 als *Support Area* bezeichnet

Mit dem gleichen Prinzip der Minima, mit dem die Kippsicherheit und der Sicherheitsabstand des Paketstapels definiert wurden, betrachtet man den kleinsten Wert des Stützbereiches aller Pakete als Stabilitätsmaß des gesamten Paketstapels. Dies kann folgendermaßen aufgeschrieben werden:

$$R_s = \min(\, R_p \mid p = 1, \ldots, n \,) \quad , \tag{1.18}$$

wobei R_p und R_s für Stützbereiche des p-ten Paketes bzw. des gesamten Paketstapels und n für die Anzahl der Pakete im Paketstapel stehen.

1.3.4 Relative Packungsdichte

Alle bis jetzt eingeführten Stabilitätsfeatures beziehen sich jeweils auf ein einzelnes Paket, wobei anschließend der minimale Wert als Maß für den ganzen Paketstapel gesetzt wird.

Im Folgenden wird auf die Stabilitätsfeatures eingegangen, die sich mit aktueller Konfiguration des Paketstapels verknüpfen und keine Aussage über die Stabilität der einzelnen Pakete liefern. Stabilitätsfeatures dieser Art müssen so definiert werden, dass sie möglichst unabhängig davon sind, wie weit der Paketstapel aufgebaut und wie homogen er gegenüber der Vielfältigkeit seiner Pakete ist.

Als erstes betrachte man die sogenannte relative **Packungsdichte**[4] eines Paketstapels. Diese wird als Verhältnis zwischen dem Gesamtvolumen aller Pakete und dem gesamten in Anspruch genommenen Raum auf der Palette verstanden und lässt sich wie folgt aufschreiben:

$$D = \frac{\sum_{i=1}^{j} V_i}{(\max(X) - \min(X)) \cdot (\max(Y) - \min(Y)) \cdot \max(Z)} \quad , \tag{1.19}$$

wobei V_i für das Volumen des i-ten Paketes und min-, max-Terme für die minimale bzw. maximale X, Y, Z-Koordinate aller Punkte des Paketstapels stehen. Die Anzahl der Pakete beträgt j.

Es ist offensichtlich, dass der maximale Wert der relativen Packungsdichte 1 beträgt und einen "ideal" gepackten Stapel repräsentiert, in dem keine Lücken vorhanden sind. Es lässt sich zusätzlich sagen, dass eine hohe Packungsdichte der Optimierung der logistischen Prozesse beiträgt [Bau 04] und deswegen unabhängig von der statischen Stabilität des Paketstapels erwünscht ist.

[4] Ab Kapitel 2 als *Pack Density* bezeichnet

1.3.5 Überbauquotient

Auf den oberen Lagen eines Paketstapels trägt der Faktor des Überbaus von Paketen zu der Stabilität des Paketstapels zusätzlich bei. Laut [Bisch 91] ist es wünschenswert, dass die Pakete beim Aufbau eines stabilen Paketstapels ähnlich wie bei der Ziegelmauer miteinander überbaut werden. Dies bedeutet, dass eine Position bei der das Paket von mehreren anderen unterstützt wird, der Position mit nur einem darunter stehenden Paket bevorzugt wird, bei gleicher Größe der gesamten Stützfläche (siehe Abbildung 1.8).

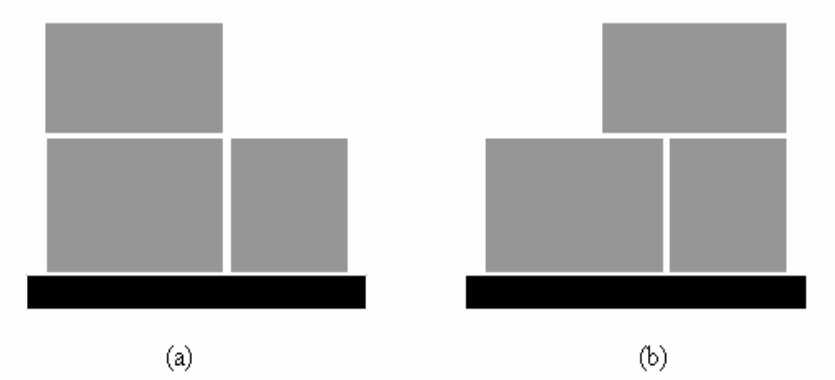


Abbildung 1.8: Paket ohne Überbau (a) und mit Überbau (b)

Daraus folgend wird vermutet, dass ein Paketstapel mit mehreren Überbaustellen sich insgesamt stabiler gegenüber einem Paketstapel ohne solche verhält. Beruhend auf dieser Vermutung wird das nächste Stabilitätsfeature definiert, welches als **Überbauquotient**[5] bezeichnet wird und sich folgendermaßen aufschreiben lässt:

$$O = \frac{\#_{alle} Stützflächen - \#_1 Stützflächen}{\#_{alle} Pakete - \#_1 Pakete} \quad , \tag{1.20}$$

wobei $\#_{alle} Stützflächen$ und $\#_{alle} Pakete$ für die gesamte Anzahl der Stützflächen und Pakete im Paketstapel und $\#_1 Stützflächen$, $\#_1 Pakete$ für die Anzahl der Stützflächen bzw. Pakete in der ersten Lage (direkt auf der Palette) stehen. Es ist offensichtlich, dass für die in der ersten Lage stehenden Pakete kein Überbau in Frage kommt, sodass die Stützflächen und Pakete nur ab der zweiten Lage mit berücksichtigt werden.

1.3.6 Höhe des Gemeinsamen Schwerpunktes

Obwohl die Pakete in dem vorliegenden Anwendungsbereich bei ihrer Positionierung keiner Restriktion bezüglich des Gewichtes oder Inhaltes unterliegen, ist es allgemein erwünscht, die schweren Pakete in die unteren und die leichten in die oberen Lagen zu bringen. Wie weit ein Paketstapel diese Eigenschaft erfüllt, kann mit Hilfe des gemeinsamen Schwerpunktes des Paketstapels beurteilt werden. Seine Z Koordinate

[5] Ab Kapitel 2 als *Overbuild Ratio* bezeichnet

wird auf die Höhe des Paketstapels normiert und als **Höhe des Gemeinsamen Schwerpunktes**[6] bezeichnet.

Nach dem Superpositionsprinzip für parallel wirkende Kräfte, welches in 1.2.4 bereits erwähnt wurde, können die Gewichtskräfte aller Pakete durch eine resultierende Gewichtskraft ersetzt werden, deren Angriffspunkt den gemeinsamen Schwerpunkt des Paketstapels repräsentiert. Die X, Y, Z-Koordinaten des Gemeinsamen Schwerpunktes können mit Hilfe von drei linear unabhängigen Momentengleichungen ermittelt werden. Nun kann die Höhe des Gemeinsamen Schwerpunktes wie folgt definiert werden:

$$H = \frac{s_{com}.Z}{h_{st}} \quad , \tag{1.21}$$

wobei $s_{com}.Z$ für die Z Koordinate des Gemeinsamen Schwerpunktes s_{com}, und h_{st} für die Höhe des Paketstapels stehen. Mit der Höhe des Paketstapels ist die Z Koordinate des obersten Punktes aller Pakete zu verstehen.

Je mehr schwere Pakete in die unteren Lagen des Paketstapels gesetzt werden, desto kleiner wird der Wert von H, unabhängig davon wie hoch der Paketstapel gebaut wurde.

[6] Ab Kapitel 2 als *Height of Common Barycenter* bezeichnet

1.4 Datenstrukturen und Algorithmen

In dem vorherigen Unterkapitel wurde das Statikmodell des Paketstapels aufgestellt und eine Reihe von Stabilitätsfeatures zu seiner Beschreibung hergeleitet. Um die statische Stabilitätsanalyse rechnergestützt durchführen zu können, benötigt man nun diverse Datenstrukturen und Algorithmen, worauf basierend eine Software-Statikkomponente entwickelt werden könnte.

Im folgenden Unterkapitel werden die relevantesten Datenstrukturen und Algorithmen der Statikkomponente vorgestellt, wobei auf die ausführliche Beschreibung aller Aspekte und Details der Implementierung verzichtet wird.

1.4.1 Geometriemodell

Geometriemodelle spezifizieren die rechnerinterne Beschreibung der modellierten geometrischen Objekte. Sie unterscheiden sich durch mathematische Beschreibung der erzeugten Elemente und im damit festgelegten Informationsgehalt [Freund 03]. Außerdem benötigen sie unterschiedliche Datenstrukturen zur Speicherung auf dem Rechner.

In Anbetracht, dass alle physikalische Objekte des Paketstapels die Form eines Quaders haben, kann klassifiziert nach dem Operationsraum ein sogenanntes 2-1/2D-Modell verwendet werden. Dies stellt eine vereinfachte 3D-Geometrie, beschrieben durch eine Ebene und einen senkrechten Verschiebungsvektor, dar.

Für eine effiziente Arbeit mit den einzelnen Punkten sowie mit den Kräften, eignet sich am besten eine vektorielle Darstellung, mit der alle wesentliche Punkte, wie etwa Schwerpunkte, Ecken oder Angriffspunkte der Kräfte in Form der Koordinatenvektoren (X, Y, Z) dargestellt werden können. So können alle Pakete nach einer Vereinbarung bezüglich ihrer Einordnung und Orientierung durch die X, Y, Z-Position, Größe (Länge, Breite, Höhe) und Gewicht dargestellt werden können.

Die Eigenflächen der Pakete sowie einzelne Kontaktflächen, deren Kanten parallel zu X, Y-Achsen verlaufen, können ebenso durch ihre X, Y, Z-Position und Größe (Länge, Breite) dargestellt werden. Allerdings eignet sich diese Art der Darstellung für die gesamte Stützfläche eines Paketes grundsätzlich nicht, da diese die in X, Y-Ebene beliebig orientierten Kanten besitzen kann und keiner Verallgemeinerung unterliegt. Deswegen wird die gesamte Stützfläche eines Paketes durch eine Liste der Punkte repräsentiert, deren Durchlauf die Ecken der Stützfläche im Gegenuhrzeigersinn wiedergibt.

Der Ursprung des verwendeten Koordinatensystems wird an einer Ecke der Palette gebunden, sodass die Länge der Palette mit X- und die Breite mit Y-Achsen zusammenfallen und mit der nach oben zeigenden Z-Achse ein rechts-orientiertes Koordinatensystem bilden, wie es in der Abbildung 1.9 gezeigt wird. Dabei stehen s für den Schwerpunkt, l für die Länge, b für die Breite und h für die Höhe des Paketes.

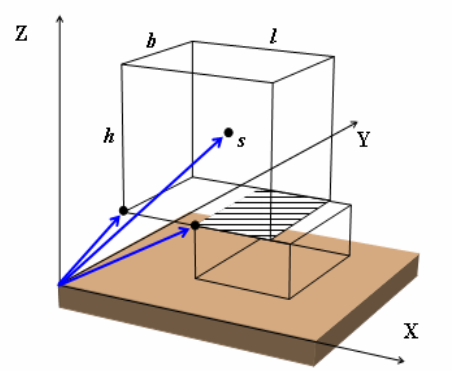


Abbildung 1.9: Koordinatensystem und Positionsvektor des Paketes und der Fläche

Die Position eines Paketes bzw. einer Fläche wird durch die Ecke mit minimalen X, Y, Z-Koordinaten repräsentiert, wobei die Länge immer mit der X- und die Breite immer mit Y-Richtung verstanden werden, um die Orientierung der Pakete nicht extra unterscheiden zu müssen.

1.4.2 Datenstrukturen

In der Informatik unterscheidet man üblicherweise zwischen Verfahren zur Lösung von Problemen und ihrer Implementation in einer bestimmten Programmiersprache auf bestimmten Rechnern. Man nennt diese Verfahren Algorithmen. Die meisten Algorithmen erfordern jeweils geeignete Methoden zur Strukturierung der von den Algorithmen manipulierten Daten. Algorithmen und Datenstrukturen gehören also zusammen. Die richtige Wahl von Algorithmen und Datenstrukturen ist ein wichtiger Schritt zur Lösung eines Problems mit Hilfe von Rechnern [Ott 96].

Die in einem Modell verwendbaren Daten können in Eingabe-, Ausgabe- und Interndaten aufgeteilt werden, wobei aus der Sicht der Programmlogik die Interndaten vom größten Interesse sind. Sie bestimmen nämlich strukturelle und funktionelle Zusammenhänge, die den Algorithmen ermöglichen das gestellte Problem mit einem zulässigen Speicher- und Zeitaufwand effizient zu lösen.

Die Implementierung der Statikkomponente erfolgte mit objektorientierter Sprache **C#**, wobei jede andere objektorientierte Programmiersprache, die dynamische Datenstrukturen, wie z.B. *Array List*, unterstützt, geeignet wäre. Nun werden die relevantesten Klassen der Statikkomponente aufgelistet ohne ihren Dateninhalt und Methoden ausführlich zu beschreiben.

Klasse Point dient der Darstellung von $3D$-Koordinaten und ermöglicht alle übliche Operationen auf $3D$-Vektoren. Zusätzlich enthält Klasse *Point* Methoden zur Transformation des Koordinatensystems, mit deren Hilfe die Koordinaten eines Punktes in verschobenen und rotierten Koordinatensystemen wiedergeben werden können. In Folge der Verwendung eines 2-1/$2D$-Geometriemodells, ist

lediglich die Rotation um die Z-Achse von Bedeutung, wodurch sich die gesamte **homogene Transformation** auf die Verschiebung und die anschließende Rotation des Koordinatensystems um eine Achse reduziert.

Klasse Edge dient grundsätzlich der Darstellung von Kippkanten eines Paketes, welche um seine konvexe Stützfläche liegen. Die Klasse *Edge* implementiert Methoden, die den Abstand zwischen einem Punkt und der entsprechenden Kippkante berechnen und prüfen, ob der entsprechende Punkt innerhalb der konvexen Stützfläche liegt.

Klasse ContactArea repräsentiert horizontale Kontaktfläche und enthält dynamische Referenzen auf die Pakete, welche diese Kontaktfläche bilden. Außerdem erfolgt durch eine Kontaktfläche die "Übergabe" der aufgeteilten bzw. aufgenommenen Gewichtskräfte, sodass sie eine Art Verbindungsknoten zwischen den Paketen innerhalb des Paketstapels darstellt.

Klasse Packet stellt das physikalische Paketobjekt dar und enthält die Daten, die sowohl Eigenschaften des Warenartikels als auch die Lage des Paketes im Paketstapel beschreiben. Durch dynamische Referenzen auf seine oberen und unteren Kontaktflächen wird ein Paketobjekt mit den anderen im Paketstapel verbunden. Die Klasse *Packet* implementiert Methoden zur Berechnung der paketbezogenen Stabilitätsfeatures, wobei die Berechnung der Kippsicherheit und des Sicherheitsabstandes (siehe 1.3.1 bzw. 1.3.2) nur nach der Aktualisierung der aufgenommenen Gewichtskräften erfolgen kann.

Klasse PacketStack ist die Hauptklasse der Statikkomponente, die den Paketstapel mit allen strukturellen und funktionalen Zusammenhängen zwischen den einzelnen Elementen repräsentiert. Sie beinhaltet Methoden, mit denen die Pakete nach Prinzip eines Stacks auf den Stapel auflegt (*push*) bzw. von dem Stapel entfernt (*pop*) und die aktuellen Werte der Stabilitätsfeatures wiedergeben werden können. Bei Abfrage der Stabilitätsfeatures erfolgt eine neue Berechnung aller Gewichtskräfte innerhalb des Paketstapels, sodass die Aktualität der Stabilitätsfeatures garantiert werden kann.

Klasse Utility ist eine Hilfsklasse, die keine Instantiierung vorsieht und lediglich die Methoden der geometrischen und physikalischen Berechnungen enthält. Zu solchen Methoden zählen die Bestimmung der konvexe Stützhülle, Lösung des linearen Gleichungssystems sowie die Such- und Sortiermethoden für die Pakete und Flächen nach den bestimmten Eigenschaften. Zusätzlich enthält die Klasse *Utility* alle verwendbaren Konstanten der Statikkomponente, wie etwa Gravitationskonstante und Koordinaten- und Winkelgenauigkeitet der Berechnung.

Die Statikkomponente wurde in Form einer dynamischen Klassenbibliothek (.dll) implementiert. Sie ist mit den Parametern **Positioniergenauigkeit** sowie **Höhentoleranz** zu instanziieren und stellt anschließend nach jedem neu aufgelegten Paket die aktuellen Werte der erforderlichen Stabilitätsfeatures zur Verfügung.

Das Format der Eingabedaten der Statikomponente wurde an die Spezifikation der Software von der Palettieranlage angepasst, wodurch die Werte der Stabilitätsfeatures sowohl beim offline als auch online Palettieren ermittelt werden können.

1.4.3 Bestimmung der konvexen Stützflächen

Falls ein Baukörper von mehreren anderen unterstützt wird, gilt als die Stützfläche bei der Bestimmung der Standsicherheit diejenige Fläche, die von den Umhüllenden der Grundfläche gebildet wird [Erlhof 94]. Daraus folgt, dass für jedes auf die andere Pakete gestellte Paket die konvexe Hülle um die einzelnen Stützflächen bestimmt werden muss, um seine gesamte Stützfläche zu gewinnen.

Eines der effizientesten Verfahren zur Bestimmung der konvexen Hülle einer endlichen Menge von Punkten ist **Graham-Scan** [Lang 02][7] Algorithmus. Seine Idee ist, aus der Punktmenge zunächst ein sternförmiges Polygon zu konstruieren und dieses anschließend in ein konvexes Polygon umzuformen. Dies wird nun etwas genauer untersucht.

Aus der Punktmenge (siehe Abbildung 1.10(a)) wird zunächst ein Punkt q mit minimaler Y-Koordinate gewählt. Ausgehend von q werden anschließend die Polarwinkel zu allen anderen Punkten bestimmt, wie es in der Abbildung 1.10(b) gezeigt wird. Die Punkte werden nach diesem Winkel sortiert und zu einem Polygonzug verbunden. Falls mehrere Punkte den gleichen Winkel besitzen, werden sie zusätzlich aufsteigend nach ihrem Abstand zu q sortiert.

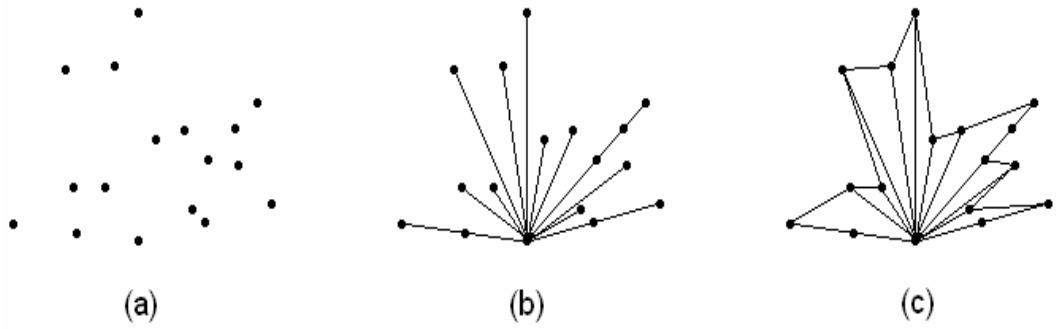


Abbildung 1.10: Aus der Punktmenge konstruierter sternförmiger Polygonzug

Anschließend entsteht ein sternförmiger Polygonzug, wie es in der Abbildung 1.10(c) zu sehen ist. Der sternförmige Polygonzug wird nun ausgehend von q durchlaufen, wobei die konkaven Ecken überbrückt werden, wie ist in der Abbildung 1.11 zu sehen ist. Als Ergebnis steht ein konvexes Polygon, welches die konvexe Hülle der Punktmenge darstellt.

Der Graham-Scan Algorithmus lässt sich mit folgender Implementierung (siehe Algorithmus 1.1) realisieren: Als Eingabe wird ein Array p mit m Punkten verwendet.

[7] Querverweis auf [Graham 72]

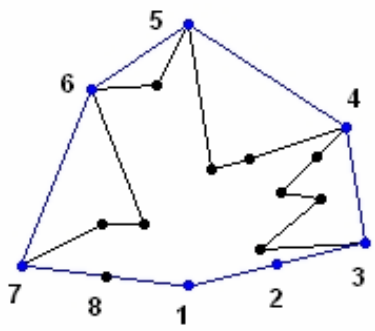


Abbildung 1.11: Erzeugtes konvexes Polygon

Als Ausgabe steht Array p derart umgeordnet, dass die ersten n Einträge die Ecken des konvexen Hüllpolygons sind.

Algorithm 1.1: Bestimmung der konvexen Hülle (Graham-Scan)

Eingabe: Array p mit m Punkten
Ausgabe: Array p mit n ersten Punkten des konvexen Hüllpolygons

```
1  begin
2      bestimme Punkt q mit minimaler Y Koordinate ;
3      subtrahiere q von allen Punkten des Arrays ;
4      sortiere die Punkte des Arrays nach ihrem Winkel und bei gleichem
       Winkel nach ihrem Abstand zum Nullpunkt (der Punkt q wird zu p_0);
       /* durchlaufe die Punkte und überbrücke konkave Ecken       */
5      setze i = 3 und k = 3;
6      while k < m do
7          vertausche p_i und p_k;
8          while p_{i-2} p_{i-1} p_i nicht konvex do
9              vertausche p_{i-1} und p_i;
10             setzte i = i − 1;
11         end
12         setzte i = i + 1 und k = k + 1;
13     end
14     setzte n = i;
15 end
```

Das sortieren der Punkte kann z.B. mit Hilfe vom Quicksort-Algorithmus [Lang 02], genauso wie mit jedem anderen effizienten Sortierverfahren erfolgen, wobei die Punkte gleichzeitig nach ihrem Polarwinkel und ihrem Abstand zum Punkt p_0 miteinander verglichen werden.

Der einzige unerhebliche Nachteil des Algorithmus besteht darin, dass das berechnete konvexe Hüllpolygon auch $\pi/2$-Ecken (siehe Abbildung 1.11 Punkt **2**) enthält und zwar alle, bis auf die entgegen dem Umlaufsinn zu p_0 benachbarten (Punkt **8**). Mit einem Durchlauf des Hüllpolygonzuges können aber diese leicht entfernt werden.

1.4.4 Rekursive Aufteilung der Gewichtskräfte

Bei der Aufteilung der Gewichtskräfte der einzelnen Pakete innerhalb des Paketstapels kann die Bestimmung der Kippkräfte des jeweiligen Paketes nur dann erfolgen, wenn alle von den oberen Paketen aufgenommene Gewichtskräfte ermittelt sind, wie bereits in 1.2.2 beschrieben. Dies führt zur Notwendigkeit, alle Gewichtskräfte innerhalb des Paketstapels rekursiv zu berechnen, wobei der erste Rekursionsaufruf bei allen oberen Paketen ausgeführt werden soll. Dabei sind mit oberen Paketen diejenige Pakete gemeint, deren Referenzen auf die oberen Kontaktflächen leer sind. Der Zusammenhang der Pakete im Stapel unterscheidet sich allerdings von einer Baumstruktur dadurch, dass manche Kinderknoten die Referenzen zu mehreren Elternknoten besitzen, wodurch die Datenstruktur sich dem Fall eines gerichteten Graphes ähnelt. So benötigt die rekursive Berechnung eine "Synchronisationserweiterung", bei der alle Kontaktflächen eine zusätzliche Markierung erhalten, durch die eine "Freigabe" des Knotens für den Rekursionsschritt erteilt wird. Die Aufteilung der Gewichtskräfte kann nun mit folgender Implementierung realisiert werden:

Algorithm 1.2: Rekursive Aufteilung der Gewichtskräfte

```
   /* Rekursiver Aufruf für alle obere Pakete                    */
 1 begin
 2 |  bestimme Menge P von oberen Paketen im Paketstapel;
 3 |  foreach paket in P do
 4 |  |  Rekursion(paket);
 5 |  end
 6 end
   /* Aufteilung der Gewichtskräfte angefangen von paket         */
 7 Rekursion(paket)  begin
 8 |  bestimme gesamte aufgenommene Gewichtskraft;
 9 |  if alle obere Kontaktflächen von paket freigegeben then
10 |  |  bestimme gemeinsamen Schwerpunkt;
11 |  |  if Liste unterer Kontaktflächen von paket leer then
12 |  |  |  return;
13 |  |  else
14 |  |  |  teile Gewichtskraft auf U darunter stehende Pakete;
15 |  |  |  foreach bottom_paket in U do
16 |  |  |  |  gebe Kontakfläche frei;
17 |  |  |  |  Rekursion(bottom_paket);
18 |  |  |  end
19 |  |  end
20 |  else
21 |  |  return;
22 |  end
23 end
```

1.4.5 Lösung der linearen Gleichungssysteme

Die Bestimmung der aufgeteilten Gewichtskräfte, die bereits in 1.2.3 erläutert wurde, erfolgt für die Anzahl der Stützflächen ≤ 3 durch die Lösung eines linearen Gleichungssystems, wobei die einzelnen Gleichungen linear unabhängig sein sollten.

Ein lineares Gleichungssystem mit n Gleichungen und n Unbekannten $x_1, \ldots, x_n$, kurz LGS hat die Form

$$\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & = & b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & = & b_2 \\ \vdots & & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n & = & b_n \end{array} \tag{1.22}$$

mit den Koeffizienten a_{ij} und den Absolutgliedern b_i (kommt eine Unbekannte in einer Gleichung nicht vor, so hat sie dort den Koeffizienten 0). Für ein solches LGS schreibt man

$$\begin{pmatrix} a_{12} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} \tag{1.23}$$

oder kurz

$$A \cdot \vec{x} = \vec{b}$$

mit der Koeffizientenmatrix $A = (a_{ij})_{n \times n}$, dem Spaltenvektor $x \in \mathbb{R}^n$ mit den unbekannten Komponenten x_i und dem Spaltenvektor $\vec{b} \in \mathbb{R}^n$ der rechten Seite [Verf 04].

Über die Berechnung der Inversen einer regulären Matrix A^{-1} kann ein LGS nach der Formel

$$\vec{x} = A^{-1} \cdot \vec{b} \tag{1.24}$$

gelöst werden, wobei dieses Verfahren bei den großen A Matrizen wesentlich mehr Operationen als das **Gaußsches Eliminationsverfahren** benötigt. Die Inverse einer regulären Matrix A^{-1} kann anhand der Adjunkten $Ad\,A$ und der Determinante $det(A)$ wie folgt bestimmt werden:

$$A^{-1} = \frac{1}{det(A)} \left(Ad\,A\right)^T \quad . \tag{1.25}$$

Der Rechenaufwand bei der Bestimmung der Inversen einer regulären Matrix A^{-1} ergibt sich zum größten Teil aus der Ermittlung der Determinanten, wobei die einzelnen **Determinanten** für kleine Matrizen mit Hilfe der Laplace-Formel effizient

berechnet werden können. Das Verfahren wird auch Laplace'scher Entwicklungssatz genannt und lässt sich wie folgt aufschreiben:

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} \cdot a_{ij} \cdot \det(A_{ij}) \quad , \tag{1.26}$$

wobei A_{ij} die $(n-1) \times (n-1)$-Untermatrix ist, die aus A durch Streichung der i-ten Zeile und j-ten Spalte entsteht.

Das Verfahren ist vor allem für eine rekursive Implementierung gut geeignet und arbeitet effizient mit den Matrizen, deren Größenordnung nicht die zulässige Tiefe der Rekursionsaufrufen übersteigt.

Kapitel 2

Palettierexperimente

Um die vorhandenen Palettieralgorithmen auf Qualität der ausgeführten Palettieraufträge zu testen und um empirische Daten für die Entwicklung eines Klassifikationssytems zu gewinnen, wurde eine Reihe von Palettierexperimenten durchgeführt, bei denen mehrere verschiedene Experimentziele gleichzeitig verfolgt wurden.

Die getesteten Palettieraufträge wurden von **vier** verschiedenen Palettieralgorithmen (Palettiersoftware) bearbeitet, wobei als Input für alle vier die gleichen Aufträge verwendet wurden.

Drei von den eingesetzten Paletieralgorithmen sind **offline**, d.h. sie verfügen zum Anfang der Ausführung über den ganzen Palettierauftrag, und können dadurch alle möglichen Varainten bei der Suche nach bester Lösung ausprobieren.

Das vierte eingesetzte Palettieralgorithmus ist **online**. Er verfügt zu jedem Palettierschritt lediglich über einen Teil des Palettierauftrags, der sich zur Laufzeit in einem begrenzten Puffer befindet. So hat Palettieralgorithmus nach einer optimalen Zwischenlösung zu suchen ohne den Rest des Auftrages mit berücksichtigen zu können. Die Größe des Puffers wurde während der Palettierexperimente auf 8 eingestellt und entsprach damit der tatsächlichen mechanischen Einrichtung der verwendeten Palettieranlage.

Durch den Einsatz verschiedener Palettieralgorithmen wurden die Paketstapel mit ganz verschiedenen Konfigurationen erzeugt, wodurch die für die statische Stabilitätsanalyse gewünschte Inhomogenität der Paketstapel erhöht werden konnte.

Der Palettiervorgang aller Aufträge sollte vollständig protokolliert werden, um anschließend alle Zwischenschritte der Entstehung eines Paketstapels und den Ereignistyp wiedergeben zu können. Aus der Sicht der vorliegenden Arbeit waren zwei Typen von Ereignissen von Bedeutung, nämlich die stabil stehenden Paketstapel (im Weiteren *Stable*[1] Paketstapel) und die während des Palettiervorgangs eingestürzten Paketstapel (im Weiteren *Crash*[2] Paketstapel). Das aufgrund der Fehler seitens Pa-

[1] Die entsprechende Klasse wird *Stable* genannt

[2] Die entsprechende Klasse wird *Crash* genannt

lettieralgorithmen vorgekommene Kollisionsereignis war allerdings uninteressant, da es keinerlei Aussage über die statische Stabilität eines Paketstapels lieferte.

Die Palettierexperimente wurden mit einer zur Verfügung gestellten Roboter-Palettieranlage durchgeführt, wobei die Protokollierung der Palettiervorgänge von der Materialfluss-Software übernommen wurde. Dabei wurde pro Paketstapel eine Logdatei in Form eines spezifizierten **XML**-Formats ablegt.

2.1 Roboter-Palettieranlage

Als Roboter-Palettieranlage wurde eine **PFB**[3] Roboterzelle verwendet (siehe Abbildung 2.1), welche eine Förderstrecke für die Paketenzufuhr und einen 4-achsigen Portalroboter für das Palettieren beihaltet.

Abbildung 2.1: Roboter-Palettieranlage

Der gesamte Materialfluss ist mit einem **SPS**-Kontroller und der Portalroboter mit einem **KRC**[4]-Kontroller gesteuert, wobei alle Rechnereinheiten, einschließend Palettiersoftware-Rechner zu einem lokalen Netzwerk zusammen geschlossen sind.

[3] Pick From Belt

[4] KUKA Robot Control

Der Portalroboter verfügt über drei Translations- und eine Rotationsachse, wodurch die Pakete an beliebige Positionen, mit jeder Z-Orientierung innerhalb des Arbeitsraums abgesetzt werden können.

Der Roboterflansch ist mit einem Vakuum-Sauggreifer ausgerüstet (siehe Abbildung 2.2), der die Pakete von bestimmten Pufferplätzen der Förderstrecke abheben und auf die von der Palettiersoftware berechneten Positionen absetzen kann. Die Positioniergenauigkeit setzt sich dabei aus mehreren einzelnen Toleranzen zusammen, wie etwa: Genauigkeit der mechanischen Einrichtungen einzelner Pufferplätze, Orientierungabweichungen der Pakete bei ihrer Zufuhr, Genauigkeit des Paketgriffs bzw. der Paketablage. Die tatsächliche während der Palettierexperimente beobachtete Positioniergenauigkeit betraff etwa $5-15$ mm und wurde anschließend bei der Initialisierung der Statikkomponente als Positioniergenauigkeit-Parameter mit berücksichtigt.

Abbildung 2.2: Sauggreifer zum Roboterpalettieren

Aus organisatorischen und technischen Gründen wurden alle Paketstapel auf einem Simulationsrechner im Voraus generiert und anschließend in Form von Inputdaten der Steuerungssoftware der Roboter-Palettieranlage übergeben. Auf diese Weise erfolgte ein Palettiervorgang unmittelbar nach der entsprechenden Initialisierung der Steuerungssoftware automatisch, bis der Paketstapel komplett aufgebaut wurde. Bei einem Einsturz des Paketstapels konnte der Palettiervorgang von einem Beobachter abgebrochen werden, wobei die entsprechenden Einträge zum *Crash*-Ereignis in die Logdatei geschrieben wurden.

Nach der Durchführung aller Palettierexperimente stand für jeden aufgebauten Paketstapel eine Logdatei zur Verfügung, in der sowohl die Informationen zu jedem Palettierschritt als auch die Klassenzugehörigkeit (*Stable*, *Crash*) eingetragen wurden.

2.2 Paketsatz

Die Palettierexperimente wurden mit zwei unterschiedlichen Typen von Paketen, nämlich mit den realen Warenpaketen und den Schaumstoff-Paketen durchgeführt, wobei die Anzahl der aufgebauten Paketstapel für reale und Schaumstoff-Pakete im Verhältnis 60% zu 40% lag.

Die verwendeten Pakete mit den realen Waren (siehe Abbildung 2.3) hatten das Gewicht zwischen $2,2$ und 12 kg und ihre Länge (als größtes geometrisches Mass) variierte zwischen $0,26$ und $0,6$ m. Die Pakete wiesen eine unterschiedliche Festigkeit, unabhängig von ihrem Gewicht auf und wurden während der dauerhaften Palettierexperimente unterschiedlich stark deformiert. Insgesamt waren 20 verschiedene Sorten von realen Paketen in die Sortenmenge der Packaufträge eingenommen, obwohl sie je nach Entscheidung der Palettieralgorithmen mit verschiedener Häufigkeit in den generierten Paketstapeln erschienen.

Abbildung 2.3: Reale Pakete

Die verwendeten Schaumstoff-Pakete (siehe Abbildung 2.4) hatten eine beinah ideale Quaderform, besaßen eine homogene Gewichtsverteilung und unterlagen keiner Deformation während der Palettierexperimente. Aufgrund des relativ kleinen Gewichtes zeigten sie sich allerdings beim Ablegen als weniger stabil als reale Pakete und veränderten leicht ihre Position durch die Gleiteffekte und die mechanischen Stöße mit den Nachbarpaketen. Insgesamt wurden 100 Sorten der Schaumstoff-Pakete in das Sortiment eingenommen, und ihre Gewichte und Längen variierten zwischen $0,1$ und $2,7$ kg und zwischen $0,21$ und $0,6$ m entsprechend.

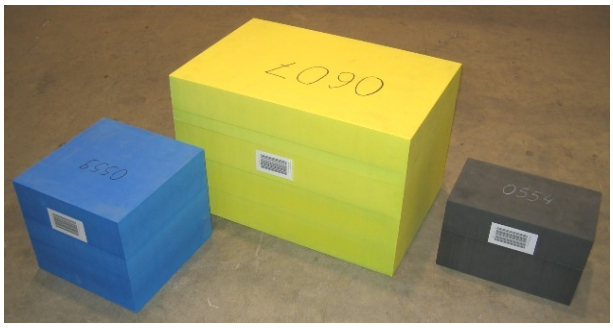


Abbildung 2.4: Schaumstoff Pakete

2.3 Experimentelle Beobachtungen

Während der Palettierexperimente wurden mehrere verschiedene Aspekte der Entstehung von den Paketstapeln beobachtet, die einerseits die Einschränkungen und die Mängel des Statikmodells herausstellten, anderseits die Komplexität und die Vielseitigkeit des zu untersuchenden Problems bestätigten. Viele von ihnen bezogen sich allerdings auf die eigentliche Qualität der Palettieralgorithmen und waren für die statische Stabilitätsanalyse eher irrelevant. Die anderen dagegen beeinflussten direkt das Erscheinen einer der beiden Ereignisklassen und müssen deswegen etwas genauer untersucht werden.

Die während der Palettierexperimente eingestürzten Paketstapel bestätigten die Annahme 1. aus 1.1 und zeigten, dass die Pakete eher durch das Kippen und nicht durch das Rutschen aus der Ruhelage gebracht werden, da die Reibungskräfte beim Zusammenhalten eines Paketstapels eine nicht zu unterschätzende Rolle spielen.

Das notwendige Ausschließen von seitlichen Kontakten (Annahme 2. in 1.1) führte allerdings dazu, dass die an sich stabilen Lagen mancher Pakete, die sich seitlich an die Nachbarpakete lehnten (siehe Abbildung 2.5) als instabil eingeschätzt wurden, falls die Schwerpunkte der Pakete außerhalb der entsprechenden Stützflächen lagen. Dies führte dazu, dass mehrere Paketstapel, in denen einige Schwerpunkte gering oder überhaupt nicht unterstützt wurden, und höchst wahrscheinlich einstürzen sollten, als *Stable*-Paketstapel erschienen. Außerdem wirken auf die Pakete mit seitlichen Kontaktflächen die Seitenkräfte, welche nicht mit berücksichtigt werden, dennoch in die tatsächliche Kräftenbilanz einfließen.

Abbildung 2.5: Seitliche Kontakte der Pakete im Paketstapel

Bezüglich der 3. Annahme in 1.1 wurde festgestellt, dass beim Überbau von Paketen mit verschiedener Höhe die Aufteilung der Last je nach Höhendifferenz unterschiedlich stark vom Modell abwich. Demzufolge erfolgt die rechnerische Aufteilung der Gewichtskräfte innerhalb des Paketstapels mit einem gewissen Fehler, wodurch die Korrektheit der Stabilitätsfeatures mehr oder weniger beeinflusst wird. Im *worst case* stützt sich ein Paket auf eine seiner unteren Stützflächen überhaupt nicht, wobei seine Gewichtskraft lediglich auf die anderen darunter stehende Pakete aufgeteilt wird, wie es am Beispiel in der Abbildung 2.6 zu sehen ist. Wurden allerdings die weiteren Pakete auf ein solches Paket aufgestellt, verursachte es häufig eine geringfügige Schrägstellung, wobei die Aufteilung der Gewichtskräfte wieder beinah modellgemäß erfolgt.

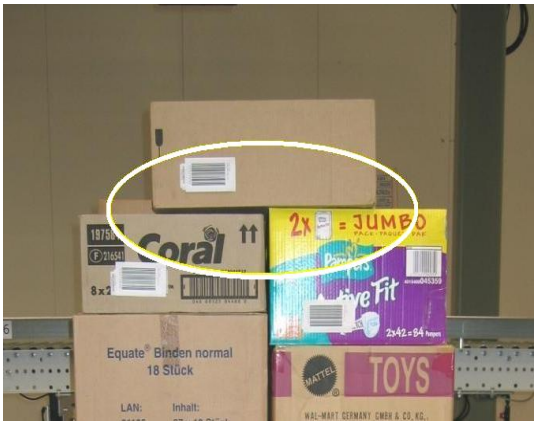


Abbildung 2.6: Überbau bei der Höhendifferenz

In vielen Fällen wurde das Problem der Höhendifferenzen dadurch ausgeglichen, dass die realen Pakete einer geringen elastischen Deformation ausgesetzt wurden, wodurch die Pakete in unteren Lagen mit dem wachsenden Paketstapel ihre Höhen aneinander anpassten.

Wie bereits in 2.2 erwähnt, wurden beim Palettieren die Positionierungenauigkeiten von 5 bis zu 15 *mm* beobachtet, wodurch die modellbezogene Aufteilung der Gewichtskräfte auf die Soll-Kontaktflächen im Paketstapel mit einem gewissen Fehler erfolgt. Anderseits waren die tatsächlichen Kontaktflächen, mit Ausnahme weniger Paketstapel, breit genug, damit alle rechnerisch ermittelten Kontaktflächen den tatsächlichen Kontaktflächen beinah entsprachen (Annahme 4. in 1.1).

Bezüglich der Annahme 5. in 1.1 wurde festgestellt, dass mehrere Pakete unterschiedlich stark einer Deformation ausgesetzt wurden, wie es in der Abbildung 2.7 gezeigt wird. Während dies bei der bereits erwähnten Anpassung der Pakethöhen einen eher positiven Einfluss auf die Stabilität eines Paketstapels hatte, verursachte häufig eine Deformation der Pakete in unteren Lagen eine geneigte Stellung der Pakettürme und als Folge das größere Einsturzrisiko des Paketstapels.

Die nächste beobachtete Eigenschaft der Entstehung eines Paketstapels hängt eng damit zusammen, in welcher Reihenfolge die Pakete auf ihre Positionen abgesetzt wurden. So kam es in manchen Fällen dazu, dass die statische Stabilität eines Paketes

Abbildung 2.7: Deformation eines Paketes

bzw. des gesamten Paketstapels von der Reihenfolge der abgesetzten Pakete abhängig war. Dies wird mit dem Beispiel aus der Abbildung 2.8 erläutert.

Abbildung 2.8: Reihenfolge vom Aufbau eines Paketstapels

In der dargestellten Konfiguration ist die Lage des Paketes **1** offensichtlich stabil, hängt allerdings davon ab, ob das Paket **2b** oder **2a** zuerst darauf gestellt wird. Ist es der Fall vom Paket **2b**, so wird die stabile Lage des Paketes **1** verstärkt und das Aufstellen des Paketes **2a** kann problemlos erfolgen. Die umgekehrte Reihenfolge kann allerdings dazu führen, dass das Paket **1** nach dem Aufstellen des Paketes **2a** seine stabile Lage verlässt und zum Einsturz des Paketstapels führt. Aus der Sicht der statischen Stabilitätsanalyse bedeutet dies, dass ein an sich stabiler Paketstapel in den Zwischenschritten seiner Aufstellung als instabil erscheinen kann.

2.4 Gewinnung der Stabilitätsfeatures

Wie bereits erwähnt, stehen nach der Durchführung aller Palettierexperimente die empirischen Daten zur Verfügung, wobei die Information für jeden aufgebauten Paketstapel in Form einer XML Logdatei gespeichert ist. Anhand der implementierten Statikkomponente und der zusätzlichen Schnittstellen können die Stabilitätsfeatures für jeden Paketstapel ausgewertet und für die weitere **datenbasierte Mustererkennung**[5] aufbereitet werden.

Insgesamt stehen 419 Logdateien zur Verfügung, aus denen zuerst alle Paketstapel mit dem Kollisionsereignis (*Collision*) ausgefiltert werden müssen, da sie keinerlei Information angesichts der statischen Stabilität liefern. Nach dem Ausschließen der erwähnten *Collision*-Paketstapel enthält der Satz von Daten 397 Objekte (ein Featuresvektor pro Paketstapel), die für statische Stabilitätsanalyse geeignet sind.

Die vorläufige Analyse der gewonnenen empirischen Daten erfolgte mit Hilfe von **PRTools4**©[Duin 04], welches als *Toolbox* innerhalb der **MATLAB**©-Umgebung die zahlreichen Funktionen und Unterprogramme für die datenbasierte Mustererkennung zur Verfügung stellt.

Die Dichtefunktion vom Stabilitätsfeature Kippsicherheit (im Weiteren *Steadiness*) wird in der Abbildung 2.9 dargestellt. Angesichts ihres Verlaufs wird eine weitere Aufbereitung (Filterung) der **Trainingsdaten** vorgenommen.

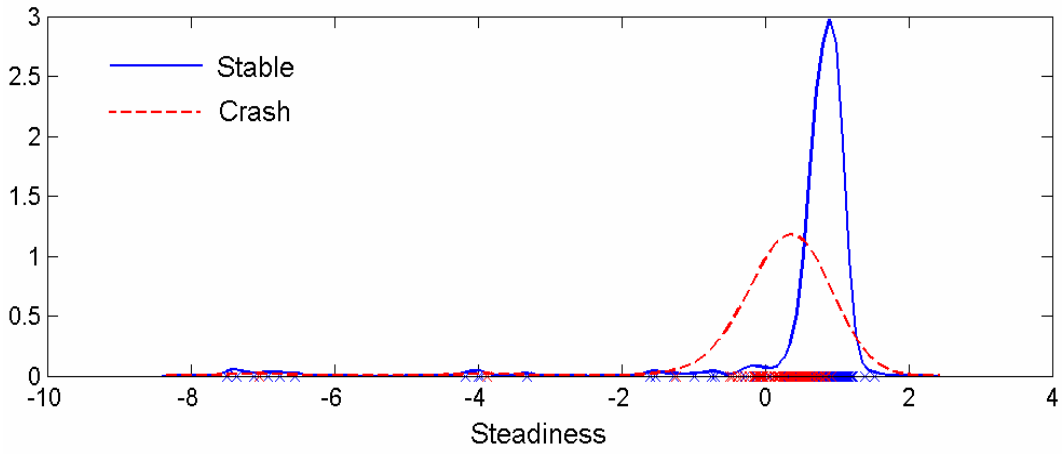


Abbildung 2.9: Dichtefunktion von *Steadiness* vor dem Filtern

Aus der Abbildung 2.9 sieht man, dass einige Werte von *Steadiness* weit im negativen Bereich liegen und stark zerstreut sind. Für *Stable*-Paketstapel entspricht dies meistens dem Fall, in dem die Paketstapel dank seitlicher Kontakte zwischen einigen Paketen stabil blieben, obwohl manche Schwerpunkte außerhalb der entsprechenden Stützflächen liegen. Dies sollte einen sicheren Einsturz des Paketstapels verursachen,

[5] Auch als *pattern classification* oder *pattern recognition* bezeichnet

wenn die Pakete nicht von Nachbarpaketen seitlich unterstützt wären (siehe Unterkapitel 1.3.1).

Damit diese Einzelfälle, die vom aufgestellten Statikmodell am stärksten abweichen, die Verteilung der Stabilitätsfeatures beider Klassen nicht zu stark beeinflussen, werden die Paketstapel beider Klassen bzw. ihre Datensätze mit negativen Werten von *Steadiness* aus den Trainingsdaten (nicht aber aus den Evaluierungsdaten) ausgefiltert. Die in Abbildung 2.9 dargestellte Dichtefunktion von *Steadiness* sieht nun folgendermaßen aus::

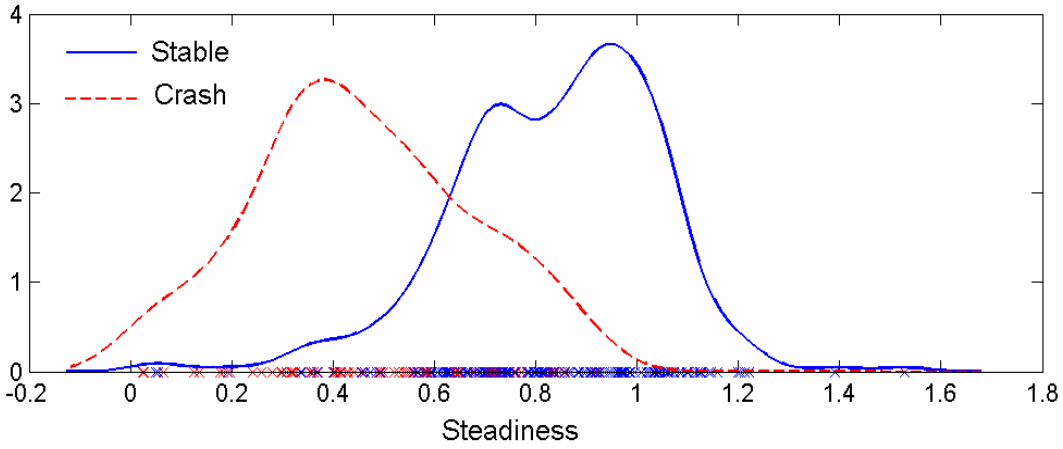


Abbildung 2.10: Dichtefunktion von *Steadiness* nach dem Filtern

Nun stehen der Stabilitätsanalyse ein Satz mit 360 Daten (Featuresvektoren) für das **Training** und ein mit 397 Daten für die **Evaluierung** zur Verfügung. Die Untersuchung und Auswahl der best geeigneten Stabilitätsfeatures werden weiter im Unterkapitel 4.1 beschrieben.

Kapitel 3

Datenbasierte Mustererkennung

Dieses Kapitel widmet sich dem Thema **datenbasierte Mustererkennung**, wobei nur ein kleiner Ausschnitt dieses breiten Wissensgebiets angegangen wird.

Zuerst werden einige Grundlagen und Vorgehensweisen der datenbasierten Klassifikation aufgezeigt. Anschließend wird die **Bayes'sche**[1] **Klassifikation**, die sich als best geeignetes Klassifikationsmodell für das vorliegende Problem herausstellte, etwas genauer untersucht. Zum Vergleich mit Bayes'scher Klassifikation werden einige Grundzüge von *Support Vector Machines* (**SVM**) sowie von künstlichen neuronalen Netzen (**KNN**) erläutert.

3.1 Grundlagen

3.1.1 Begriff und verwandte Gebiete

Datenbasierte Mustererkennung ist ein Bereich vom **"Maschinenlernen"**, wobei dieser Begriff von mehreren Forschungsgebieten auf unterschiedliche Art definiert wird. Er ist umfassend und beinhaltet etwa künstliche Intelligenz, Statistik, Komplexitätstheorie, künstliche Neuronale Netze, Adaptive Kontrolle und weitere Wissensgebiete. Wie der Name bereits verrät, befasst sich das Maschinenlernen mit **"Machines that learn to perform a task from experience"** [Fuku 90], und hat das Training und den Einsatz von Systemen mit Intelligenz verschiedenes Grades zum Ziel.

Eine Task kann dabei meist durch eine mathematische Funktion ausgedrückt werden

$$\mathbf{y} = f(\mathbf{x}; \mathbf{w}) \quad ,$$

[1] Thomas Bayes (ca. 1702-1761) - Gründer der Wahrscheinlichkeitstheorie als erweiterte Logik

wobei x für Eingabe-, y für Ausgabe-Symbole (Vektoren) und w für die Modellparameter, die gelernt werden müssen, stehen. Dabei unterscheidet man zwischen:

- **Regression**: Die Ausgabe y besitzt einen kontinuierlichen Charakter. Als Beispiel kann Steuerung eines autonom navigierenden Fahrzeuges genannt werden.
- **Klassifikation**: Die Ausgabe y ist diskret und dient meist der Bestimmung einer Klassenzugehörigkeit. Zu den zahlreichen Anwendungsgebieten zählen Sprach- und Schrifterkennung, Bildverarbeitung, Material- und Warenhandhabung.

Das Kernproblem der Klassifikation ist es, die Klassenzugehörigkeit eines Musters anhand eines korrespondierenden Merkmalsvektors $\mathbf{x}$ vorherzusagen. Für c gegebene Klassen $\{w_1, \ldots, w_c\}$ wird im allgemeinen Fall eine Entscheidungsfunktion (*decision function*) $\alpha : \mathbb{R}^p \rightarrow \{1, \ldots, c\}$ gesucht, sodass

$$\alpha(\mathbf{x}) = j \ , \quad wobei \quad x \in \mathbb{R}^p \quad . \tag{3.1}$$

Die Funktion α bildet die Menge aller Merkmalsausprägungen auf einen diskreten, endlichen Wertebereich ab. Während die Entscheidungsfunktion "harte" Klassengrenzen zieht und somit entlang der Entscheidungsgrenzen unstetig ist, liegt ihr meist eine stetige **Diskrimantenfunktion** g zugrunde, die im Weiteren genauer angegangen wird.

Die zu lernenden Parameter w charakterisieren die Familie der Funktionen bzw. das Modell, indizieren den Raum der Hypothesen und können in Form von Vektoren, Verbindungsmatrizen, Graphen oder anderen Strukturen vertreten werden.

Neben der Unterscheidung zwischen den **statistischen** und **verteilungsfreien** Verfahren werden sie häufig in **parametrische** und **nicht-parametrische** Klassifikationsverfahren unterteilt, wobei einige linearen (nicht-parametrischen) Klassifikationsverfahren auch zu den parametrischen gezählt werden, weil bei ihrem Training eine Anzahl der Parameter bestimmt bzw. geschätzt wird.

- Parametrische Klassifikationsverfahren bedienen sich der in funktionaler Form gegebenen Modelle, deren Parameter beim Training des Systems geschätzt (gelernt) werden müssen. Diese Modelle sind meistens schnell und einfach zu trainieren und auszuwerten, besitzen allerdings einen großen Nachteil: sie setzen die Kenntnis der Verteilungsfunktion der Merkmale voraus. Diese kann in der Realität häufig schwer oder gar nicht geschätzt werden. Außerdem reicht in einigen Fällen die Anzahl der Trainingsdaten für eine genügende Parameterschätzung nicht aus.

  Trotz dieser Nachteile werden die parametrischen Klassifikationsverfahren in verschiedenen Bereichen mit Erfolg eingesetzt, vor allem dort, wo die berechneten physikalischen oder gemessenen Größen einer bekannten Verteilung unterliegen. Typische Vertreter dieser Verfahren sind **Bayes'sche Klassifikation**, **Support Vector Machines**.

- Zu den nicht-parametrischen Klassifikationsverfahren gehören zum einen die Verfahren, bei denen die Dichtefunktion der Merkmale ohne Annahme einer bestimmten Form der Verteilung direkt aus den Daten geschätzt wird. Zum anderen sind es die Verfahren, die sich gar keiner Dichtefunktion der Merkmale bedienen.

  Die nicht-parametrischen Verfahren zeigen sich bei mehreren Problemstellungen äußerst effizient, verursachen allerdings einen viel größeren Zeit- und Speicheraufwand als die parametrischen Verfahren und lassen sich allgemein schwerer entwerfen und implementieren. Zu diesen Verfahren gehören die **Histogramme**, **Parzen-Fenster**, **K-Nearest Neighbor**, sowie **Fuzzy-Klassifikation** und **künstliche neuronale Netze** verschiedenster Art.

3.1.2 Mustererkennungssysteme

Allgemein kann ein datenbasiertes Mustererkennungssystem in mehrere Komponenten aufgeteilt werden, die sich durch ihre Funktionsbereiche voneinander abgrenzen lassen, und je nach Problemstellung verschiedene Komplexität besitzen.

Viele Mustererkennungssysteme können gemäß der Abbildung 3.1 strukturiert werden, wobei die einzelnen Komponenten erweitert bzw. reduziert werden können.

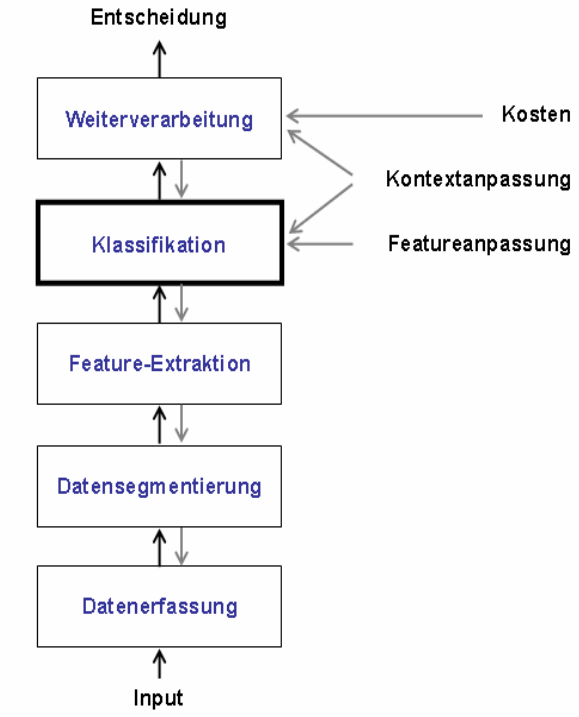


Abbildung 3.1: Struktur eines Mustererkennungssystems

Datenerfassung: Als Input ist häufig ein Messwertwandler, wie etwa Videokamera oder Mikrophon gegeben [Duda 01]. Es kann aber genauso gut ein beliebi-

ger, von einem anderen System gelieferter Datenstrom sein, der entweder diskret oder kontinuierlich in das Mustererkennungssystem einfließt. Die Qualität der Inputdaten hängt häufig von den Aspekten wie Bandbreite, Auflösung, Verzögerung, Weißrauschen ab, die bei der Datenerfassung mit berücksichtigt werden müssen.

Datensegmentierung: Die von der Datenerfassung gelieferten Daten können Informationen von mehreren Objekten genau wie vom "Hintergrund" enthalten. Die Aufgabe der Datensegmentierung ist es nun, die zusammengehörenden Daten in Segmente (Kategorien) aufzuteilen, wobei bereits segmentierte Daten auch als **labeled** genannt werden.

Feature-Extraktion: Das herkömmliche Ziel von Feature-Extraktion ist das Objekt so zu charakterisieren, dass es bei der Messung seiner Features erkannt wird, deren Werte ähnlich wie bei Objekten gleicher Kategorie (Klasse) und verschieden zu den von anderen Kategorien (Klassen) sind [Duda 01]. Um dies zu erreichen, werden einzelne Eigenschaften der Objekte häufig verschiedenen Arten der Transformation, Normierung, Skalierung oder Abbildung unterzogen.

Klassifikation: Die eigentliche Aufgabe eines Mustererkennungssystems besteht in der Klassifikation. Dabei wird die Zugehörigkeit eines Objektes zu einer bestimmten Kategorie oder Klasse anhand seiner Features entschieden.

Weiterverarbeitung: Nach der Klassifikation kann die getroffene Entscheidung unter der Berücksichtigung von **Fehlerkosten**[2] oder Kontexteffekten korrigiert werden. Eine andere Möglichkeit der Weiterverarbeitung bietet sich durch die Zusammenführung der Entscheidungen mehrerer Klassifikationen. Dabei kann mit Hilfe von verschiedenen Verfahren eine ausgewählte oder fusionierte Entscheidung getroffen werden.

Einige Strukturen der Mustererkennungssysteme sehen zusätzlich eine Feedback-Verbindung zwischen einzelnen Komponenten vor, um die Intelligenz und Anpassungsfähigkeit des Systems zu erhöhen.

Das Design eines Mustererkennungssystems führt allgemein zur Wiederholung verschiedener Aktivitäten: Zusammenstellung von Daten, Auswahl der Features, Auswahl des Modells, Training und Evaluierung [Duda 01] (siehe Abbildung 3.2).

Die Datenzusammenstellung enthält erstaunlicherweise den größten Aufwandsanteil bei der Entwicklung eines Mustererkennungssystems. Für die Entwicklung eines leistungsfähigen Mustererkennungssystems werden allgemein große Datenmengen benötigt, die vorläufig analysiert und für weitere Verwendung aufbereitet werden müssen.

[2] Auch Loss-Funktion oder Kostenfunktion genannt

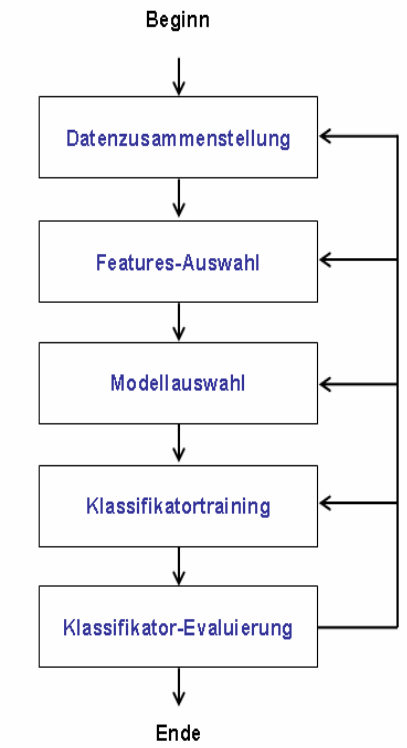


Abbildung 3.2: Design-Zyklus eines Mustererkennungssystems

Die Features Auswahl ist ein kritischer Schritt des Designs, von dem häufig die Funktionsfähigkeit des Gesamtsystems abhängt. Eine erfolgreiche Auswahl der Features setzt ausreichende Vorkenntnisse der Eigenschaften der Objekte voraus, anhand derer die Klassifikation erfolgen soll.

Um die Aussagekraft einzelner Features zu erhöhen, werden sie häufig miteinander kombiniert oder verschiedenen Transformationen unterzogen. Dabei gilt allgemein, dass die kausalen Abhängigkeiten einzelner Features das Erkennungsvermögen der Objekte reduzieren.

Die Modellauswahl erfolgt meist in mehreren Iterationsschritten, mit denen verschiedene Modelle auf den vorhandenen Daten ausprobiert werden können. Dabei ist die vorläufige Analyse der Verteilung von Features hilfreich und bestimmt häufig die allgemeine Richtung, in der ein Modell gesucht werden kann.

Das Klassifikatortraining findet häufig bereits bei der Auswahl des Modells statt, wenn verschiedene Modelle vorläufig ausprobiert werden. Soweit ein Modell ausgewählt und alle Einzelheiten der Weiterverarbeitung bestimmt sind, erfolgt das Training mit den vorhanden Daten, nach dem ein **trainierter** Klassifikator zur Verfügung steht.

Die Klassifikator-Evaluierung führt häufig zu einem neuen Durchlauf des ganzen Design-Zyklus oder seiner einzelnen Schritte, um die Funktionalität des Systems zu verbessern bzw. zu optimieren. Die Evaluierung erfolgt auf den Testdaten, die nicht zu den Trainingsdaten gehören sollten. Allerdings versucht

man häufig alle vorhandenen Daten für das Training des Systems zu verwenden, da allgemein die Qualität des Trainings mit dem Umfang der Trainingsdaten steigt.

Eine sehr effiziente Methode, die so genannte **Kreuzvalidierung** (weiter als *cross validation*), erlaubt allerdings die Verwendung derselben Daten für das Training und die Evaluierung, ohne dass die Daten sich dabei überlappen. Aus n Datenobjekten werden k ausgenommen, und das System wird anschließend mit den restlichen $n - k$ trainiert. Die Evaluierung erfolgt mit den k Datenobjekten, worauf folgend die anderen k Objekte aus den ursprünglichen n ausgenommen werden und der Trainings- / Evaluierungsvorgang wiederholt wird. Mit dem Verfahren können alle vorhandenen Daten sowohl für das Training als auch für die Evaluierung verwendet werden.

Mit verschiedenen graphischen und analytischen Methoden können die Fehler der Klassifikation analysiert und verglichen werden, um ein Feedback zu den früheren Design-Schritten zu ermöglichen.

Um nun ein besseres Verständnis für die datenbasierten Mustererkennungssysteme zu erhalten, wird ein Beispiel aus [Duda 01] ausführlich vorgestellt:

In einer Fischfabrik soll automatisch anhand eines Grauwertbildes zwischen Lachsen und Brassen unterschieden werden. Das System muss im laufenden Betrieb pro Fisch (Muster-Objekt) folgende Arbeitsschritte durchlaufen:

1. *Sensor-Messung (Bildaufnahme)*
2. *Vorverarbeitung (z.B. Rauschfilterung)*
3. *Segmentierung, Labeling*
4. *Merkmalsberechnung (Helligkeit, Länge)*
5. *Klassifikation*
6. *Weiterverarbeitung*

Wir beschäftigen uns im Folgenden ausschließlich mit den Punkten 4 und 5 (Merkmalsauswahl und Auswahl/Training des Klassifikators) als mit relevantesten Teilen eines Mustererkennungssystems.

Nehmen wir an, dass je 100 Brassen und Lachse vermessen wurden und uns somit 200 korrekt mit ihrer Klassen-Zugehörigkeit "gelabelte" Merkmalsvektoren zur Verfügung stehen. Die Güte eines Merkmals hängt davon ab, wie einfach und schnell es berechnet werden kann und wie "diskriminativ" es ist, d.h., wie gut es sich zwischen den interessierenden Klassen unterscheidet. Die Häufigkeit der ausgewählten Merkmale (Helligkeit, Länge) beider Fischarten wird in der Abbildung 3.3 [Duda 01] dargestellt.

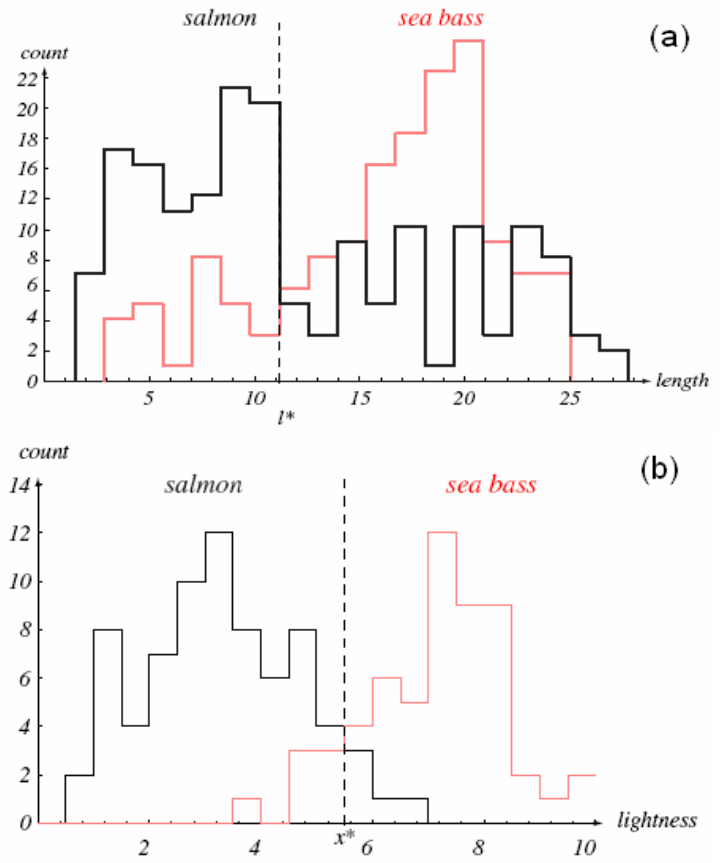


Abbildung 3.3: Häufigkeiten der Merkmale Länge (a) und Helligkeit (b)

Obwohl Lachse eher länger als Brassen sind, ist das Merkmal Länge für sich allein nur schlecht geeignet, um zwischen den beiden Fischarten zu unterscheiden. Die klassenspezifischen Ausprägungen des Merkmals Helligkeit überlappen sich zwar in geringerem Maße, jedoch lässt auch dieses Merkmal keine eindeutige, fehlerfreie Klassifikation zu. Die Kombination mehrerer Merkmale führt oft zu besseren Ergebnissen. Die beiden Klassen sind im zwei-dimensionalen Merkmalsraum bereits recht gut separiert, wie in Abbildung 3.4 [Duda 01] zu sehen ist.

Das nächste Problem ist die Auswahl eines geeigneten Klassifikators (Modells). In Abbildung 3.4(a) ist ein Beispiel für einen einfachen, linearen Klassifikator zu sehen: Dieser ist offensichtlich nicht in der Lage, die beiden Klassen fehlerfrei zu unterscheiden. Der in 3.4(b) stehende Klassifikator leistet zwar eine fehlerfreie Klassifikation der Trainingsdaten, jedoch auf Kosten einer komplexen Entscheidungsgrenze.

Nachdem man sich für einen bestimmten Klassifikator (Modell) entschieden hat, muss dieser noch auf den vorhandenen Daten trainiert werden; z.B. kann die Gerade in Abbildung 3.4(a) mittels der Methode der kleinsten Quadrate bestimmt werden.

Das Ziel des Designs/Trainings besteht letztendlich nicht darin, die Trainingsdaten, sondern die Gesamtheit aller Muster bzw. aller möglichen Merkmalsausprägungen korrekt bzw. mit möglichst geringem "mittleren Fehler" zu klassifizieren. Man spricht

in diesem Zusammenhang auch von der Generalisierungsfähigkeit des Klassifikators.

Während zu einfache Modelle bereits auf dem Trainingssatz zu schlechten Ergebnissen führen, weil sie die den Daten zugrundeliegende Struktur nicht erklären können, sind zu komplexe Modelle sehr sensitiv bezüglich der Auswahl der Trainingsdaten sowie auf zufällige Messfehler (Rauschen) in den Trainingsdaten. Letzteres kann ebenfalls zu schlechter Generalisierungsfähigkeit führen. Ein Beispiel für den quadratischen Klassifikator ("mittlere Komplexität") ist in der Abbildung 3.4 (c) dargestellt.

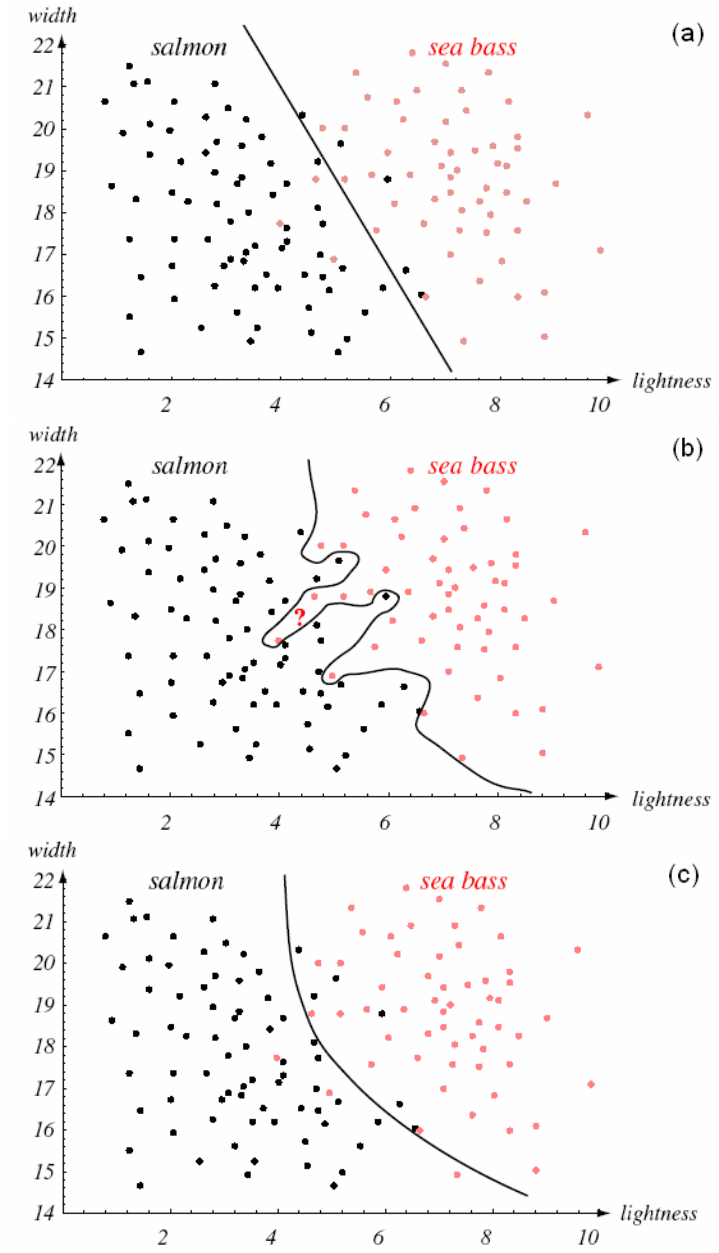


Abbildung 3.4: Merkmalsraum mit verschiedenen Klassifikationsgrenzen

3.2 Bayes'sche Klassifikation

3.2.1 Bayes'sche Entscheidungstheorie

Die Bayes'sche Entscheidungstheorie basiert auf dem Bayes'schen Theorem für bedingte Wahrscheinlichkeiten [Litz 01], welche besagt, dass:

$$P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)} \quad . \tag{3.2}$$

Dabei nennt man $P(B)$ die **a priori** Wahrscheinlichkeit von B, und $P(B|A)$ die **a posteriori** Wahrscheinlichkeit von B unter A. Repräsentiert insbesondere X ein Merkmal und w die Klassenzugehörigkeit von Mustern, so gibt im Falle der beobachteten Merkmalsausprägung $X = i$ oder kurz X_i

$$P(w_j|X_i) = \frac{P(X_i|w_j) \cdot P(w_j)}{P(X_i)} \tag{3.3}$$

die Wahrscheinlichkeit an, dass das Muster zur Klasse j gehört. Die Gleichung (3.3) transformiert die **a priori** Wahrscheinlichkeit $P(w_j)$, dass ein Muster in die j-te Klasse fällt, nach Beobachtung einer Merkmalsausprägung X_i in die a posteriori Wahrscheinlichkeit $P(w_j|X_i)$, welche diese zusätzliche Information über den Versuchsausgang widerspiegelt. Es gilt:

$$\sum_{j=1}^{c} P(w_j|X_i) = 1 \quad \textit{und} \quad P(X_i) = \sum_{j=1}^{c} P(X_i|w_j) \cdot P(w_j) \quad . \tag{3.4}$$

Wir nehmen im Folgenden eine stetige Merkmalsvariable X mit zugeordneter Wahrscheinlichkeitsdichtefunktion $p(\mathbf{x})$ an. Daraus folgend wird die Gleichung (3.3) zu

$$P(w_j|\mathbf{x}) = \frac{p(\mathbf{x}|w_j) \cdot P(w_j)}{p(\mathbf{x})} \quad . \tag{3.5}$$

Als Funktion von $\mathbf{x}$ wird $p(\mathbf{x}|w_j)$ als bedingte Wahrscheinlichkeitsdichtefunktion von $\mathbf{x}$ bezüglich w_j bezeichnet. Diese beschreibt die Verteilung des Merkmals X für eine gegebene Klasse w_j und besitzt alle Eigenschaften einer Dichtefunktion. Betrachtet man $p(\mathbf{x}|w_j)$ hingegen als Funktion der Klasse w_j für festes $\mathbf{x}$, so spricht man von der Wahrscheinlichkeit von w_j bezüglich $\mathbf{x}$ (*likelihood*). Den Nenner $p(\mathbf{x})$ in der Gleichung (3.5) erhält man analog zu (3.4) als:

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}|w_j) \cdot P(w_j) \quad . \tag{3.6}$$

Diese Wahrscheinlichkeit fungiert als Normierungsfaktor und stellt sicher, dass die Summe der a posteriori Wahrscheinlichkeiten über alle Klassen den Wert 1 ergibt. Man bemerkt jedoch, dass $p(\mathbf{x})$ für alle Klassen identisch ist und daher keinen Einfluss auf das Verhältnis der a posteriori Wahrscheinlichkeiten hat. Für die Bestimmung der Klasse mit der größten a posteriori Wahrscheinlichkeit ist daher das Verhältnis der mit den korrespondierenden a priori gewichteten Wahrscheinlichkeiten $p(\mathbf{x}|w_j) \cdot P(w_j)$ hinreichend.

Abbildung 3.5 [Duda 01] zeigt ein Beispiel mit bedingten Wahrscheinlichkeitsdichtefunktionen und korrespondierenden a posteriori Wahrscheinlichkeiten für zwei Klassen mit $P(w_1) = 2/3$ und $P(w_2) = 1/3$.

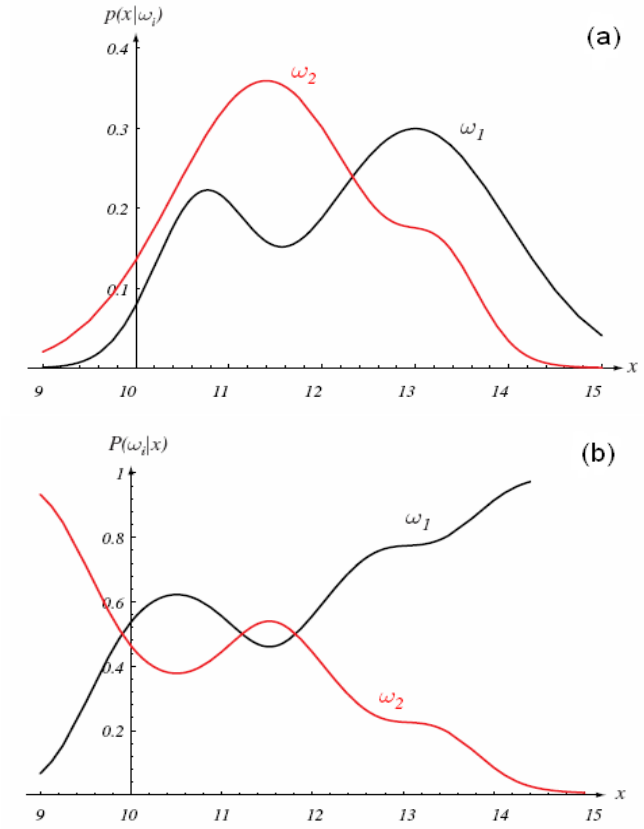


Abbildung 3.5: Dichtefunktion (a) und a posteriori Wahrscheinlichkeiten (b)

Die obigen Überlegungen führen für den Fall $c = 2$ zu folgender Formulierung der Bayes'schen Entscheidungsregel [Duda 01]:

Definition 3.1 (Bayes'sche Entscheidungsregel):
Entscheide für w_1, falls

$$P(w_1|\mathbf{x}) > P(w_2|\mathbf{x}) \quad \textit{oder} \quad p(\mathbf{x}|w_1) \cdot P(w_1) > p(\mathbf{x}|w_2) \cdot P(w_2) \quad . \tag{3.7}$$

Laut Bayes-Theorem (siehe Gleichung (3.2)) ergibt sich für jede Merkmalsausprägung $\mathbf{x}$ die bedingte Wahrscheinlichkeit der Fehlklassifikation $P(error|\mathbf{x})$ zu

– $P(w_2|\mathbf{x})$, falls man sich für w_1 entscheidet,

– $P(w_1|\mathbf{x})$, falls man sich für w_2 entscheidet.

Da die Erwartung einer Funktion $h(X)$ einer Zufallsvariable X als

$$E(h(X)) = \int_{-\infty}^{+\infty} h(X) \cdot p(\mathbf{x})dx \tag{3.8}$$

definiert ist, berechnet sich der mittlere Fehler $P(error)$ (die Fehlerrate) als

$$P(error) = \int_{-\infty}^{+\infty} P(error|\mathbf{x}) \cdot p(\mathbf{x})dx \quad . \tag{3.9}$$

Die Bayes'sche Entscheidungsregel entscheidet für die Klasse w_k mit der höchsten a posteriori Wahrscheinlichkeit

$$k = \arg\max_j P(w_j|\mathbf{x}) \quad . \tag{3.10}$$

Daher ergibt sich die bedingte Fehlerwahrscheinlichkeit $P(error|x)$ zu

$$\min[P(w_1|\mathbf{x}), P(w_2|\mathbf{x})] = 1 - \max[P(w_1|\mathbf{x}), P(w_2|\mathbf{x})] \quad . \tag{3.11}$$

Die Bayes'sche Entscheidungsregel minimiert also den Integranden $P(error|\mathbf{x})$ in Gleichung 3.9 für jede Merkmalsausprägung $\mathbf{x}$ und folglich auch die mittlere Fehlerwahrscheinlichkeit $P(error)$.

Die Entscheidungsfunktion $\alpha(\mathbf{x}) : \mathbf{x} \mapsto j$ assoziiert mit jeder Merkmalsausprägung x eine bestimmte Aktion j, im Allgemein die Zuweisung eines Klassenlabels $j \in \{1 \dots c\}$. Dabei partitioniert α den Merkmalsraum vollständig in c disjunkte Entscheidungsregionen R_i und es gilt

$$R_i = \{\mathbf{x} : \alpha(\mathbf{x}) = i\} \quad . \tag{3.12}$$

Die Grenze zwischen jeweils zwei Entscheidungsregionen wird als Entscheidungsgrenze bezeichnet, entlang welcher die sogenannten Schwellen in Form von Merkmalsausprägungen bestehen, welche bezüglich des gewählten Klassifikationskriteriums, z.B. der a posteriori Wahrscheinlichkeit, denselben Wert erzielen. Dabei müssen die Entscheidungsregionen nicht unbedingt zusammenhängend sein.

Im Fall der Bayes'schen Entscheidungsregel verschieben größere a priori Wahrscheinlichkeiten die Entscheidungsgrenze in Richtung der a priori weniger wahrscheinlichen Klassen.

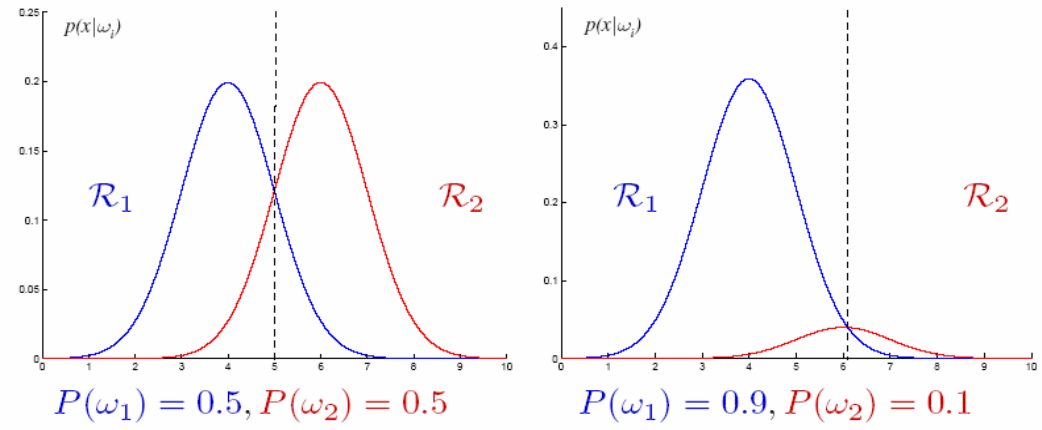


Abbildung 3.6: Entscheidungsgrenzen abhängig von a priori Wahrscheinlichkeiten

Die in Abbildung 3.6 dargestellte Entscheidungsgrenze (schwarz gestrichelt) trennt die korrespondierenden Entscheidungsregionen R_1 und R_2 für zwei Klassen w_1 und w_2 mit normalverteilten Merkmalen. Dabei werden die Dichtefunktionen für beide Klassen mit ihrer a priori Wahrscheinlichkeit gewichtet: $p(\mathbf{x}|w_1)P(w_1)$, $p(\mathbf{x}|w_2)P(w_2)$.

Loss-Function $L(\alpha(\mathbf{x}), w_j)$ gibt die mit der Entscheidung $\alpha(x)$ verbundenen Kosten an, wenn die wahre Klassenzugehörigkeit durch w_j gegeben ist. Es gilt

$$L(\alpha(\mathbf{x}), w_j) = \begin{cases} K, & \text{wenn } \alpha(\mathbf{x}) \neq j \\ 0, & \text{wenn } \alpha(\mathbf{x}) = j \end{cases} , \tag{3.13}$$

wobei K den Kostenwert einer falschen Entscheidung angibt. Der für eine gegebene Merkmalsausprägung x im Mittel erwartete Wert der Loss-Funktion ergibt sich zu

$$r(\alpha(\mathbf{x}), \mathbf{x}) = \sum_{j=1}^{c} L(\alpha(\mathbf{x}), w_j) \cdot P(w_j|\mathbf{x}) \tag{3.14}$$

und wird als **Risiko** genannt. Da $r(\alpha(\mathbf{x})|\mathbf{x})$ den Erwartungswert von L bezüglich aller Klassen an der Stelle $\mathbf{x}$ berechnet, wird $r(\alpha(\mathbf{x})|\mathbf{x})$ auch als bedingtes Risiko bezüglich $\mathbf{x}$ bezeichnet. Offensichtlich hängt $r(\alpha(\mathbf{x})|\mathbf{x})$ von $\alpha(\mathbf{x})$ ab. Um die optimale Entscheidung im Punkt $\mathbf{x}$ zu bestimmen, führen wir zunächst folgende Kurzbezeichnung ein: Sei λ_{ij} der Wert der Loss-Funktion in dem Fall, dass $\mathbf{x}$ zur Klasse w_j gehört und die Entscheidungsfunktion $\alpha(\mathbf{x})$ den Wert i zurückliefert (kurz: α_i), mit

$$\lambda_{ij} = L(\alpha_i, w_j) \quad . \tag{3.15}$$

Die Gleichung (3.14) lässt sich somit folgendermaßen schreiben

$$r(\alpha_i, \mathbf{x}) = \sum_{j=1}^{c} \lambda_{ij} \cdot P(w_j|\mathbf{x}) \quad . \tag{3.16}$$

Das Risiko $r(\alpha_i, \mathbf{x})$ wird in jedem Punkt $\mathbf{x}$ minimal, wenn die Entscheidungsfunktion $\alpha(\mathbf{x})$ die Bayes'sche Entscheidungsregel implementiert, d.h. das Label der Klasse mit der größten a posteriori Wahrscheinlichkeit zurückliefert (siehe Gleichung 3.10).

Für den bereits betrachteten Fall von 2 Klassen ($c = 2$) schreiben wir die Gleichung (3.16) für die beiden möglichen Entscheidungen $\alpha(\mathbf{x}) = 1$ und $\alpha(\mathbf{x}) = 2$ explizit aus und erhalten

$$\begin{aligned} r(\alpha_1, \mathbf{x}) &= \lambda_{11} \cdot P(w_1|\mathbf{x}) + \lambda_{12} \cdot P(w_2|\mathbf{x}) \\ r(\alpha_2, \mathbf{x}) &= \lambda_{21} \cdot P(w_1|\mathbf{x}) + \lambda_{22} \cdot P(w_2|\mathbf{x}) \quad . \end{aligned} \tag{3.17}$$

Um das Risiko im Punkt $\mathbf{x}$ zu minimieren, entscheidet man sich für die Klasse w_1, falls

$$\begin{aligned} r(\alpha_2, \mathbf{x}) &> r(\alpha_1, \mathbf{x}) && \textit{oder} \\ \lambda_{21} \cdot P(w_1|\mathbf{x}) + \lambda_{22} \cdot P(w_2|\mathbf{x}) &> \lambda_{11} \cdot P(w_1|\mathbf{x}) + \lambda_{12} \cdot P(w_2|\mathbf{x}) && \textit{oder} \\ (\lambda_{21} - \lambda_{11}) \cdot p(\mathbf{x}|w_1) \cdot P(w_1) &> (\lambda_{12} - \lambda_{22}) \cdot p(\mathbf{x}|w_2) \cdot P(w_2) && \textit{oder} \\ \lambda_{21} \cdot p(\mathbf{x}|w_1) \cdot P(w_1) &> \lambda_{12} \cdot p(\mathbf{x}|w_2) \cdot P(w_2) && , \end{aligned} \tag{3.18}$$

wobei $\lambda_{11} = \lambda_{22} = 0$ die Kosten der richtigen Entscheidung angibt, wie in (3.13) definiert. Man sieht, dass die Loss-Funktion effektiv die a priori Wahrscheinlichkeiten neu gewichtet und somit die Entscheidungsgrenze von der stärker gewichteten Klasse weg verschiebt, wie in der Abbildung 3.7 gezeigt.

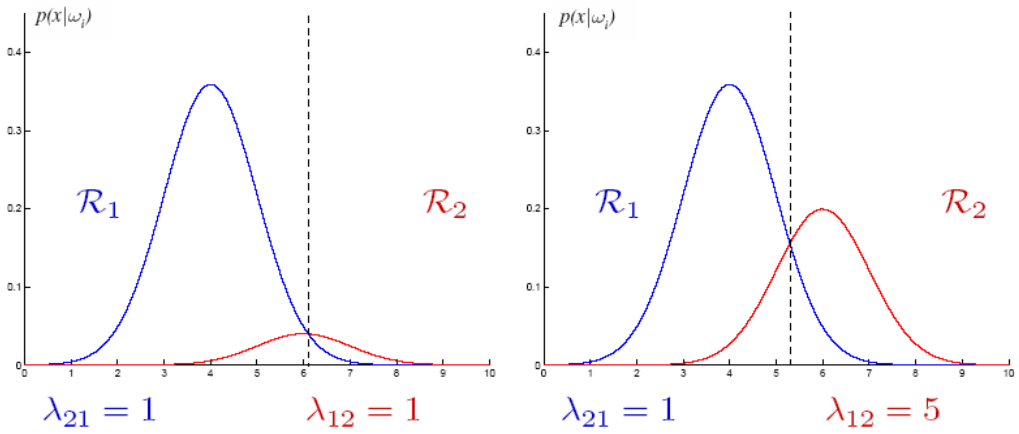


Abbildung 3.7: Entscheidungsgrenzen abhängig von der Loss-Funktion

3.2.2 Multivariante Normalverteilung

Die **Normalverteilung** findet ihre Anwendung in mehreren Bereichen, wie zum Beispiel bei Qualitätsabweichungen von Serienteilen oder bei additiven Messfehlern [Litz 01]. Neben ihrer Bedeutung für die Beschreibung solcher und vieler anderer technischer Vorgänge ist die Normalverteilung auch vom großen theoretischen Interesse. Sie beschreibt nämlich einige Zusammenhänge, die man erhält, wenn bestimmte Größen gegen unendlich streben, wie etwa die Zahl der Versuche oder die Anzahl der Summanden in einer Summe von Zufallsvariablen.

Die Dichtefunktion der Normalverteilung einer Zufallsvariable X lautet

$$f_x(x) = \frac{1}{\sigma_x\sqrt{2\pi}} \cdot \exp^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}} \quad . \tag{3.19}$$

Man sagt, X sei $N(\mu, \sigma^2)$-verteilt, also normalverteilt mit dem stochastischem Mittelwert μ und Varianz σ^2. Die Verteilungs- und Dichtefunktion einer normalverteilten Zufallsvariable sind in der Abbildung 3.8 dargestellt, wobei der Mittelwert μ bei 0 und die Varianz σ^2 bei 1 liegen (**Standardnormalverteilung**). Den Verlauf der Dichtefunktion bezeichnet man als **Gauß'sche Glockenfunktion**.

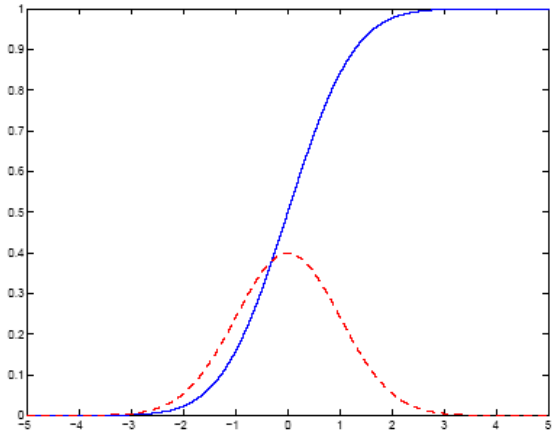


Abbildung 3.8: Verteilungs- und Dichtefunktion normalverteilter Zufallsvariable

Interessanterweise gilt nun, dass die Faltung einer großen Zahl von positiven Funktionen näherungsweise die Form einer Gauß'schen Glockenfunktion besitzt. Dies ist die Aussage des **zentralen Grenzwertsatzes** [Litz 01], der lautet:

Satz 3.1 (Zentraler Grenzwertsatz):
Sind $X_1, X_2, \ldots, X_n$ ***stochastisch unabhängige*** *Zufallsvariablen mit beliebigen Verteilungen, dann tendiert unter bestimmten allgemeinen Bedingungen die Summme $Z = \sum_{i=1}^{n} X_i$ für $n \to \infty$ zur Normalverteilung.*

Zu den allgemeinen Bedingungen gehört, dass keine der Zufallsvariablen derart dominant sein darf, dass die anderen in der Summe keine Rolle mehr spielen. Es soll

sich also um eine Summe von im wesentlichen "gleichberechtigten" Summanden handeln. Mit derartigen Summanden hat man es bei technischen Problemen oft zu tun [Litz 01]. Eine bestimmte interessierende Größe setzt sich oft näherungsweise additiv aus mehreren Einflussgrößen zusammen, die voneinander unabhängig sind und unterschiedlichen, oft unbekannten, Verteilungen gehorchen. Hat die Summe nur hinreichend viele Summanden, kann man erwarten, dass sie einer Normalverteilung gehorcht. Diese hat mit μ und σ^2 nur zwei Bestimmungsgrößen, die sich experimentell recht einfach bestimmen lassen.

Sind mehrere normalverteilte Zufallsvariblen vorhanden, spricht man von einer Verbundnormalverteilung, die durch Angabe des Mittelwertvektors $\vec{\mu}$ und der Kovarianzmatrix Σ vollständig festgelegt ist. Dabei gilt der Satz der **stochastischen Unabhängigkeit** und **Unkorreliertheit** bei Normalverteilung [Litz 01], der lautet:

Satz 3.2 (Stochastische Unabhängigkeit):
Sind verbundnormalverteilte Zufallsvariablen $X_1, X_2, \ldots, X_n$ ***unkorelliert*** *(Kovarianzen $C_{x_i x_j} = 0$), dann sind sie auch stochastisch unabhängig.*

Die Dichtefunktion eines normalverteilten Zufallsvektors $\vec{X} \sim N(\vec{\mu}, \Sigma)$ mit Mittelwert $\vec{\mu}$ und Kovarianzmatrix Σ ist wie folgt definiert [Duda 01]

$$p(\vec{x}) = \frac{1}{(2\pi)^{d/2}\,|\Sigma|^{1/2}}\ \exp^{-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1}(\vec{x}-\vec{\mu})} \quad , \tag{3.20}$$

wobei $\vec{x}$ für die Realisierung des d-dimensionalen Zufallsvektors $\vec{X}$, $\vec{\mu}$ für den d-dimensionalen Mittelwert-Vektor, $|\Sigma|$ und Σ^{-1} für die Determinante und die Inverse der $d \times d$-Kovarianzmatrix Σ und T für das Transponieren eines Vektors bzw. einer Matrix stehen. Die Abbildung 3.9 zeigt ein Beispiel für eine zweidimensionale Normalverteilung.

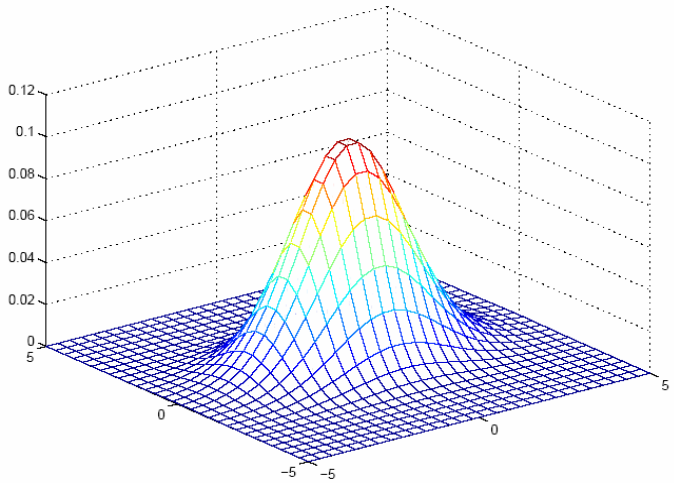


Abbildung 3.9: Eine zweidimensionale Normalverteilung

Die Kovarianzmatrix Σ hat die Form

$$\Sigma = E[(\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T] = \int (\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T p(x)dx \quad , \tag{3.21}$$

wobei E für den Erwartungswert steht. Falls x_i die i-te Komponente des Vektors $\vec{x}$, μ_i die i-te Komponente des Vektors $\vec{\mu}$ und σ_{ij} die ij-te Komponente der Matrix Σ darstellen, gilt

$$\mu_i = E[x_i] \quad und \quad \sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)] \quad . \tag{3.22}$$

Die Kovarianzmatrix Σ ist immer symmetrisch und positiv **semidifinit**. Wir beschränken uns auf den Fall, dass die Kovarianzmatrix Σ positiv **definit** ist, so dass ihre Determinante $|\Sigma|$ immer positive Werte hat. Andernfalls ($|\Sigma| = 0$) ist $p(\vec{x})$ degeneriert, d.h. entweder ist die Varianz eines Elementes des $\vec{x}$-Vektors gleich 0 oder zwei Elemente des x-Vektors sind linear abhängig und führen zur Redundanz der Zufallsvariablen. Die Diagonalelemente σ_{ii} der Kovarianzmatrix Σ sind die Varianzen (im allgemein σ_i^2) der entsprechenden Zufallsvariablen x_i, und die Nicht-Diagonalelemente σ_{ij} sind die Kovarianzen der Zufallsvariablen x_i und x_j. Sind dabei alle Zufallsvariablen stochastisch unabhängig (mit $\sigma_{ij} = 0$), wird die Kovarianzmatrix Σ zu einer **Diagonalmatrix**, und die Wahrscheinlichkeit $p(\vec{x})$ reduziert sich auf das Produkt der einzelnen Dichtefunktionen der normalverteilten Komponenten des Vektors $\vec{x}$.

Der Exponent in Gleichung (3.20) hängt vom Wert der quadratischen Form

$$(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu}) = m^2(\vec{x}) \tag{3.23}$$

ab, welche auch als **Mahalanobis-Distanz** bezeichnet wird [Fuku 90]. Die Menge aller Punkte $\{x : m^2(\vec{x}) = c\}$, für welche die Mahalanobis-Distanz gleich einer Konstanten c ist, ist durch ein Hyperellipsoid im $\mathbb{R}^d$ mit Mittelpunkt $\vec{\mu}$ gegeben.

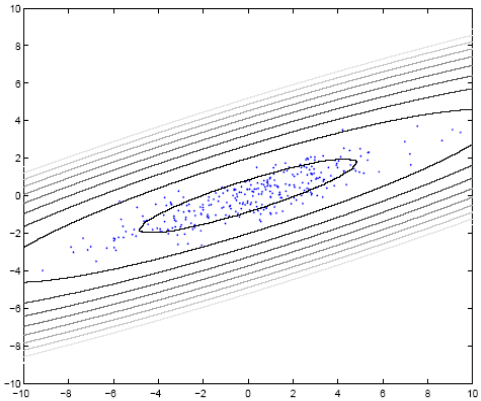


Abbildung 3.10: Mahalanobis-Distanz

Für alle auf einem solchen Hyperellipsoid liegenden Punkte liefert die Wahrscheinlichkeit $p(\vec{x})$ denselben Wert. Dies wird in Abbildung 3.10 veranschaulicht.

Für den Fall der zweidimensionalen Normalverteilung ergibt sich eine Ellipse, wobei die Hauptachse in Richtung der größten Varianz liegt. Das Verhältnis der Achsen der Ellipse hängt vom Absolutbetrag des Korrelationskoeffizienten ρ (siehe 4.1.2) ab: Je größer $|\rho|$ ist, desto elongierter bzw. je kleiner $|\rho|$, desto kreisförmiger erscheint die Ellipse. Für $\rho = 0$ (die Zufallsvariablen sind stochastisch unabhängig) erhält man einen Kreis.

3.2.3 Diskriminanten-Funktionen

Gemäß der Bayes'schen Entscheidungsregel, die in (3.7) bereits definiert wurde, wähle man beim gegebenen Merkmalsvektor $\vec{x} \in \mathbb{R}^d$ die Klasse w_k mit der größten a posteriori Wahrscheinlichkeit

$$\alpha(\vec{x}) = k = \arg \max_j P(w_j|\vec{x}) \quad 1 \leq j \leq c \quad . \tag{3.24}$$

Die Enscheidungsfunktion $\alpha(\vec{x})$ lässt sich allgemein durch so genannte **Diskriminanten-Funktionen** $g_j(\vec{x})$ ausdrücken

$$\alpha(\vec{x}) = k = \arg \max_j g_j(\vec{x}) \quad . \tag{3.25}$$

Die Entscheidungsgrenze zwischen den Klassen w_j und w_k ist durch die Gleichung $g_j(\vec{x}) = g_k(\vec{x})$ gegeben. Berechnen sich die $g_j(\vec{x})$ als streng monoton wachsende Funktion der a posteriori Wahrscheinlichkeit

$$g_j(\vec{x}) = f(P(w_j|\vec{x}))\,, \quad wobei \quad x > y \Rightarrow f(x) > f(y) \quad , \tag{3.26}$$

so ist die Enscheidungsregel (Gleichung (3.25)) wiederum optimal, zum Beispiel für

$$g_j(\vec{x}) = P(w_j|\vec{x}) \cdot p(\vec{x}) = \frac{p(\vec{x}|w_j) \cdot P(w_j)}{p(\vec{x})} \cdot p(\vec{x}) = p(\vec{x}|w_j) \cdot P(w_j) \quad . \tag{3.27}$$

Sind im speziellen die Merkmale für alle Klassen normalverteilt, d.h. $(X|w_j) \sim N(\mu_j, \Sigma_j)$, so lässt sich die Verteilungsdichtefunktion gemäß Gleichung (3.20) folgendermaßen aufschreiben

$$p(\vec{x}|w_j) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_j|^{1/2}}\, \exp^{-\frac{1}{2}(\vec{x}-\vec{\mu}_j)^T \Sigma_j^{-1} (\vec{x}-\vec{\mu}_j)} \quad . \tag{3.28}$$

Durch das Logarithmieren der a posteriori Wahrscheinlichkeit erhält man die folgenden Diskriminanten-Funktionen

$$\begin{aligned} g_j(\vec{x}) &= \ln \frac{p(\vec{x}|w_j) \cdot P(w_j)}{p(\vec{x})} = \\ &= -\frac{1}{2}(\vec{x} - \vec{\mu}_j)^T \Sigma_j^{-1}(\vec{x} - \vec{\mu}_j) - \frac{1}{2}\ln|\Sigma_j| + \ln P(w_j) - \frac{d}{2}\ln 2\pi - \ln p(\vec{x}) \,. \end{aligned} \tag{3.29}$$

Man bemerkt, dass die beiden Terme in der letzten Zeile

$$-\frac{d}{2}\ln 2\pi - \ln p(\vec{x})$$

nicht von w_j abhängen und daher beim Vergleich der g_j nicht berücksichtigt werden müssen. Die g_j sind im Falle normalverteilter Merkmale in $\vec{x}$ quadratische Funktionen

$$g_j(\vec{x}) = -\frac{1}{2}m_j^2(\vec{x}) - \frac{1}{2}\ln|\Sigma_j| + \ln P(w_j) \quad , \tag{3.30}$$

wobei $m_j^2(\vec{x})$ die in (3.23) bereits eingeführte Mahalanobis-Distanz der Klasse w_j ist.

Im allgemeinen Fall des n-dimensionalen Merkmalsraums hat die Etscheidungsgrenze, die durch Diskriminanten-Funktionen gegeben wird, die Form einer gekrümmter Hyperebene. Im einfachsten Fall des zweidimensionalen Merkmalsraums wird eine Hyperebene zu einer Gerade oder einer Kurve zweites Grades.

Wir betrachten im Folgenden zwei Fälle, die zum linearen und quadratischen Verlauf der Entscheidungsgrenze führen und entsprechend als linearer bzw. quadratischer Bayes-Klassifikatoren bezeichnet werden.

3.2.4 Linearer Bayes-Klassifikator

Unterliegen die Merkmale aller Klassen w_j gleicher Struktur einer multivarianten Normalverteilung, d.h. alle Klassen haben dieselbe Kovarianzmatrix Σ, spricht man von einem linearen Bayes-Klassifikator.

Schreibt man in Gleichung (3.30) die Mahalonobis-Distanz $m_j^2(\vec{x})$ aus und lässt den von w_j unabhängigen Term $-\frac{1}{2}\ln|\Sigma_j|$ weg, so erhält man

$$g_j(\vec{x}) = -\frac{1}{2}(\vec{x} - \vec{\mu}_j)^T \Sigma_j^{-1}(\vec{x} - \vec{\mu}_j) + \ln P(w_j) \quad . \tag{3.31}$$

Die Mahalonobis-Distanz $m_j^2(\vec{x})$ zerfällt in einen quadratischen und einen affinen Anteil

$$m_j^2(\vec{x}) = \vec{x}^T\Sigma^{-1}\vec{x} - 2\vec{\mu}_j^T\Sigma^{-1}\vec{x} + \vec{\mu}_j^T\Sigma^{-1}\vec{\mu}_j \quad , \tag{3.32}$$

wobei der quadratische Anteil $\vec{x}^T\Sigma^{-1}\vec{x}$ wiederum nicht von w_j abhängt, und somit weggelassen werden kann. Nun ergeben sich Diskrimininanten-Funktionen zu folgender Form:

$$g_j(\vec{x}) = \vec{\mu}_j^T\Sigma^{-1}\vec{x} - \frac{1}{2}\vec{\mu}_j^T\Sigma^{-1}\vec{\mu}_j + \ln P(w_j) \quad . \tag{3.33}$$

Weil Diskrimininanten-Funktionen linear sind, besitzen die Entscheidungsgrenzen die Form einer geraden Hyperebene, wie am Beispiel von zwei bivariaten Normalverteilungen mit $\Sigma_1 = \Sigma_2$ in Abbildung 3.11 [Duda 01] gezeigt ist, wobei eine gerade Hyperebene im zweidimensionalen Merkmalsraum zu einer Gerade wird.

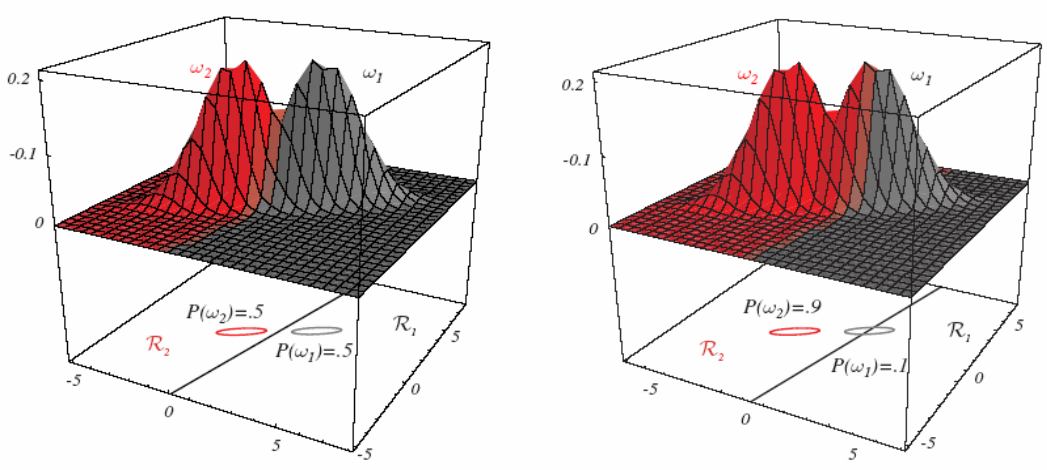


Abbildung 3.11: Lineare Entscheidungsgrenzen für zwei bivariate Normalverteilungen

Für gleiche a priori Wahrscheinlichkeiten verläuft die Entscheidungsgrenze durch $(\mu_i + \mu_j)/2$, ansonsten wird sie von der a priori wahrscheinlicheren Klasse weg verschoben, wie in 3.2.1 bereits erwähnt.

3.2.5 Quadratischer Bayes-Klassifikator

Im Falle der beliebigen Kovarianzmatrix Σ_j bei allen Klassen w_j spricht man von einem quadratischen Bayes-Klassifikator, deren Diskriminanten-Funktionen sich aus Gleichung (3.30) folgendermaßen ausschreiben lassen:

$$g_j(\vec{x}) = -\frac{1}{2}\vec{x}^T\Sigma^{-1}\vec{x} + \vec{\mu}_j^T\Sigma^{-1}\vec{x} - \frac{1}{2}\vec{\mu}_j^T\Sigma^{-1}\vec{\mu}_j - \frac{1}{2}\ln|\Sigma_j| + \ln P(w_j) \quad . \tag{3.34}$$

Die Entscheidungsgrenzen sind durch so genannte *Hyperquadrics* gegeben (siehe Abbildung 3.12 [Duda 01]), wobei die korrespondierenden Entscheidungsregionen nicht unbedingt zusammenhängend sein müssen.

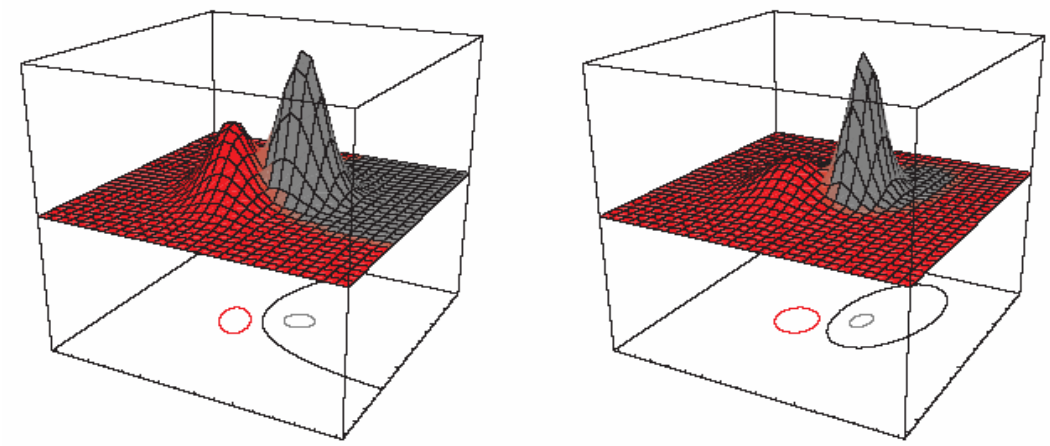


Abbildung 3.12: Quadratische Entscheidungsgrenzen für zwei bivariate Normalverteilungen

Welche genaue Form das jeweilige Hyperquadric annimmt (im zweidimensionalen Fall: Gerade, Ellipse, Parabel oder Hyperbel) hängt von den Kovarianzmatrizen aller zu unterscheidenden Klassen ab.

3.2.6 Parameterschätzung und Hypothesentest

Oft weiß man aufgrund der Erfahrung, dass sich bestimmte technische Probleme mit bestimmten Verteilungen diskreter oder stetiger Zufallsvariablen beschreiben lassen. Sie besitzen meist nur wenige freie Parameter, die oft auch noch in einfacher Weise mit dem Erwartungswert und der Varianz der Zufallsvariable zusammenhängen. Hat man diese Parameter bestimmt, ist auch die Verteilung bekannt [Litz 01]. Man nennt die Bestimmung derartiger Parameter ausgehend von einer Stichprobe eine Parameterschätzung. Bezeichnet man den gesuchten Parameter (auch der wahrer Parameter genannt) allgemein mit ϑ, so liefert das Schätzverfahren einen Näherungswert. Man nennt ihn Schätzwert $\hat{\vartheta}$. Man kann ihn als Realisierung einer Zufallsvariable $\hat{\Theta}$ auffassen, die eine Funktion des Stichprobenvektors $\vec{X}$ darstellt. Sie heißt Schätzfunktion oder kurz **Schätzer**:

$$\hat{\Theta}_n = g_n(X_1, X_2, \ldots, X_n) = g_n(X) \quad . \tag{3.35}$$

Offensichtlich handelt es sich beim Schätzer um eine Stichprobenfunktion. Der index n ist bei Schätzfunktionen üblich und soll auf die Anzahl n der Zufallsvariablen im Stichprobenvektor hinweisen. Es ist nun im qualitativen Sinn möglich, die Güte

eines Schätzers unabhängig von dessen Realisierung zu beurteilen. Dies geschieht anhand dreier wichtiger Eigenschaften, die im Folgenden kurz erläutert werden: die **Erwartungstreue**, die **Konsistenz** und die **Effizienz**.

Definition 3.2 (Erwartungstreue und asymptotische Erwartungstreue):
Ein Schätzer $\hat{\Theta}_n = g_n(X)$ *heißt erwartungstreu, falls gilt:*

$$E(\hat{\Theta}_n) = \vartheta \quad . \tag{3.36}$$

Die Folge $\hat{\Theta}_n, \hat{\Theta}_{n+1}, \ldots$ *heißt asymptotisch erwartungstreu, falls gilt:*

$$\lim_{n\to\infty} E(\hat{\Theta}_n) = \vartheta \quad . \tag{3.37}$$

Bei einem erwartungstreuen Schätzer ist der Erwartungswert der Zufallsvariable $\hat{\Theta}_n$ für beliebige n gleich dem wahren Wert. Bei einem asymptotischen erwartungstreuen Schätzer gilt dies nur für $n \to \infty$, also für "sehr große n". Asymptotische Erwartungstreue ist demzufolge die schwächere Eigenschaft.

Definition 3.3 (Konsistenz):
Ein Schätzer $\hat{\Theta}_n = g_n(X)$ *heißt konsistent, falls gilt:*

$$\lim_{n\to\infty} P(|\hat{\Theta}_n - \vartheta| > \varepsilon) = 0 \quad . \tag{3.38}$$

Dies bedeutet: die Wahrscheinlichkeit dafür, dass die durch den Schätzer gebildete Zufallsvariable $\hat{\Theta}_n$ betragsmäßig um mehr als ein beliebig kleines positives ε vom wahren Wert abweicht, geht gegen Null, falls der Stichprobenumfang n gegen unendlich geht. Man sagt dafür auch "der Schätzer wird für zunehmenden Stichprobenumfang immer besser". Allerdings existiert ein einfaches Kriterium zur Konsistenz, das sich der Momente bedient:

Satz 3.3 (Hinreichendes Konsistenzkriterium):
Ein Schätzer $\hat{\Theta}_n = g_n(X)$ *heißt konsistent, falls gilt:*

$$E(\hat{\Theta}_n) = \vartheta \quad , \quad \textit{und} \qquad \lim_{n\to\infty} E((\hat{\Theta}_n - \vartheta)^2) = 0 \tag{3.39}$$

In Worten: jeder erwartungstreue Schätzer mit asymptotisch verschwindender Varianz ist konsistent.

Definition 3.4 (Effizienz):
Ein erwartungstreuer Schätzer $\hat{\Theta}_{n,e} = g_{n,e}(X)$ *heißt effizient oder wirksam, falls für alle anderen erwartungstreuen Schätzer* $\hat{\Theta}_{n,a} = g_{n,a}(X)$ *gilt:*

$$E((\hat{\Theta}_{n,e} - \vartheta)^2) < E((\hat{\Theta}_{n,a} - \vartheta)^2) \quad \forall\, a \neq e \quad . \tag{3.40}$$

Somit handelt es sich bei einem effizienten Schätzer bei jedem n um denjenigen mit der kleinsten Varianz. Die Erwartungstreue muss bei der Effizienz vorausgesetzt werden, da eine noch so kleine Varianz nichts nutzt, wenn der Ensemblemittelwert nicht auf dem wahren Wert liegt.

Schätzung des Mittelwertes

Gegeben seien n d-dimensionale Beobachtungen x_i, welche als Spaltenvektoren in der **sample matrix** $X = (x_1, \ldots, x_n) \in \mathbb{R}^{d \times n}$ zusammenfasst werden. Der erwartungstreue Schätzer des Mittelwerts ergibt sich als (ohne Beweis)

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i \quad . \tag{3.41}$$

Dabei beachte man, dass $\hat{\mu}$ wiederum ein Zufallsvektor ist.

Schätzung der Kovarianz-Matrix

Ein erwartungstreuer Schätzer der Kovarianz ist durch

$$\hat{\Sigma} = (\hat{\sigma}_{ij}) = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \hat{\mu}) \cdot (X_i - \hat{\mu})^T \quad , \tag{3.42}$$

gegeben (ohne Beweis). Alle Komponenten $\hat{\sigma}_{ij}$ sind wiederum Zufallsvariablen und $\hat{\Sigma}$ somit eine Zufallsmatrix.

Maximum likelihood-Methode

Maximum *likelihood*-Methode (ML) zur Schätzung der Parameter fasst die Stichprobe, genauer gesagt deren Realisation, als Funktion des gesuchten Parameters θ (*likelihood*-Funktion) auf

$$l(\theta, X) = p(X|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \quad , \tag{3.43}$$

wobei der letzte Schritt (Produktbildung) aus der Unabhängigkeit der X_i folgt. Dabei muss beachtet werden, dass die *likelihood*-Funktion $p(X|\theta)$ als Funktion des Parameters θ keine Dichtefunktion ist.

Die ML-Methode wählt jenen Wert des Parameters θ, welcher die gemeinsame Wahrscheinlichkeit (Gleichung 3.43) maximiert. Oft ist es einfacher, den Logarithmus der Gleichung 3.43 zu maximieren; dies führt zur log-*likelihood*-Funktion

$$\ln l(\theta, X) = p(X|\theta) = \sum_{i=1}^{n} \ln p(x_i|\theta) \quad . \tag{3.44}$$

Aus der Gleichungen (3.43) und (3.43) folgt, dass der Verlauf der *likelihood*-Funktion $p(X|\theta)$ mit steigendem Umfang der Stichprobe n enger wird und der geschätzte Wert des Parameters θ sich seinem wahren Wert nähert.

Hypothesentest

Bei einem Hypothesentest soll ausgehend von einer Stichprobe eine Ja-Nein-Entscheidung zu einer angenommenen Hypothese gefällt werden. Man spricht bei diesen Verfahren auch von Tests. Die Ja-Nein-Entscheidung kann z.B. darin bestehen, eine bestimmte Verteilungsform überhaupt anzunehmen oder nicht [Litz 01].

Alle Hypothesentests gehen von zwei Hypothesen aus, nämlich von der Prüf- oder Nullhypothese H_0 und der Alternativhypothese H_1, wobei sich beide Hypothesen gegenseitig ausschließen müssen. Die Entscheidung für die eine oder die andere Hypothese fällt anhand einer speziellen Stichprobenfunktion, der so genannten Prüffunktion oder Teststatistik. Falls außer dem Merkmal X keine weiteren Merkmale eine Rolle spielen, hat sie die Form

$$T = g_T(X_1, X_2, \cdots, X_n) \quad . \tag{3.45}$$

T muss eine bekannte Verteilung bzw. Dichtefunktion f_T besitzen und die Information enthalten, über die entschieden werden soll. Zum Beispiel wären dies bei einem Parametertest der Parameter und seine Schätzerfunktion. Die Teststatistik liefert beim Einsetzen einer Realisierung x des Stichprobenvektors den numerischen Wert

$$t = g_T(x_1, x_2, \cdots, x_n) \quad . \tag{3.46}$$

Über- bzw. unterschreitet t eine bestimmte Grenze, folgt daraus die Annahme oder Ablehnung der Nullhypothese. Dies ist jedoch nicht fehlerfrei möglich. Es sind prinzipiell zwei Fehlertypenmöglich [Kreng 00]:

Fehler 1. Art: Eine richtige Hypothese wird abgelehnt.

Fehler 2. Art: Eine falsche Hypothese wird angenommen.

Man hat es nun in der Hand, mit welcher Wahrscheinlichkeit beide Fehler auftreten können. Dies geschieht über die Festlegung eines Gebietes, das man den kritischen Bereich K nennt. Die Zufallsvariable T soll bei richtiger Nullhypothese mit einer sehr geringen Wahrscheinlichkeit α in diesen kritischen Bereich K fallen[3]:

$$P(T \in K \mid H_0) = \alpha \quad . \tag{3.47}$$

[3]Es handelt sich bei dieser Wahrscheinlichkeit nicht um eine bedingte Wahrscheinlichkeit, da H_0 kein Ereignis ist. Die Nullhypothese ist entweder wahr oder falsch. Die Schreibweise "$\mid H_0$" bedeutet also "unter der Annahme, dass H_0 wahr ist".

Man nennt α als Signifikanzniveau des Tests. Typische Werte sind $0,01$ bis $0,1$. es ist die Wahrscheinlichkeit für das Eintreten eines Fehlers 1. Art, da man die Nullhypothese ablehnt, wenn sich aufgrund der Stichprobe ein $t \in K$ herausstellt. Ebenso existiert eine Wahrscheinlichkeit β für einen Fehler 2. Art, also für die Annahme einer falschen Nullhypothese:

$$P(T \notin K \mid H_1) = \beta \quad . \tag{3.48}$$

β heißt Operationscharakteristik des Tests. Das Komplement dazu, also $1 - \beta$, ist die Wahrscheinlichkeit dafür, dass wir ein falsches H_0 zurückweisen (auch bezeichnet als die Machtfunktion des Tests) [Litz 01]:

$$P(T \in K \mid H_1) = 1 - \beta \quad . \tag{3.49}$$

Ziel ist es nun, einen kritischen Bereich K so festzulegen, dass die beiden Fehlerwahrscheinlichkeiten, also α und β, möglichst klein werden. Ungünstigerweise sind diese nicht unabhängig voneinander. Eine Verringerung von α resultiert in einer Erhöhung von β, so dass man einen Kompromiss eingehen muss.

3.3 Support Vector Machines (SVM)

Das Prinzip der Klassifikation bzw. Mustererkennung mit *Support Vector Machines* beruht auf dem Finden einer optimal trennenden Hyperebene in einem hochdimensionalen Merkmalsraum - typischerweise mit wesentlich höherer Dimension als der ursprüngliche Merkmalsraum, der auch als Eingaberaum bezeichnet wird[Crist 04].

Man betrachte zuerst die lineare Trennung, um anschließend zu dem komplexeren Fall der nicht linear separierbaren Merkmale zu übergehen.

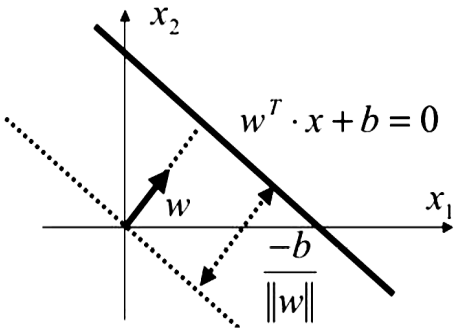


Abbildung 3.13: Hyperebene definiert durch Normalenvektor und Verschiebung

Falls die Daten (Merkmalsausprägungen) linear separierbar sind, so existiert eine trennende Hyperebene (siehe Abbildung 3.13), die durch den Normalenvektor $\mathbf{w}$ und die Verschiebung b wie folgt definiert ist:

$$\chi = \{ \mathbf{x} \mid \omega^T \mathbf{x} + b = 0 \} \quad . \tag{3.50}$$

Dabei steht $\mathbf{w}^T\mathbf{x}$ für das **Skalarprodukt** vom Merkmals- und Normalenvektor. Die Hyperebene separiert die Daten, wie in Abbildung 3.14 gezeigt. Man bezeichne dabei die Punkte in Richtung des Normalenvektors als **positiv** und die Punkte auf der anderen Seite der Hyperebene als **negativ**.

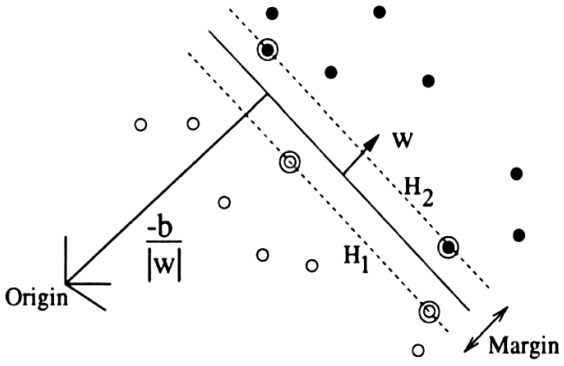


Abbildung 3.14: Trennende Hyperebene mit dem Margin

Um aus verschiedenen möglichen Verläufen der Hyperebene den optimalen zu wählen, betrachte man die minimale Distanz von der Hyperebene zum nächsten positiven und zum nächsten negativen Punkt. Falls man diese als d_+ bzw. d_- bezeichnet, ist der Margin der Hyperebene durch

$$d_+ + d_-$$

gegeben, wobei dieser während des Trainings vom SVM zu maximieren ist[Bishop 98].

Da die Daten linear separierbar sind, existiert eine Hyperebene mit

$$\begin{array}{llll} \mathbf{w}^T\mathbf{x}_i + b \geq +1 & \text{für} & y_i = +1 & , \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 & \text{für} & y_i = -1 & . \end{array} \tag{3.51}$$

Dies lässt sich in einer Gleichung kombinieren:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0 \; \forall \, i \quad . \tag{3.52}$$

Man kann $\mathbf{w}$ so wählen (skalieren), sodass

$$\begin{array}{llll} \mathbf{w}^T\mathbf{x}_i + b = +1 & \text{für ein} & y_i = +1 & , \\ \mathbf{w}^T\mathbf{x}_i + b = -1 & \text{für ein} & y_i = -1 & . \end{array} \tag{3.53}$$

Der Abstand eines Punktes $\mathbf{x}_i$ der Klasse $y_i \in \{+1, -1\}$ zu einer Hyperebene χ lässt sich berechnen als:

$$d(\chi, \mathbf{x}_i) = y_i(\frac{\mathbf{w}}{||\mathbf{w}||}^T\mathbf{x}_i + \frac{b}{||\mathbf{w}||}) \quad . \tag{3.54}$$

Aus gewählter Skalierung der Hyperebene und der Gleichung (3.54) folgt der Margin der Hyperebene als

$$d_+ + d_- = \frac{2}{||\mathbf{w}||} \quad . \tag{3.55}$$

Den Margin (Trennspanne) zu maximieren, heißt $||\mathbf{w}||$ zu minimieren. Dies ist gleich bedeutend mit dem Ziel:

Minimiere $||\mathbf{w}||^2$.

Das Training vom SVM erstreckt sich nun auf das Finden einer Hyperebene, die $||\mathbf{w}||^2$ minimiert, unter der Bedingung, dass die Gleichung (3.52) erfüllt ist. Man führe dafür die Lagrange-Multiplikatoren $\alpha_i \geq 0$ ein und fasse das Optimierungsproblem in der sogenannten Lagrange-Funktion $L(\mathbf{w}, b, \alpha)$ zusammen:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1) \quad . \tag{3.56}$$

$L(\mathbf{w}, b, \alpha)$ zu minimieren, bedeutet das Finden von $\mathbf{w}$, b und α wenn die partiellen Ableitungen der Lagrange-Funktion gleich 0 gesetzt sind:

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad ,$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \quad . \tag{3.57}$$

Wenn man die Ergebnisse aus (3.57) wieder in $L(\mathbf{w}, b, \alpha)$ einsetzt, erhält man nach einigen Umformungen das sogenannte **duale Problem**:

Maximiere

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{3.58}$$

unter den Nebenbedingungen

$$\alpha_i \geq 0 \quad und \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad .$$

Die Trainingsdaten, für die $\alpha_i > 0$ gilt, werden *support vectors* (sv) genannt. Es sind die Punkte, die der Hyperebene am nächsten liegen und ihre Lage bestimmen. Alle anderen Punkte haben keinen Einfluss auf den Verlauf der Hyperebene.

Nach der Lösung des dualen Problems erhält man die α_i, die $W(\alpha)$ maximieren. Damit kann der Normalenvektor $\mathbf{w}$ berechnet werden, der die Trennebene mit dem maximalen Margin (Trennspanne) angibt:

$$\mathbf{w} = \sum_{i=1}^{\#sv} \alpha_i y_i \mathbf{x}_i^{sv} \quad . \tag{3.59}$$

Die Entscheidungsfunktion eines trainierten SVM ist dann wie folgt gegeben:

$$f(\mathbf{x}_{neu}) = sign(\mathbf{w}^T \mathbf{x}_{neu} + b) \quad . \tag{3.60}$$

Häufig können nicht alle Daten separiert werden (siehe Abbildung 3.15). Dies hat einen bestimmten unvermeidlichen Fehler bei der Klassifikation zur Folge, der allerdings während des Trainings durch die Berücksichtigung des Fehler-Gewichts bzw. Fehler-Kosten minimiert werden kann.

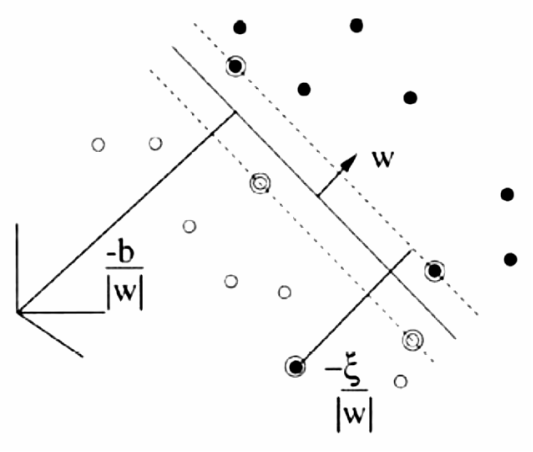


Abbildung 3.15: Nicht vollständig separierbare Daten

Die Ungleichungen aus 3.51 können nun nicht für alle Trainingsdaten erfüllt werden. Stattdessen schreibt man:

$$\begin{array}{lll} \mathbf{w}^T\mathbf{x}_i + b \geq +1 - \xi_i & \text{für} & y_i = +1 \quad , \\ \mathbf{w}^T\mathbf{x}_i + b \geq -1 + \xi_i & \text{für} & y_i = -1 \quad , \end{array} \tag{3.61}$$

wobei

$$\xi_i \geq 0 \quad \forall\, i \quad .$$

Anstelle von $||\mathbf{w}||^2/2$ minimiert man $||\mathbf{w}||^2/2 + C\sum \xi_i$, wobei C das Fehlergewicht angibt.

Nach einigen Rechenschritten ergibt sich das gleiche duale Problem (siehe Gleichung (3.58)) aber mit anderen Nebenbedingungen:

$$0 \leq \alpha_i \leq C \quad \textit{und} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad . \tag{3.62}$$

Als nächstes betrachte man den meist verbreiteten Fall, bei dem die Daten nicht linear separierbar sind. Wie bereits erwähnt, besteht die Grundidee bei SVM in der Transformation der Daten aus dem ursprünglichen Eingaberaum in einen Merkmalsraum größerer Dimension, in dem sich Daten als linear separierbar ergeben.

Man verarbeite die Datenpunkte zuerst mit einer Funktion Φ (*features map*)

$$\begin{array}{rccl} \Phi & : & \mathbb{R}^g & \rightarrow & H \\ & & \mathbf{x} & \mapsto & \Phi(\mathbf{x}) \end{array} \quad , \tag{3.63}$$

wobei H ein Raum ist, in dem ein Skalarprodukt erklärt ist. Anschließend können die Daten im Raum H linear separiert werden, wie bereits beschrieben. Ein Beispiel eines solchen Übergangs zeigt die Abbildung 3.16 [Mark 03].

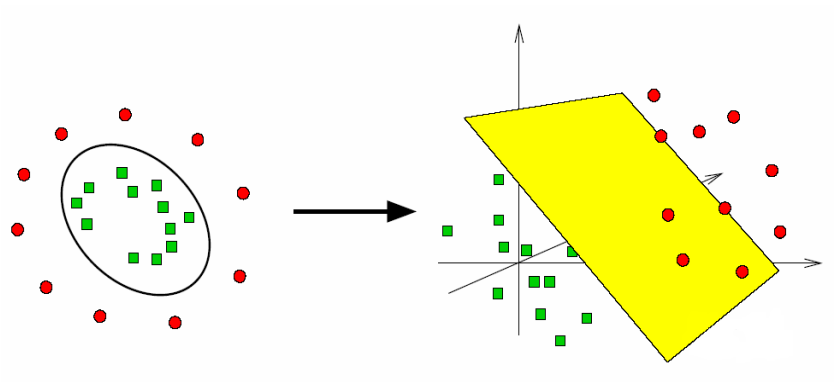


Abbildung 3.16: Lineare Separation von Daten beim $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ Übergang

Das *features map* des dargestellten Beispiels sieht folgendermaßen aus:

$$\begin{array}{rccl} \Phi & : & \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ & & (g_1, g_2) & \mapsto & (z_1, z_2, z_3) = (g_1^2,\ \sqrt{2} g_1 g_2,\ g_2^2) \end{array} \quad . \tag{3.64}$$

Um die Klassifikation in einem hoch-dimensionalen Raum durchzuführen, muss eine trennende Hyperebene konstruiert werden. Infolgedessen müssen die Skalarprodukte der Form

$$\Phi(p)^T \Phi(q)$$

berechnet werden. Dies kann bei hoch dimensionalen H problematisch bzw. sehr rechenaufwendig sein. Als Ausweg benutze man statt des Skalarprodukts eine sogenannte **Kernel-Funktion** K, die im $\mathbb{R}^g$ definiert ist, sich aber wie ein Skalarprodukt in H verhält:

$$K(p, q) = (\Phi(p), \Phi(q)) \quad . \tag{3.65}$$

Zu den meist anwendbaren Kernel-Funktionen gehören der lineare, polynomielle sowie der radial basis Kernel.

Um sicher zu stellen, ob eine Funktion die Kernel-Funktion für den Eingaberaum $\mathbb{R}^g$ ist, verwende man das Theorem von **Mercer**, das besagt:

1. Die Funktion K muss symmetrisch sein: $K(p_i, p_j) = K(p_j, p_i)$.
2. Die Kernel-Matrix K mit $K_{ij} = K(p_i, p_j)$ ist positiv (semi-)definit für alle Trainingsdaten $p_1, p_2, \ldots, p_n$, d. h. $a^T K a = \sum_{i,j} a_i a_j K_{ij} \geq 0 \quad \forall a \in \mathbb{R}^n$.

Man muss nicht wissen, wie der Merkmalsraum H wirklich aussieht, es reicht nur die Kernel-Funktion als ein Mass der Ähnlichkeit[Crist 04]. Häufig weiß man auch nicht, was im Kernel tatsächlich passiert und interessiert sich lediglich für das Ergebnis. Allerdings hat man die geometrische Interpretation der trennenden Hyperebene, wodurch sich die SVMs transparenter als z. B. künstliche Neuronale Netze zeigen.

Ein Beispiel für SVM niedriger und höherer Komplexität aus der medizinischen Röntgenaufnahme zeigt die Abbildung 3.17 [Mark 03].

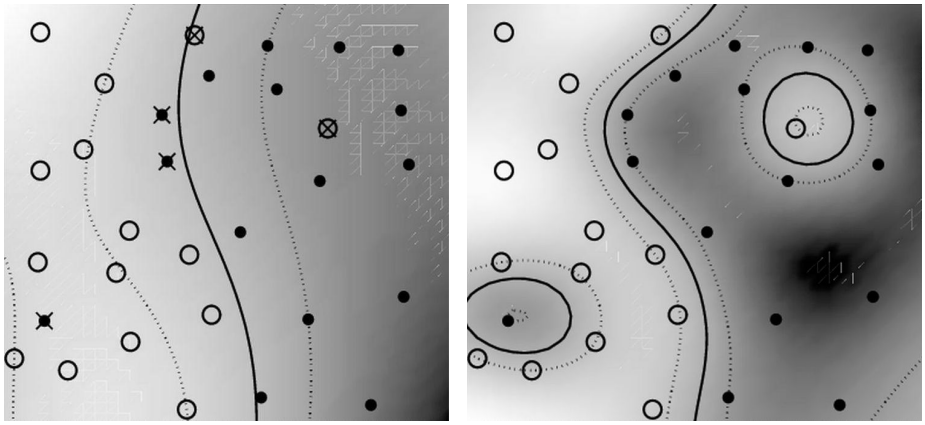


Abbildung 3.17: SVM niedriger (links) und höherer (rechts) Komplexität

3.4 Künstliche Neuronale Netze (KNN)

Ein künstliches neuronales Netz besteht aus stark idealisierten Neuronen. Wie ihr biologisches Vorbild bestehen sie aus drei Komponenten (siehe Abbildung 3.18): dem Zellkörper, den Dendriten und einem Axon. Die Dendriten summieren die Eingabe des Netzes in die Zelle auf, das Axon leitet die Ausgabe der Zelle an die Dendriten nachfolgender Synapsen weiter [Lippe 05].

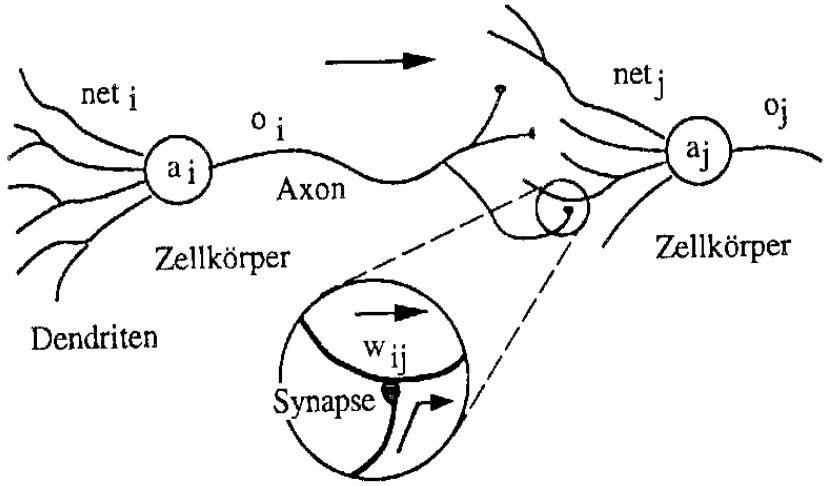


Abbildung 3.18: Vereinfachtes Modell von Neuronen

Die Stärke der Synapsen wird durch einen numerischen Wert, das Verbindungsgewicht, dargestellt. Daher lässt sich die Verbindung zwischen Neuronen als direkte gewichtete Verbindung zwischen den beiden Zellen i und j folgendermaßen darstellen:

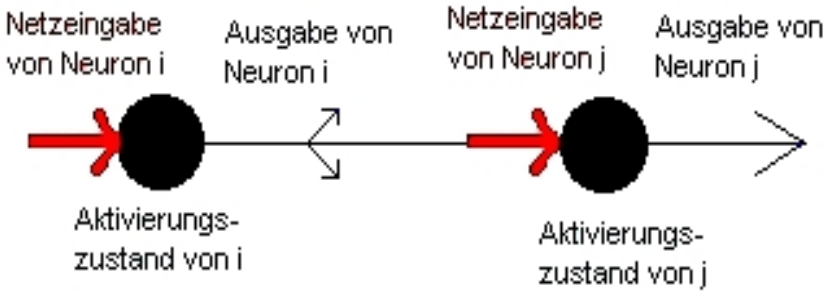


Abbildung 3.19: Verbindung zwischen zwei Neuronen

Formal kann man ein einzelnes Neuron folgendermaßen beschreiben [Bishop 98]:

Definition 3.5 (Künstliches Neuron):
Ein künstliches Neuron ist ein Tupel (x, w, f_a, f_o, o) *bestehend aus einem Eingabevektor* $x = (x_1, \ldots, x_n)$*, einem Gewichtsvektor* $w = (w_1, \ldots, w_n)$*, einer Aktivierungsfunktion* f_a *mit* $f_a : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *und einer Ausgabefunktion* f_o*, für die* $f_o : \mathbb{R} \to \mathbb{R}$ *gilt. Dabei wird durch* $f_o(f_a(x, w)) = o$ *der Ausgabewert des Neurons erzeugt, der an die nachfolgenden Neuronen über die Axonkollaterale weitergeleitet wird.*

Fasst man Aktivierungs- und Ausgabefunktion zu einer **Transfer-Funktion** zusammen, verwendet man häufig die gewichtete Summe:

$$f_a(\vec{x}, \vec{w}) = \sum_{k=1}^{n} x_i w_i \quad . \tag{3.66}$$

Außerdem kann man dem einzelnen Neuron, analog zur biologischen Nervenzelle, einen inneren Aktivierungszustand zuordnen. Der Aktivierungszustand Z hängt vom alten Zustand und der Veränderung der Aktivierungsfunktion ab und kann beispielsweise durch

$$Z^{neu} = Z^{alt} + f_a(\vec{x}, \vec{w}) \tag{3.67}$$

ausgedrückt werden. In der technischen Realisierung wird er unterschiedlich dargestellt. Man unterscheidet (quasi-)kontinuierliche und diskrete Wertebereiche. Bei kontinuierlichen Wertebereichen beschränken die meisten Modelle den Aktivierungszustand auf ein Intervall. Das liegt daran, dass es sich meistens um eine nichtlineare, häufig sigmoide, Aktivierungsfunktion und die Identität als Ausgabefunktion handelt. Dadurch wird die Ausgabe identisch mit der Aktivierung, und der Wertebereich der Aktivierungsfunktion gibt den Wertebereich des Aktivierungszustandes an. Das ursprüngliche Hopfield-Modell und andere Modelle verwenden diskrete Aktivierungszustände. Diese werden dann auch in der Implementierung als binäre Werte gespeichert und verarbeitet.

Damit eine biologische Nervenzelle "feuert", also ein Aktionspotential ausgelöst wird, muss ein bestimmter **Schwellenwert** S überschritten werden. Auch für künstliche Neuronen gibt es eine Schwellenwertfunktion:

$$f_o(\sum_{k=1}^{n} x_i w_i) = \begin{cases} 1: & \text{falls } \sum_{k=1}^{n} x_i w_i \geq S \\ 0: & \text{sonst.} \end{cases} \quad , \tag{3.68}$$

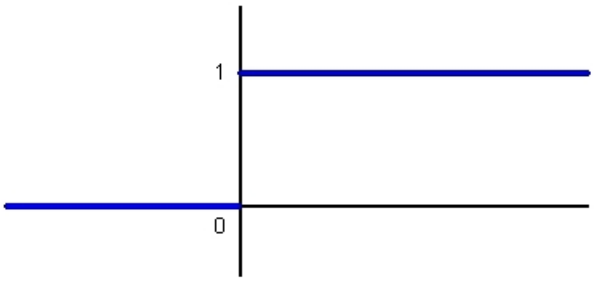


Abbildung 3.20: Schwellenfunktion

Da diese Art der Ausgabefunktion nicht die Intensität der aufeinander folgenden Aktionspotentiale eines biologisches Neurons simuliert, werden lineare Ausgabefunktio-

nen verwendet. Der zeitliche Abstand, in dem die Aktionspotentiale durch die biologische Nervenzelle weitergereicht werden, ist nach unten beschränkt. Daher sollte auch im künstlichen Neuronenmodell eine beschränkte Ausgabefunktion Verwendung finden.

Sinnvoll ist es auch, die Aktivierung beziehungsweise die Ausgabe durch glatte, also differenzierbare Funktionen zu beschreiben. Die folgende Abbildung enthält einige gebräuchliche Aktivierungs- bzw. Ausgabefunktionen, wobei die beiden letzteren am häufigsten verwendet werden.

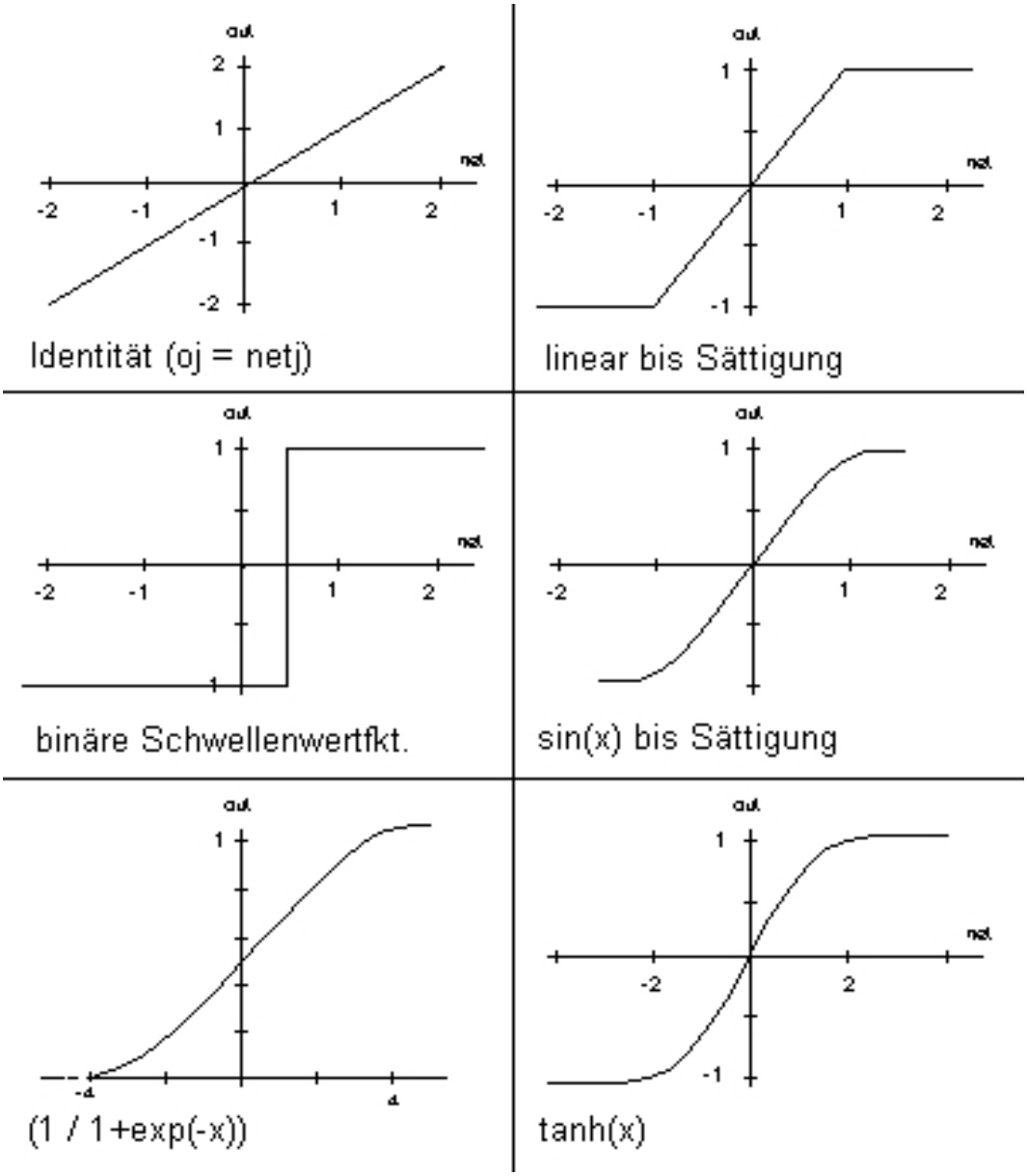


Abbildung 3.21: Gebräuchliche Aktivierungs- bzw. Ausgabefunktionen der KNN

Verbindet man nun mehrere Neuronen miteinander, so erhält man ein neuronales Netz.

Definition 3.6 (Neuronales Netz):
Ein Neuronales Netz ist ein Paar (N, V) mit einer Menge N von Neuronen und einer Menge V von Verbindungen. Es besitzt die Struktur eines gerichteten Graphen, für den die folgenden Einschränkungen und Zusätze gelten:

1. *Die Knoten des Graphen heißen Neuronen.*
2. *Die Kanten heißen Verbindungen.*
3. *Jedes Neuron kann eine beliebige Menge von Verbindungen empfangen, über die es seine Eingabe erhält.*
4. *Jedes Neuron kann genau eine Ausgabe über eine beliebige Menge von Verbindungen aussenden.*
5. *Das Neuronale Netz erhält aus Verbindungen, die der "Außenwelt" entspringen, Eingaben und gibt seine Ausgaben über in der "Außenwelt" endende Verbindungen ab.*

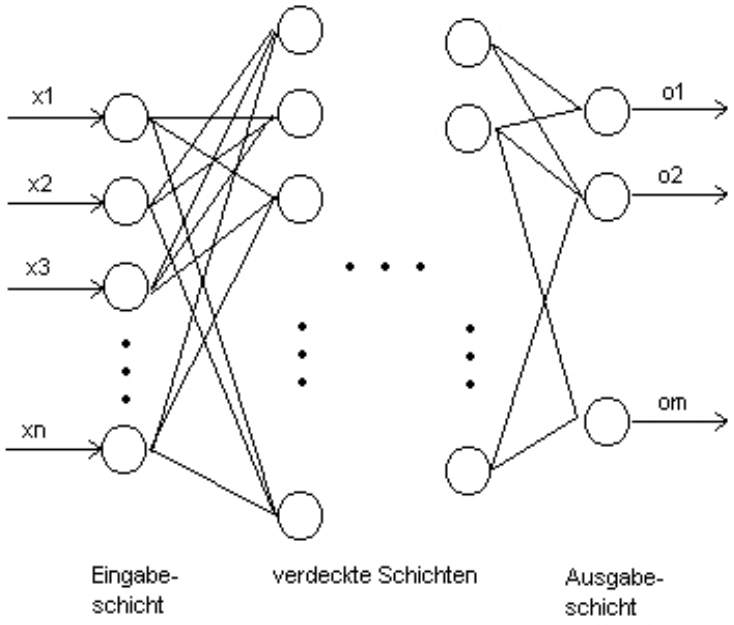


Abbildung 3.22: Neuronales Netz mit verdeckten Schichten

Alle Verbindungen, die von anderen Neuronen zu einem einzelnen Neuron j gehen, ergeben den Eingabevektor x_j von j. Da bei den meisten neuronalen Netzen die Eingabe gewichtet wird, kann man die Verbindungsstruktur (Topologie) in Form einer Matrix beschreiben [Lippe 05]. Zeilen und Spalten identifiziert man mit den Neuronen. In den Kreuzungspunkt schreibt man das Gewicht der Verbindung. In der Schreibweise $W = [w_{i,j}]$ gilt dann:

$w_{i,j} = 0$: keine Verbindung von Neuron i zu Neuron j.

$w_{i,j} < 0$: hemmende Verbindung der Stärke $|w_{i,j}|$.

$w_{i,j} > 0$: anregende Verbindung der Stärke $|w_{i,j}|$.

Neuronale Netze lassen sich gemäß der folgenden Topologien klassifizieren:

1. Netze ohne Rückkopplung (*feedforward*-Netze).
2. Netze mit Rückkopplungen (rekurrente Netze).

Netze mit Rückkopplungen unterteilt man meist folgendermaßen:

1. **Netze mit direkten Rückkopplungen** (*direct feedback*). Die Neuronen haben eine Verbindung von ihrer Ausgabe zurück zur Eingabe und können dadurch ihre eigene Aktivierung verstärken oder abschwächen. Diese Verbindungen bewirken oft, dass Neuronen die Grenzzustände ihrer Aktivierungen annehmen, weil sie sich selbst verstärken oder hemmen.
2. **Netze mit indirekten Rückkopplungen** (*indirect feedback*). Diese Netze besitzen Rückkopplungen von Neuronen höherer Ebenen zu Neuronen niederer Ebenen. Dadurch erreicht man eine Aufmerksamkeitssteuerung auf bestimmte Bereiche von Eingabeneuronen oder auf bestimmte Eingabemerkmale durch das Netz.
3. **Netze mit Rückkopplungen innerhalb einer Schicht** (*lateral feedback*). Netze mit Rückkopplungen innerhalb derselben Schicht werden oft für Aufgaben eingesetzt, bei denen nur ein Neuron einer Gruppe aktiv werden soll. Jedes Neuron hat dann hemmende Verbindungen zu den anderen Neuronen und oft noch eine aktivierende direkte Rückkopplung zu sich selbst. Das Neuron mit der stärksten Aktivierung, der Gewinner, hemmt dann die anderen Neuronen. Daher heißt eine solche Topologie auch *winner-takes-all*-Netzwerk.
4. **Vollständig verbundene Netze**. Vollständig verbundene Netze haben Verbindungen zwischen allen Neuronen. Sie sind insbesondere als Hopfield- Netze bekannt geworden. Bei diesen muss allerdings auch die Verbindungsmatrix symmetrisch sein und die Diagonale darf nur Nullen enthalten.

Netze mit Rückkopplungen werden auch eingesetzt, um Zeitabhängigkeiten bei Daten modellieren zu können. Über die Rückkopplung erhält man als Netzeingabe nicht nur die neuen Daten, sondern auch wieder die bereits verarbeiteten alten Daten.

Ein neuronales Netz "lernt", indem es sich gemäß einer fest vorgegebenen Vorschrift, der Lernregel [Bishop 98], selbst modifiziert. Prinzipiell kann der Lernprozess bestehen aus:

- Entwicklung neuer Verbindungen
- Löschen existierender Verbindungen
- Modifikation der Verbindungsstärke (Veränderung der Gewichte)
- Modifikation des Schwellenwertes
- Modifikation der Aktivierungs- bzw. Ausgabefunktion
- Entwicklung neuer Zellen
- Löschen bestehender Zellen

Von diesen Möglichkeiten wird die dritte, also das Lernen durch Veränderung der Gewichte, am häufigsten verwendet. Erst in letzter Zeit haben Verfahren, die auch eine Veränderung der Topologie beinhalten, an Bedeutung gewonnen [Lippe 05].

Eine weitere Unterscheidungsmöglichkeit besteht in der Art des verwendeten Lernparadigmas. Hier lassen sich drei Arten unterscheiden:

1. **Überwachtes Lernen** (*supervised learning*). Beim überwachten Lernen gibt ein externer Lehrer dem Netz zu jeder Eingabe die korrekte Ausgabe oder die Differenz der tatsächlichen zur korrekten Ausgabe an. Anhand dieser Differenz wird dann das Netz über die Lernregel modifiziert. Diese Technik setzt allerdings voraus, dass Trainingsdaten exisitieren, die aus Paaren von Ein- und Ausgabedaten bestehen.
2. **Bestärkendes Lernen** (*reinforcement learning*). Im Gegensatz zum überwachten Lernen wird dem Netz hier lediglich mitgeteilt, ob seine Ausgabe korrekt oder inkorrekt war. Das Netz erfährt nicht den exakten Wert des Unterschieds.
3. **Unüberwachtes Lernen** (*unsupervised learning*). Hierbei gibt es überhaupt keinen externen Lehrer, daher heißt dieses Lernparadigma auch *self-organized learning*. Das Netz versucht ohne Beeinflussung von außen die präsentierten Daten in Ähnlichkeitsklassen aufzuteilen.

Das Überwachte Lernen ist die meist verbreitete Technik beim Einsatz der künstlichen neuronalen Netze für die datenbasierte Mustererkennung bzw. Klassifikation. Sie umfasst allerdings eine Vielzahl verschiedener Lernverfahren, die aufgrund des großen Umfangs und anderer Problemstellung der vorliegenden Arbeit nicht weiter angegangen werden.

Kapitel 4

Klassifikatoreinsatz und Ergebnisse

Nach der Gewinnung und der vorläufigen Aufbereitung experimenteller Daten kann ein ausgewähltes Klassifikationsmodell mit den gewonnenen empirischen Daten erlernt (trainiert) und getestet werden. Ein trainierter Klassifikator kann anschließend für die Vorhersage der statischen Stabilität der Paketstapel im Rahmen des Palettiervorgangs eingesetzt werden. Dabei ist die Generalisierungsfähigkeit [Fuku 90] des Klassifikators in erster Linie gefragt.

Dieses Kapitel beschreibt die Selektion der Stabilitätsfeatures und die Realisierung eines Bayes'schen-Klassifikators für normalverteilte[1] Merkmale. Zum Vergleich wird der Einsatz der im **PRTools4**©[Duin 04] zur Verfügung gestellten SVM- und KNN-Klassifikatoren vorgestellt. Am Ende werden die Evaluierungsergebnisse mit ihrer Interpretation beschrieben.

4.1 Selektion der Stabilitätsfeatures

Aus dem aufgestellten und implementierten Statikmodell gehen sechs hergeleitete Stabilitätsfeatures hervor, deren Realisierung zum beliebigen Zeitpunkt des Palettiervorgangs die statische Stabilität eines Paketstapels wiedergibt. Wie bereits in 1.3 beschrieben, beziehen sich einige von ihnen [Kippsicherheit, Sicherheitsabstand und Stützbereich, - im Weiteren *Steadiness*, *Safety Margin* und *Support Area*] einzeln auf das jeweilige Paket, wobei der minimale Wert als Aussage für den ganzen Paketstapel betrachtet wird. Die anderen [Relative Packungsdichte, Überbauquotient und Höhe des gemeinsamen Schwerpunktes, - im Weiteren *Pack Density*, *Overbuild Ratio* und *Height of Common Barycenter*] sind im Gegenteil für den gesamten Paketstapel definiert und beinhalten keine Information über die statische Stabilität einzelner Pakete.

[1] Beruhend auf dem zentralen Grenzwertsatz (siehe 3.2.2) und auf der Analyse der experimentellen Daten wird die Normalverteilung für einige Stabilitätsfeatures angenommen.

Die Auswahl eines aussagekräftigen Teilraums der Stabilitätsfeatures erfolgt anhand von **ROC**[2]-Kurven (siehe 4.1.1), der Korrelation sowie des Verlaufs der Dichtefunktion einzelner Stabilitätsfeatures. Dabei steht das PRTools4 als ein mächtiges und umfangreiches Hilfswerkzeug zur Verfügung.

Die Dichtefunktionen aller Stabilitätsfeatures zeigt die Abbildung 4.1.

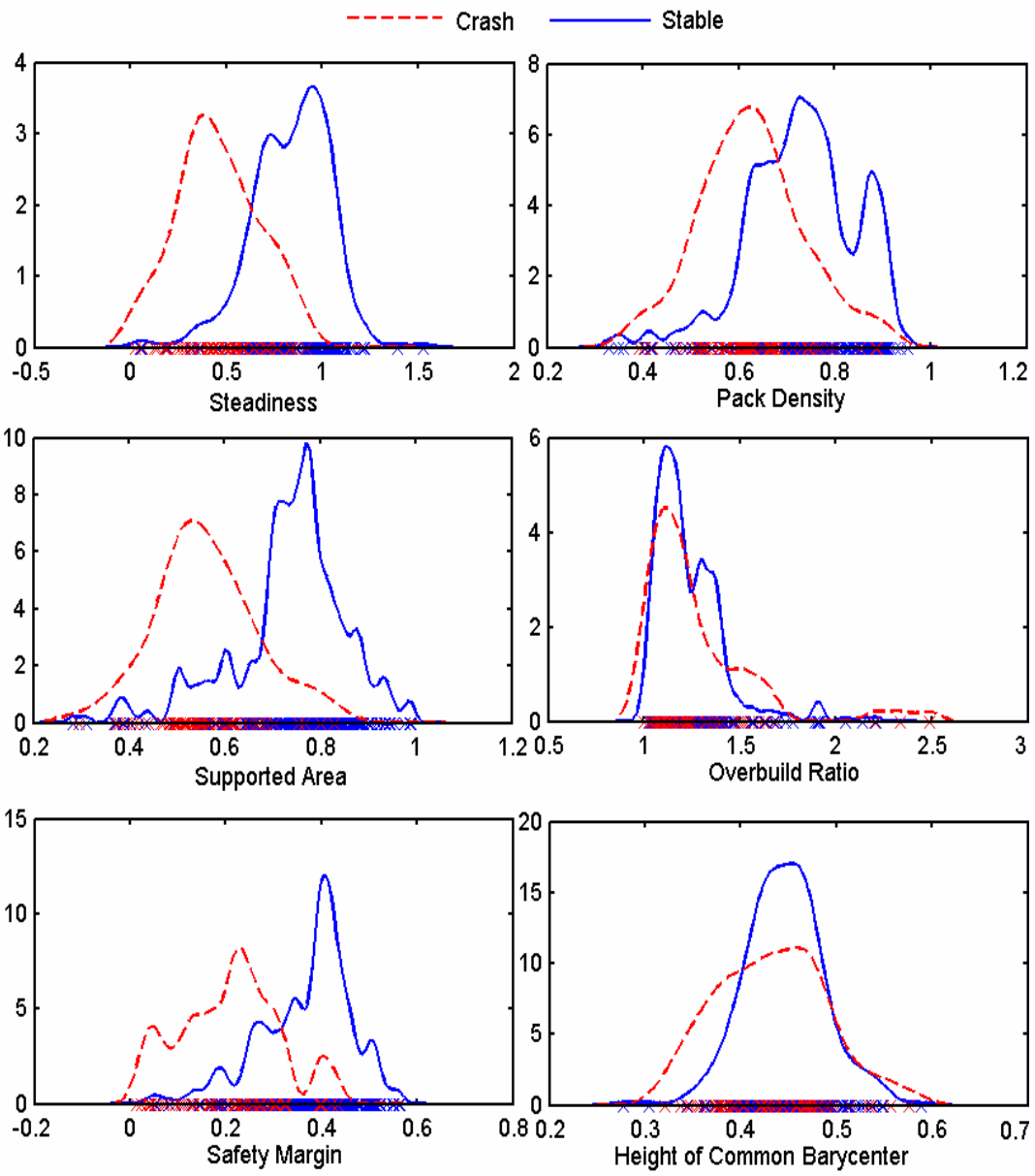


Abbildung 4.1: Dichtefunktionen der Stabilitätsfeatures von Trainingsdaten

Aus Abbildung 4.1 wird ersichtlich, dass die Verteilung der Stabilitätsfeatures *Overbuild Ratio* und *Height of Common Barycenter* beider Paketstapel-Klassen dicht

[2]Receiver Operating Characteristics

übereinander liegt, während die restlichen Stabilitätsfeatures sich in deutlich separierbare Regionen aufteilen. Die folgende Tabelle zeigt die Mittelwerte μ und Standardabweichungen σ der Stabilitätsfeatures beider Klassen:

Stabilitätsfeatures	Mittelwert μ		Standardabweichung σ	
	Crash	*Stable*	*Crash*	*Stable*
Steadiness	0,4562	0,8338	0,2222	0,2054
Safety Margin	0,2061	0,3635	0,1048	0,0979
Support Area	0,5611	0,7342	0,1067	0,1208
Pack Density	0,6288	0,7285	0,1077	0,1140
Overbuild Ration	1,2329	1,2009	0,2402	0,1640
Common Barycenter	0,4337	0,4446	0,0565	0,0435

Tabelle 4.1: Mittelwert und Standardabweichung der Stabilitätsfeatures

4.1.1 Receiver Operating Characteristics

Receiver Operating Characteristics haben ihren Ursprung in der Radartechnologie; sie wurden ursprünglich für den Zweck konzipiert, anhand eines gemessenen Signals X (z.B. Spannung) einen interessierenden Impuls (Radarsignal) vom Hintergrundrauschen zu unterscheiden. Im Zusammenhang mit ROC wird meist die Annahme getroffen, dass sowohl der interessierende Impuls als auch das Rauschen mit gleicher Varianz σ normalverteilt sind. Bezeichne im Folgenden w_1 das Rauschen und w_2 den Impuls, und seien die Verteilungen durch $N(\mu_i, \sigma)$ gegeben, wobei $\mu_2 > \mu_1$ angenommen wird. Rauschen und Impuls werden umso leichter zu unterscheiden sein, je größer die Differenz ihrer Mittelwerte relativ zur Standardabweichung ist. Die von der Entscheidungsgrenze x^* (siehe Abbildung 4.2) unabhängige Kenngröße

$$d' = \frac{|\mu_1 - \mu_2|}{\sigma} \tag{4.1}$$

wird auch **discriminability** genannt. Bei der Klassifikation des Signals können vier verschiedene Ereignisse eintreten:

$X > x^* \,|\, w_2$: *Hit*, Impuls wurde erkannt
$X < x^* \,|\, w_2$: *Miss*, Impuls wurde nicht erkannt
$X < x^* \,|\, w_1$: *Correct Rejection*, Rauschen wurde erkannt
$X > x^* \,|\, w_1$: *False Alarm*, Rauschen wurde als Impuls erkannt.

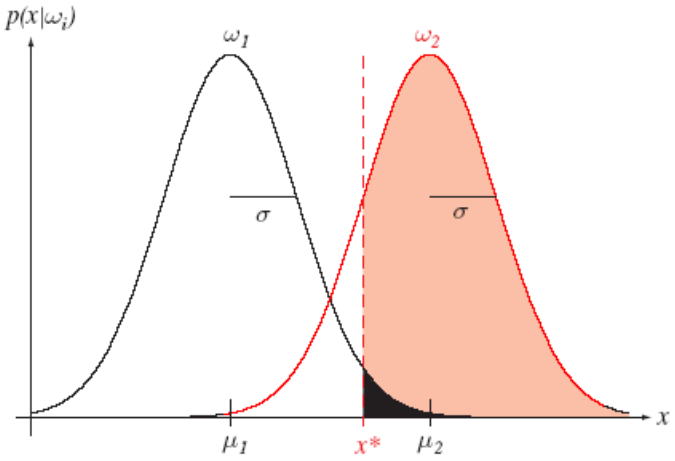


Abbildung 4.2: Wahrscheinlichkeit von *Hit* (rot) und *False Alarm* (schwarz)

Von Bedeutung ist hier insbesondere das Verhältnis der Wahrscheinlichkeiten $P(Hit)$ zu $P(False\ Alarm)$. Wünschenswert ist natürlich eine große *Hit*-Rate bei gleichzeitig möglichst geringer Wahrscheinlichkeit für einen *False Alarm*. Dieser Zusammenhang wird i.A. durch sogenannte ROC-Kurven dargestellt. Jede ROC-Kurve ist durch die *discriminability* des Systems eindeutig festgelegt. Je größer d', desto schneller konvergiert die Kurve (als Funktion von $P(False\,Alarm)$ betrachtet) gegen 1, wie in der Abbildung 4.3 gezeigt. Jeder Punkt auf einer solchen Kurve entspricht dabei einer Entscheidungsgrenze x^*.

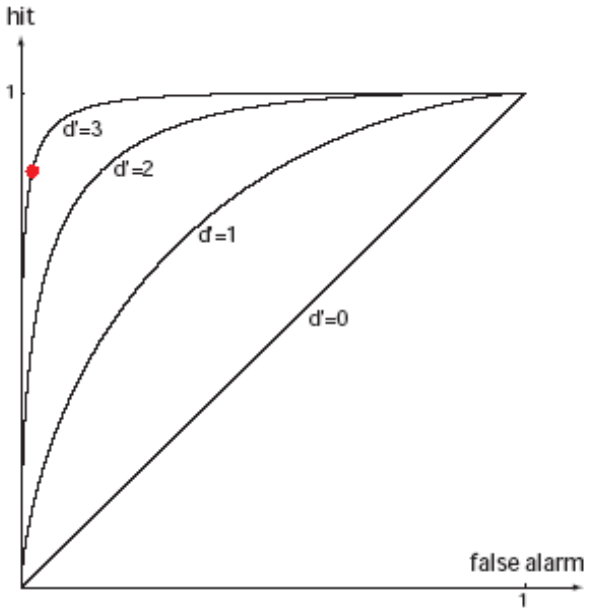


Abbildung 4.3: ROC-Kurven

Bei der Anwendung von ROC-Kurven auf das betrachte Problem treten *Crash-*

Paketstapel als Signal und *Stable*-Paketstapel entsprechend als Rauschen auf, denn eine richtige Vorhersage der *Crash*-Paketstapel ist offensichtlich von größerer Bedeutung. Das korrekte Erkennen der *Stable*-Paketstapel spielt allerdings auch eine Rolle, da alle als *Crash* erkannten Paketstapel nicht akzeptiert und von der Palettiersoftware neu erzeugt (berechnet), bzw. die anderen offline berechneten Paketstapel gewählt werden müssen. Daraus folgt, dass das das Verhältnis der Wahrscheinlichkeiten $P(Miss)$ zu $P(False\ Alarm)$ eher von Interesse ist. Unter *Miss* (im Weiteren *Error*) ist das nicht-Erkennen von den *Crash*-Paketstapeln und unter *False Alarm* das falsche Klassifizieren von den *Stable*-Paketstapeln zu verstehen.

Die Abbildung 4.4 zeigt die ROC-Kurven aller Stabilitätsfeatures einzeln, wobei $P(Error)$ als Funktion von $P(False\ Alarm)$ dargestellt ist. Jeder Punkt der Kurve entspricht einer Position der Entscheidungsgrenze, bei der die entsprechenden Wahrscheinlichkeitswerte beider Fehlklassifikationen in Kauf genommen werden müssen. Es ist offensichtlich, dass je schneller die ROC-Kurve (als Funktion der Wahrscheinlichkeit von *Error*) gegen 0 konvergiert, desto besser die Klassifikation anhand dieses Stabilitätsfeatures durchgeführt werden kann.

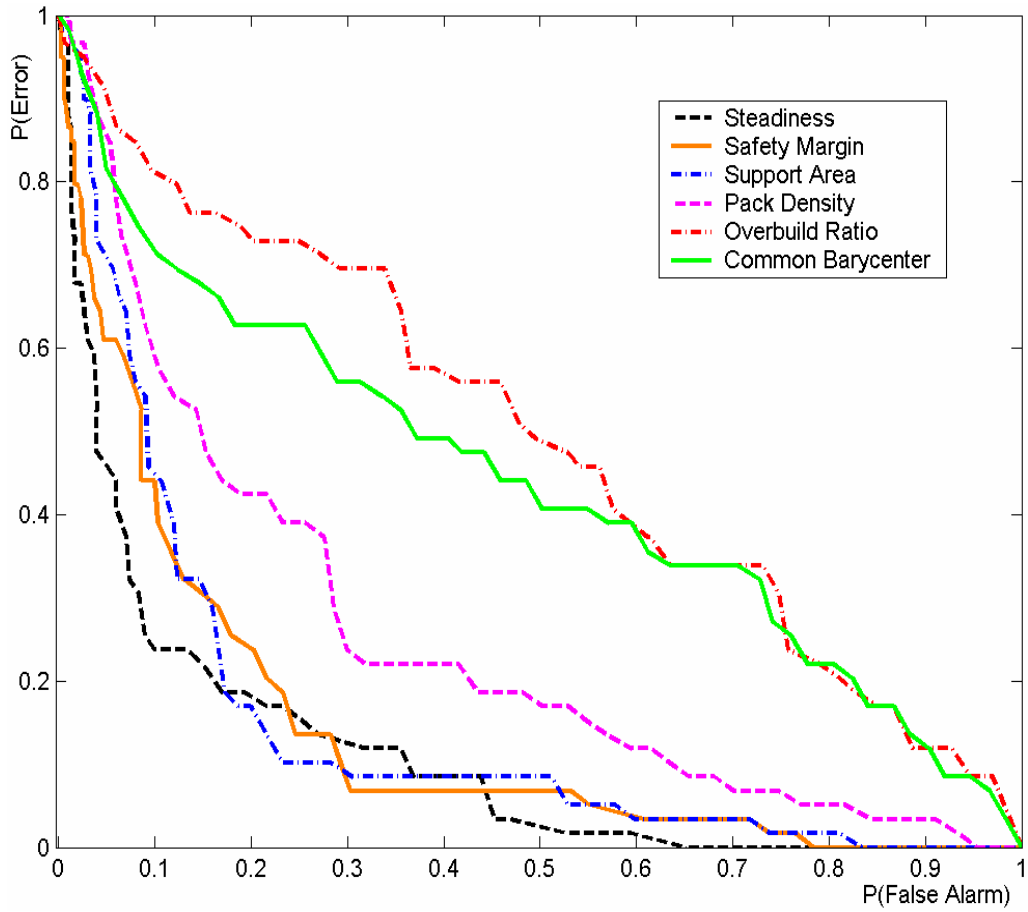


Abbildung 4.4: ROC-Kurven der Stabilitätsfeatures

Die Abbildung 4.4 zeigt, dass die Wahrscheinlichkeit $P(Error)$ als Funktion von $P(False\ Alarm)$ für *Steadiness*, *Safety Margin* und *Support Area* beinah gleich schnell

gegen 0 konvergiert. *Pack Density* zeigt einen etwas ungünstigeren Verlauf der ROC-Kurve, während die ROC-Kurven von *Overbuild Ratio* und *Common Barycenter* etwa dem Wert $d' = 0$ entsprechen, wodurch sie sich für Unterscheidung der Klassen als ungeeignet erweisen.

Wenn nun mehrere Stabilitätsfeatures miteinander kombiniert werden, lässt sich zeigen, dass die ROC-Kurven einen besseren Verlauf gegenüber dem aus der Abbildung 4.4 aufweisen, d.h. schneller gegen 0 konvergieren. Dies bedeutet, dass die Klassifikation anhand von mehreren effizienter als anhand von einzelnen Stabilitätsfeatures erfolgen kann. Die folgende Abbildung zeigt die ROC-Kurven des SVM- (Polynom 3. Grades) bzw. Bayes-Klassifikator bei Verwendung von allen bzw. einigen gewählten Stabilitätsfeatures.

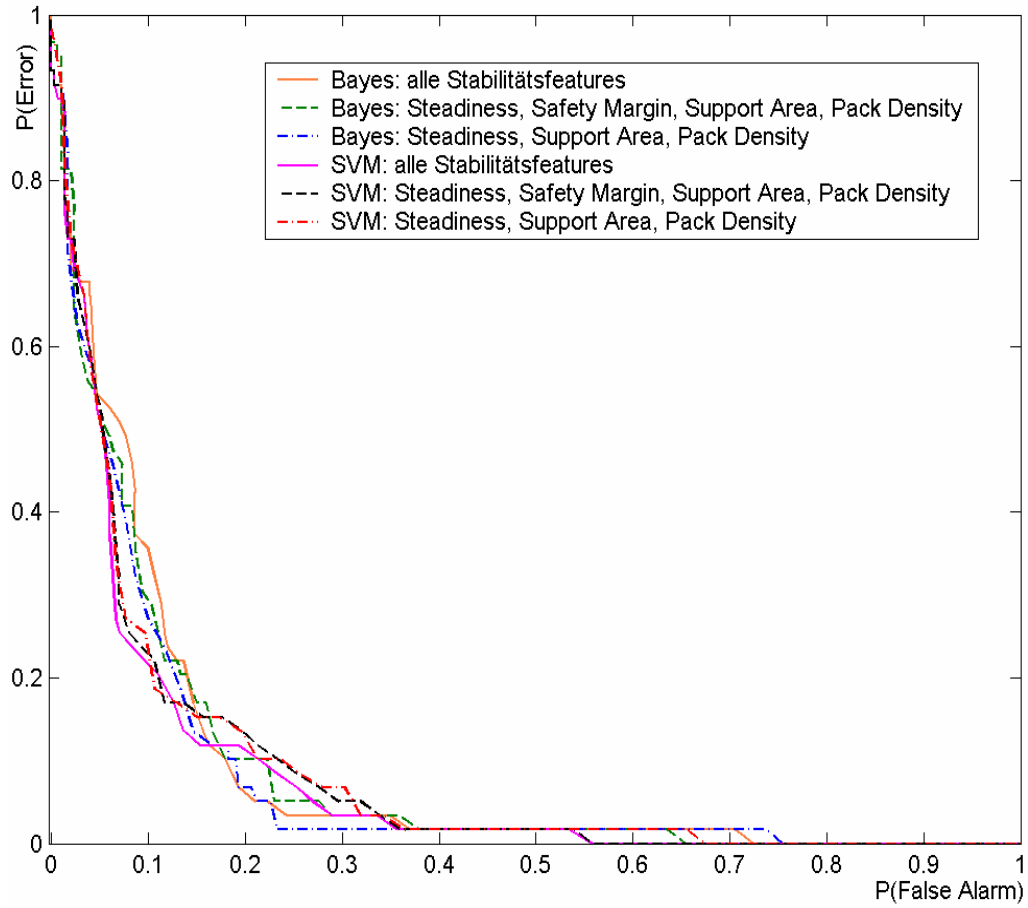


Abbildung 4.5: ROC-Kurven der Kombination von Stabilitätsfeatures

Aus Abbildung 4.5 sieht man, dass die Verwendung aller Stabilitätsfeatures gegenüber z. B. der Kombination *Steadiness*, *Support Area*, und *Pack Density* keine Vorteile bringt, vor allem im interessierenden Arbeitsbereich $P(False\ Alarm) < 0,3$. Die Stabilitätsfeatures *Overbuild Ratio* und *Common Barycenter* sind so schwach separierbar, dass ihre Verwendung die Klassifikation kaum verbessert. Dagegen können die restlichen vier Stabilitätsfeatures als "Kandidat-Features" betrachtet werden.

4.1.2 Korreliertheit der Stabilitätsfeatures

Als nächstes untersuchen wir die Korreliertheit einzelner Stabilitätsfeatures, wobei eine starke Korrelation (als Folge der mathematischen Abhängigkeit) das Klassifizierungsvermögen der Features reduziert und eher unerwünscht ist.

Die folgende Definition der unkorrelierten Elemente eines Zufallsvektors [Litz 01] erlaubt, die Stabilitätsfeatures anhand ihrer Kovarianzmatrix auf die Korrelierheit bzw. Abhängigkeit zu untersuchen.

Definition 4.1 (Unkorreliertheit):
Die Zufallsvariablen $X_1, X_2, \ldots, X_n$ *heißen unkorreliert, wenn alle Kovarianzen* $C_{X_iX_j}$ *verschwinden:* $C_{X_iX_j} = E[(X_i - \mu_{X_i})(X_j - \mu_{X_j})] = 0 \quad \forall i \neq j$.

Um Kovarianzen einzelner Stabilitätsfeatures vergleichen zu können, benötigt man allerdings ein normiertes Maß. Als normiertes Maß für die Unkorrelierheit der Zufallsvariablen X und Y dient der **Korrelationskoeffizient** [Litz 01], welcher folgendermaßen definiert ist:

$$\rho_{xy} = \frac{C_{xy}}{\sigma_x \cdot \sigma_y} \quad . \tag{4.2}$$

Der Korrelationskoeffizient hat die Eigenschaft, dass

$$-1 \leq \rho_{xy} \leq 1$$

ist, wobei $\rho_{xy} = 0$ dem Fall einer absoluten Unkorrelierheit zwischen X und Y, und $|\rho_{xy}| = 1$ dem Fall einer vollständigen Korrelation (lineare Abhängigkeit) entsprechen.

Aus den geschätzten Kovarianzmatrizen beider Klassen lassen sich folgende Korrelationskoeffizienten der "Kandidat-Features" berechnen:

Crash/Stable	*Steadiness*	*Saf. Marg.*	*Supp. Area*	*Pack Dens.*
Steadiness	1,0000/1,0000			
Saf. Marg.	0,7201/0,5616	1,0000/1,0000		
Supp. Area	0,4131/0,4754	0,4981/0,6158	1,0000/1,0000	
Pack Dens.	-0,1813/0,2127	-0,1884/0,2962	0,0910/0,3346	1,0000/1,0000

Tabelle 4.2: Korrelationskoeffizienten der Stabilitätsfeatures beider Klassen

Aus der Tabelle 4.2 wird ersichtlich, dass sich die stärkste Korrelation zwischen *Steadiness* und *Safety Margin* ergibt. *Safety Margin* und *Steadiness* nehmen in ihren

Definitionen den Bezug auf die gleiche physikalische Größe, nämlich auf den kleinsten Abstand zur Kippkante des Paketes (siehe 1.3). Dadurch kann eine starke stochastische Abhängigkeit entstehen, die sich in einer starken Korrelation ausdrückt.

Die Korrelation von *Steadiness* mit *Support Area* ist im Vergleich zu der mit *Safety Margin* zufriedenstellend, während *Pack Density* die beste Unkorreliertheit zu allen drei Stabilitätsfeatures aufweist. Dies lässt sich dadurch erklären, dass *Pack Density* im Unterschied zu den drei übrigen Stabilitätsfeatures keinen Bezug auf die einzelnen Pakete nimmt, vor allem auf ihre Schwerpunkte und Stützkräfte, und eher eine Aussage über die Güte der Ausfüllung des Packraums liefert.

Basierend auf den genannten Kriterien, wie dem Verlauf der Dichtefunktionen mit ihren Mittelwerten und Standardabweichungen, den ROC-Kurven, der Korreliertheit einzelner Stabilitätsfeatures werden *Steadiness*, *Support Area* und *Pack Density* als optimale Kandidat-Features für die Klassifikation der Paketstapel gewählt. Ihre Verteilung im 3D-Featuresraum zeigt die Abbildung 4.6.

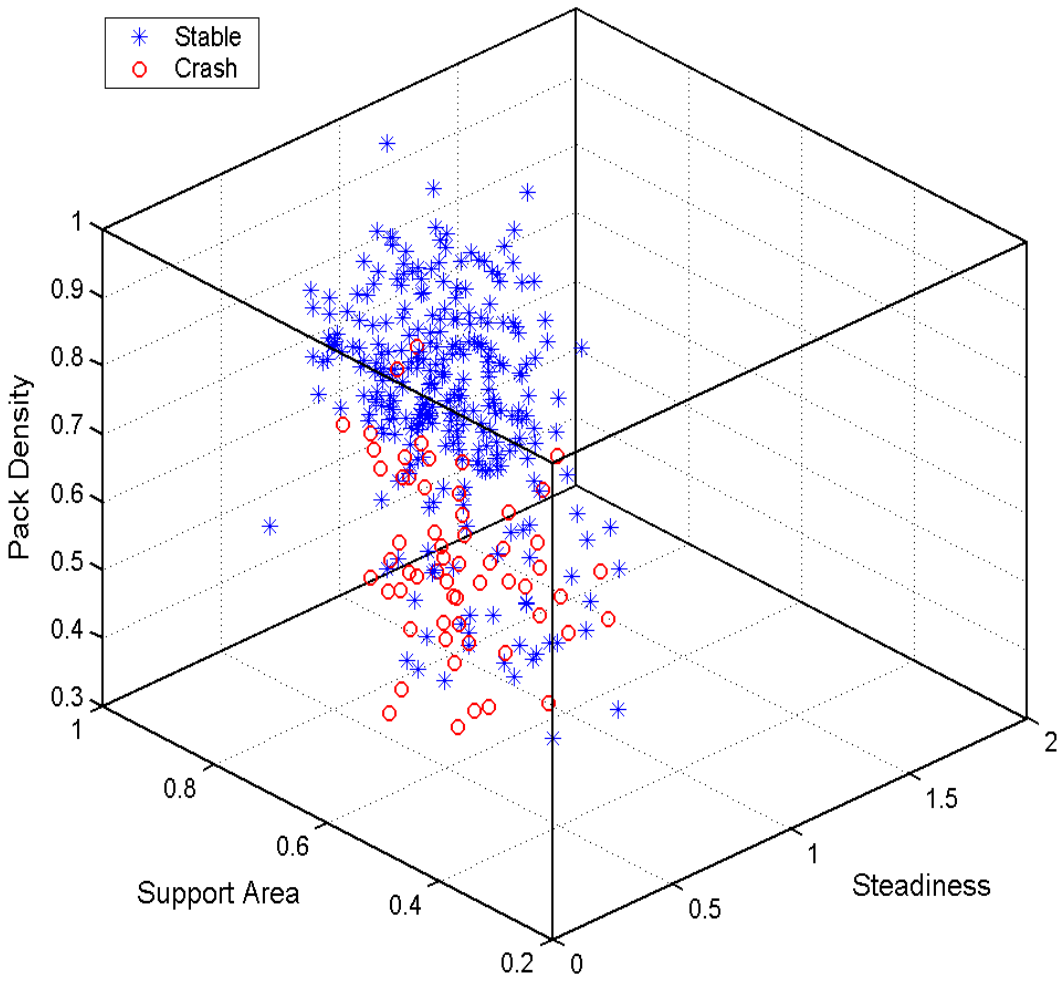


Abbildung 4.6: 3D-Featuresraum für die Klassifikation der Paketstapel

Gegen Verwendung einer noch größeren Anzahl der Dimensionen im Featuresraum (4 oder mehr) sprechen zusätzlich nicht nur die dadurch erschwerte Vorstellungs- bzw. Visualisierungsmöglichkeit sondern vor allem die steigende Komplexität der Entscheidungsgrenze und als Folge die Reduzierung der Generalisierungsfähigkeit des zu verwendenden Klassifikators. Außerdem beeinflusst eine hohe Anzahl von Features erheblich den Rechenaufwand, der in mehreren Fällen nicht zu unterschätzen ist [Fuku 90] und bei dem vorliegenden Problem eine ausschlaggebende Rolle spielt[3].

[3] Nach der Anforderung an das zu entwickelte Verfahren muss das Klassifikationssystem für die Paketstapel in dem laufenden dynamischen System eingesetzt werden können. Daraus folgend darf die Trainings- und Klassifikationslaufzeit die anderen Prozessintervalle, wie etwa Berechnung einer neuen Paketposition oder die Abfuhr eines fertig gebauten Paketstapels, nicht erheblich überschreiten.

4.2 Klassifikatoreinsatz

Wie bereits in 3.1.2 erwähnt, hängt die Güte eines Klassifikationssystems nicht nur von der Auswahl der geeigneten Merkmale sondern auch von dem gewählten Klassifikationsmodell ab, dem ein Klassifikationsverfahren bzw. mathematisches Modell zugrunde liegt.

4.2.1 Vergleich der Klassifikationsmodelle

Wenn man sich lediglich auf die Fehler-Raten der Klassifikation konzentriert, ohne den Rechenaufwand einzelner Klassifikationsverfahren zu berücksichtigen, kann man mittels *cross validation* die Fehler-Raten aller zu unterscheidenden Klassen für verschiedene Klassifikationsmodelle untersuchen.

Die folgende Tabelle zeigt die erzielten *False Alarm*- und *Error*-Raten für quadratischen Bayes-Klassifikator sowie für SVM-Klassifikator mit dem polynomialen und exponentialen Kernel und *Backpropagation* Neuronal-Klassifikator mit MATLAB *default* Initialisierungsparameter (1 *hidden layer*, 5 *units*). Dabei wurden die Klassifikatoren getrennt auf alle bzw. auf die gewählten Kandidat-Stabilitätsfeatures (*Steadiness*, *Support Area*, *Pack Density*) innerhalb des PRTools4[Duin 04] trainiert.

Klassifikationsmodell	alle Stabilitätsfeatures			Kandidat-Features		
	F. Al.	*Error*	*Total*	*F. Al.*	*Error*	*Total*
quadratischer Bayes	0,0806	0,0778	0,1584	0,0625	0,0736	0,1361
SVM (polyn. 2. Grad)	0,0458	0,0917	0,1375	0,0444	0,0917	0,1361
SVM (polyn. 3. Grad)	0,0500	0,0889	0,1389	0,0444	0,0889	0,1333
SVM (exponential)	0,0528	0,0694	0,1222	0,0520	0,0694	0,1214
Backpropagation KNN	0,0639	0,0611	0,1250	0,0500	0,0708	0,1208

Tabelle 4.3: Ergebnisse der *cross validation* von SVM-, KNN- und Bayes-Klassifikator

Die Tabelle 4.3 zeigt, dass der Gesamtfehler der Klassifikation bei allen ausprobierten Klassifikationsmodellen den beinah gleichen Wert um 13% beträgt. Dies führt zur Schlussfolgerung, dass die Trainingsdaten unabhängig vom einzusetzenden Klassifikationsmodell nicht besser separiert werden können.

Bezüglich des Rechenaufwands lässt sich sagen, dass während ein einzelnes Training für den Bayes-Klassifikator in *ms* Bereich lag, nahm dieses für SVM von 3 bis 5 *min* und für KNN von 5 bis 50 *min* in Anspruch. Gemäß der Anforderung an das zu entwickelnde Verfahren zur statischen Stabilitätsanalyse von Paketstapeln muss der Rechenaufwand (Rechenzeit und Speicheraufwand) des Klassifikati-

onssystems bestimmte Grenzen, die durch den Palettierprozess gegeben sind, nicht überschreiten. Dadurch kommt das Bayes'sche Klassifikationsverfahren, als ein parametrisches (schnelles) Verfahren (siehe 3.2), für die Lösung des vorliegenden Problems als best geeigneter Kandidat in Frage.

Abgesehen von Systemressourcen- und Zeitaufwand sprechen für den Einsatz des Bayes'schen Klassifikationsmodells folgende Gründe:

- Die hergeleiteten Stabilitätsfeatures werden von einer Vielzahl unabhängiger physikalischer Größen, anders gesagt der Zufallsvariablen, beeinflusst. Zurückgreifend auf den zentralen Grenzwertsatz[4] tendieren sie zu der Normalverteilung (siehe Abbildung 4.1). Das Streben der gewählten Stabilitätsfeatures gegen ihre Mittelwerte, und zwar für beide Klassen, spricht von vorne herein für die Generalisierungsfähigkeit des Bayes'schen Klassifikationsverfahrens (angewandt auf das vorliegende Problem).
- Außer einer diskreten Antwort (*Stable*/*Crash*) liefert Bayes'sches Klassifikationsverfahren die Wahrscheinlichkeitswerte der Klassenzugehörigkeit, anhand von denen der Einsatzbereich der Klassifikation für statische Stabilitätsanalyse erweitert werden kann.
- Der Kostenfaktor der falschen Klassifikation (Loss-Funktion) kann **nach dem Training** berücksichtigt werden.
- Der asymptotische Verlauf der erzielten Ebtscheidungsgrenzen entspricht am meistens dem physikalischen Sinn der hergeleiteten Stabilitätsfeatures. Die Abbildung 4.7 zeigt den Verlauf der Entscheidungsgrenzen für beispielsweise genommenes Paar der Stabilitätsfeatures beim Einsatz von SVM- (polynomial 5. Grades und exponential), KNN- und Bayes-Klassifikator.

Die Bayes'sche Klassifikation für normalverteilte Merkmale kann in lineare und quadratische Bayes-Klassifikatoren unterteilt werden, wie in 3.2.5 und 3.2.6 beschrieben. Die erfolgreiche Verwendung eines der beiden Typen hängt mit der Struktur der Kovarianzmatrizen aller zu unterscheidenden Klassen zusammen.

Abgesehen von der bedingten Wahrscheinlichkeit beinhaltet die Bayes'sche Entscheidungsregel (siehe 3.2.1) die Terme von a priori Wahrscheinlichkeit und Loss-Funktion (Kosten), deren Werte die Dichtefunktionen beider Klassen neu gewichten und die Entscheidungsgrenze von der a priori wahrscheinlichsten bzw. bei der Fehlentscheidung "teuersten" Klasse weg schieben:

Im Fall eines linearen Klassifikators wird die Entscheidungsgrenze zu dem Mittelwert der entsprechenden Klasse verschoben.

Im Fall eines quadratischen Klassifikators wird die Entscheidungsgrenze um den Mittelwert zusammen bzw. auseinander gezogen.

[4]Gemäß dem zentralen Grenzwertsatz tendiert die Summe mehrerer stochastisch unabhängiger Zufallsvariablen gegen eine Normalverteilung.

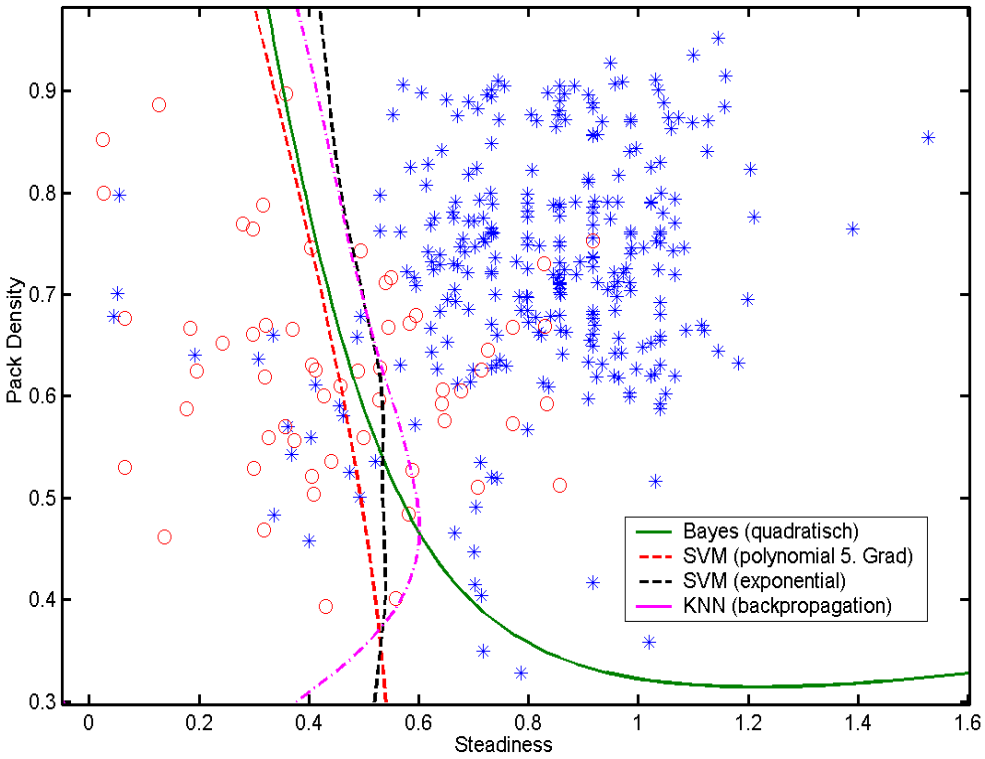


Abbildung 4.7: Entscheidungsgrenzen von SVM-, KNN- und Bayes-Klassifikator

Zu bemerken ist, dass die Verwendung eines linearen Klassifikators bei mehr als zwei Dimensionen des Merkmalsraums gegenüber dem Einsatz des quadratischen Klassifikators benachteiligt wird. Der Grund liegt darin, dass die Erwartung, dass die Kovarianzmatrizen aller Klassen gleich bleiben, mit steigender Anzahl der Merkmalsdimensionen immer geringer wird. Die steigende Dimension des Merkmalsraums erhöht die Komplexität der Entscheidungsgrenze, wodurch der lineare Klassifikator sich als zu "grob" erweisen kann.

4.2.2 Implementation des Bayes-Klassifikators

Beide Typen von Bayes-Klassifikatoren (*linear*, *quadratic*) wurden in Form einer dynamischen Klassenbibliothek implementiert (siehe SW-Komponentendiagramm in Abbildung 4.8), die zwei separate Interfaces (eines zum Training und ein anderes zur Verwendung der Klassifikatoren) zur Verfügung stellt.

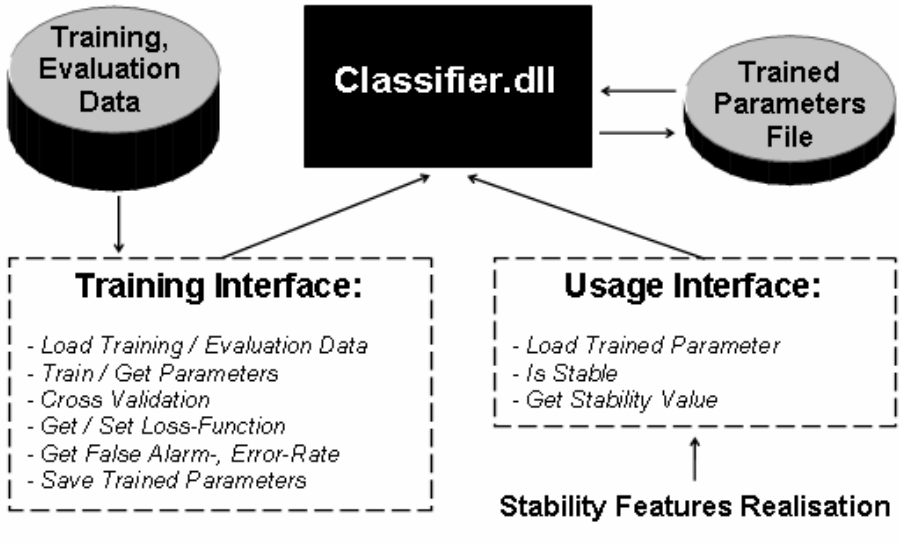


Abbildung 4.8: SW-Komponentendiagramm der Klassifikatoren

Die Trainigs- und Evaluierungsdaten können separat voneinander geladen werden und müssen nicht unbedingt aus der gleichen Menge entstehen. Bei *cross validation* wird allerdings geprüft, dass die Werte des zu klassifizierenden Stabilitätsfeatures-Vektors nicht in dem Satz von Trainingsdaten enthalten sind.

Die Werte der Loss-Funktion für beide Klassen (λ_{ij}) sind frei einstellbar, während die a priori Wahrscheinlichkeiten ($P(w_i)$) sowie die Parameter der Verteilung beider Klassen ($N(\mu_i, \Sigma_i)$) aus den Trainingsdaten geschätzt werden.

Für die Evaluierung der Klassifikatoren stehen Methoden der *cross validation* zur Verfügung, mit deren Hilfe die Beurteilung der Klassifikation anhand der *False Alarm-*/ und *Error*-Raten erfolgen kann.

Ein trainierter Klassifikator (d. h. Haupt- und Zusatzparameter) kann in Form einer binären Datei abgelegt werden. Mit dieser wird bei der Verwendung des *Usage Interface* ein trainierter Klassifikator zur Vorhersage der Stabilität von den Paketstapeln initialisiert.

Für eine benutzerfreundliche Verwendung von *Classifier.dll* zusammen mit der Statikkomponente wurde ein *StaticsTrainer* Programm implementiert, das die für das Training und Evaluierung der Klassifikatoren notwendigen Funktionalitäten in Form einer *GUI*[5] dem Endbenutzer zur Verfügung stellt. Die Abbildung 4.9 zeigt das *GUI*

[5] Graphical User Interface

vom Trainigsrogramm, wobei die Einzelheiten seiner Benutzung hier nicht weiter beschrieben werden (siehe *Manual* im Anhang A).

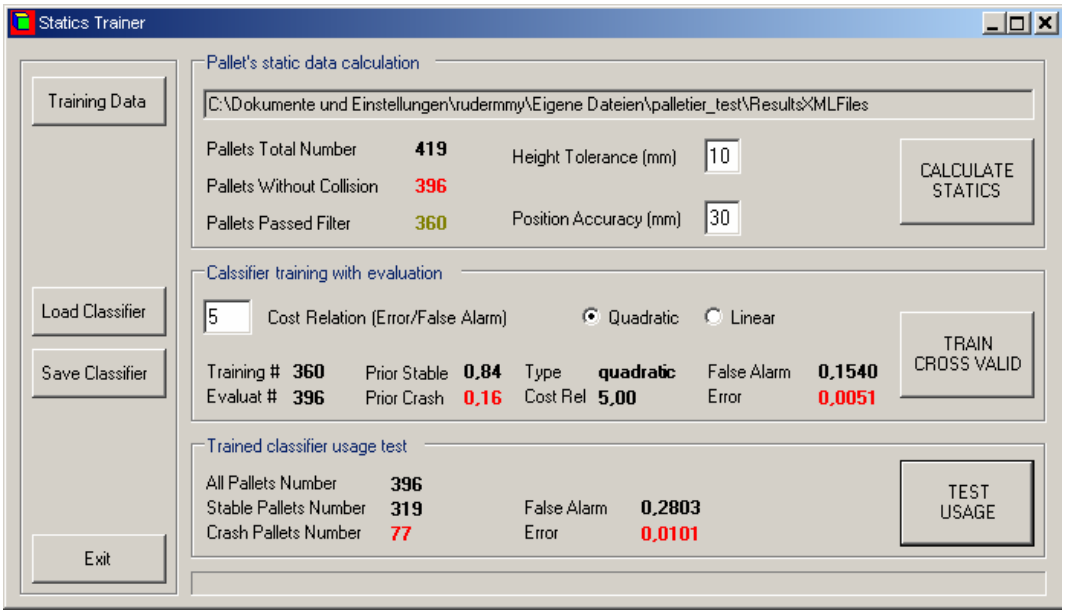


Abbildung 4.9: Trainingsprogramm der Klassifikatoren

Um die Einsatzfähigkeit der trainierten Klassifikatoren besser beurteilen und die Aussagekraft der Stabilitätsfeatures zusätzlich testen zu können, wird im *Statics-Trainer* Programm außer der *cross validation* eine andere Evaluierung (*usage test*) zur Verfügung gestellt. Diese klassifiziert den jeweiligen Paketstapel nach jedem aufgelegten Paket neu und testet auf diese Art den Klassifikator bei allen "Zwischenschritten" der Entstehung eines Paketstapels. Dies entspricht dem realistischen Einsatz der Klassifikatoren für die Vorhersage der statischen Stabilität von den Paketstapeln im Rahmen des Palettiervorgangs, bei dem die Stabilität der Pakete zu jedem sequentiellen Palettierschritt gewährleistet werden soll.

4.3 Ergebnisse

Für das Training der Klassifikatoren wurden die Daten von 360 aufpalettierten Paketstapeln verwendet, die eine Teilmenge der Evaluierungsdaten (396 aufpalettierte Paketstapel) darstellen, so wie in 2.4 beschrieben. Dabei lag die a priori Wahrscheinlichkeit für die *Stable*- und *Crash*-Paketstapel bei $0,84$ bzw. $0,16$.

Mehrere Durchläufe des Trainings mit verschieden eingestellten Werten der Loss-Funktion und anschließender *cross validation* zeigten, dass beim quadratischen Klassifikator die *Error*-Rate (als relevantestes Beurteilungskriterium) sich beinah zu 0 minimieren lässt, wodurch allerdings die *False Alarm*-Rate ansteigt. Der lineare Klassifikator stellte sich bei vorhandenen Trainingsdaten als ungeeignet heraus, unabhängig von den Werten der Loss-Funktion.

Da beim späteren Einsatz der trainierten Klassifikatoren die Auswertung der Stabilitätsfeatures bzw. die Klassifikation eines Paketstapels zu jedem "Zwischenschritt" des Palettiervorgangs erfolgen soll, wurde eine zusätzliche Evaluierung (*usage test*) des Klassifikators vorgenommen, bei der ein trainierter Klassifikator alle[6] vorhandenen Paketstapel schrittweise klassifiziert und die Ergebnisse mit der wahren Klassenzugehörigkeit (Paketstapel eingestürzt oder stehen geblieben) verglichen werden. Dabei reicht mindestens eine falsche "Zwischenklassifikation" eines *Stable*-Paketstapels um die *False Alarm*-Rate zu erhöhen, während bei einem *Crash*-Paketstapel keine richtige Klassifikation von "Zwischenschritten" für die Erhöhung der *Error*-Rate sorgt.

Die folgende Tabelle zeigt die erzielten *False Alarm-/* und *Error*-Raten der *cross validation* sowie des *usage test* beim Einsatz des quadratischen Bayes-Klassifikators mit verschiedenen Werten der Loss-Funktion (Kosten):

Kosten	*cross validation*		*usage test*	
False Alarm/Error	*False Alarm*	*Error*	*False Alarm*	*Error*
1/1	0,1035	0,0177	0,1566	0,0202
1/2	0,1237	0,0101	0,2121	0,0101
1/5	0,1540	0,0051	0,2803	0,0101
1/10	0,1894	0,0051	0,3636	0,0000
1/15	0,2197	0,0000	0,4520	0,0000

Tabelle 4.4: *False Alarm*- und *Error*-Raten vom quadratischen Bayes-Klassifikator

Aus Tabelle 4.4 wird ersichtlich, dass die Ergebnisse bei *cross validation* und *usage*

[6] *Usage test* kann für eine beliebige Datenmenge verwendet werden, die in keinerlei Beziehung zur Trainingsdatenmenge stehen muss.

test voneinander abweichen und die erzielten Raten der falschen Klassifikation mit steigendem Wert der Loss-Funktion unterschiedlich stark anwachsen. Beim Einsatz einer hohen Kostenrelation wird die Entscheidungsgrenze in Richtung der *Stable*-Paketstapel soweit verschoben, dass immer mehr von ihnen bei der Klassifizierung zu *False Alarm* fallen, während der Verzicht auf den Einfluss der Loss-Funktion (Kosten im Verhältnis 1 zu 1) die unerwünschten und zu einem gewissen Grad unzulässigen *Errors* nicht vermeiden lässt.

Error (der nicht vorhergesagte Einsturz des Paketstapels) bedeutet für die Palettieranwendungen den Abbruch des Palettiervorgangs, mit dem entsprechenden Zeit- und Arbeitsaufwand für die Aufräumung der Roboterzelle und ihre Vorbereitung für die Fortsetzung des Arbeitsprozesses. Abgesehen davon gehen die Waren in den eingestürzten Paketen teilweise oder vollständig verloren, wodurch weitere Kosten verursacht werden.

False Alarm (Fehlklassifikation eines Paketstapels als *Crash*) ist im Gegenteil zu *Error* nicht so kritisch, hat allerdings das Verwerfen des analysierten Paketstapels zur Folge. Dies verursacht in Abhängigkeit von der Gestaltung und der Funktionalität der Palettiersoftware den zusätzlichen Rechenaufwand und mögliche Zeitverzögerungen im Arbeitsprozess oder eine notwendige Auswahl eines anderen Paketstapels mit kleinerer Packungsdichte beim offline Einsatz.

Bei allen möglichen Klassifikationsverfahren ist allerdings zu beachten, dass die Güte eines trainierten Klassifikators, vor allem seine Generalisierungsfähigkeit, von dem verwendeten Satz der Trainingsdaten wesentlich abhängt. So wird der Einsatz des realisierten Klassifikators bei denjenigen Palettieralgorithmen benachteiligt, die mit stark verschiedener a priori Häufigkeit die *Stable*- bzw. *Crash*-Paketstapel erzeugen.

Die Ergebnisse aus Tabelle 4.4 zeigen, dass das realisierte und trainierte Klassifikationssystem sich zur Vorhersage der statischen Stabilität der Paketstapel während des Palettiervorgangs (sowohl offline als auch online) eignet. Dies bietet **eine qualitativ neue Lösung für das Stabilitätsproblem beim gemischten Palettieren** an, welches in keiner bis jetzt realisierten Palettiersoftware berücksichtigt wurde.

Da das entwickelte Klassifikationssystem nicht nur eine *Stable*/*Crash* Aussage, sondern auch die prozentualen Werte der Klassenzugehörigkeit liefert, eignet es sich zusätzlich als ein "Kritiker" oder Optimierungsparameter beim Erlernen bzw. Optimieren verschiedener Palettieralgorithmen.

Zusammenfassung und Ausblick

An dieser Stelle wird das entwickelte Verfahren zur statischen Stabilitätsanalyse des Paketstapels beim Roboterpalettieren kurz zusammengefasst, und es werden einige interessante Erweiterungs- und Verbesserungsmöglichkeiten aufgezeigt.

Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde ein grundsätzlich neues Statikmodell eines beim **gemischten Palettieren** entstehenden Paketstapels entwickelt, aus dem eine Reihe von **Stabilitätsfeatures** hervorgeht. Das Modell beinhaltet diverse Annahmen, die eine korrekte Berechnung der Statik eines Paketstapels in verschiedenem Grade beeinflussen. Die Konsequenzen, die aus getroffenen Annahmen gezogen werden, wurden während der **Palettierexperimente** und bei der Analyse empirischer Daten beobachtet und zum größeren Teil beschrieben. Das aufgestellte Modell mit den hergeleiteten Stabilitätsfeatures wurde in einer performanten SW-Komponente umgesetzt, die sowohl offline als auch online die Stabilitätsfeatures eines Paketstapels im Rahmen des Palettiervorgangs wiedergibt.

Durch die Durchführung zahlreicher Palettierexperimente wurden empirische Daten gewonnen, nach derer Aufbereitung sich einige Stabilitätsfeatures als beinah **normalverteilt** herausstellten, und zwar für beide Klassen von Ereignissen: stabil stehende Paketstapel (*Stable*) und eingestürzte Paketstapel (*Crash*).

Nach einer Untersuchung der Grundlagen und Einsätze von datenbasierten Mustererkennungs- und Klassifikationssystemen wurde die **Bayes'sche Klassifikation** (als best geeignetes Klassifikationsverfahren für das vorliegende Problem) ausführlicher untersucht, und darauf folgend den linearen und quadratischen Bayes-Klassifikatoren implementiert, **trainiert** und anschließend **getestet**.

Insgesamt gelang der erste Durchbruch in der Stabilitätsanalyse beim gemischten Palettieren: Die trainierten Klassifikatoren eignen sich zusammen mit der entwickelten Statikkomponente für die Vorhersage der statischen Stabilität eines Paketstapels im Rahmen des Palettiervorgangs.

Ausblick

Im Rahmen des weiteren Verlaufs des Projektes, aus dem diese Diplomarbeit entstanden ist, ergeben sich mehrere Möglichkeiten zur Weiterentwicklung der Statikkomponente genau wie des gesamten Verfahrens zur Stabilitätsanalyse:

- Eine der größten Schwächen des aufgestellten Modells, ist das Ausschließen von seitlichen Kontaktflächen und daraus folgendes Ignorieren der seitlichen Stützkräften, die die Kräftebilanz wesentlich beeinflussen. Daraus ergibt sich die Möglichkeit, das Statikmodell um die seitlichen Stützkräfte zu erweitern, wobei die Reibungsmechanismen innerhalb des Paketstapels sorgfältig zu untersuchen sind.
- Als nächstes kann das Geometriemodell erweitert werden, so dass nicht nur quaderförmige, parallel zu den X, Y-Achsen des Basis-Koordinatensystems stehende Packeinheiten behandelt werden könnten. Dies fordert allerdings eine andere universelle Darstellung der Pack-Einheiten und ihrer Elemente, sowie komplexere Algorithmen zu Bestimmung der Flächen und dadurch verteilter Kräfte.
- Um die Intelligenz und Anpassungsfähigkeit des Klassifikationssystems zu erhöhen, kann ein Feedback bei dem Palettierprozess vorgenommen werden, wodurch die Trainingsdatenmenge in einem laufenden Prozess durch jeden palettierten Packetstapel erweitert wird und das relativ schnell verlaufende Training nach jedem ausgeführten Palettierauftrag neu stattfindet.

Literaturverzeichnis

[Aczel 75] Aczel J., Daroczy Z.: *On measures of information and their characterization*, Academic Press, 1975

[Anger 04] Angermann A., et. al.: *Matlab - Simulink - Stateflow*, Oldenbourg Verlag, 2004

[Bau 04] Baumann G., et. al.: *Logistische Prozesse*, Bildungsverlag EINS-Gehlen, 2004

[Bisch 91] Bischoff E.: *Stability aspects of pallet loading*, OR Spektrum Vol. 13 (1991), p. 189-197

[Bishop 98] Bishop C.: *Neural Networks and Machine Learning*, Springer Verlag, 1998

[Carpen 85] Carpenter H., Dowsland W.: *Practical Considerations of the Pallet - Loading Problem*, Journal of the Operational Research Society Vol. 36, No.6 (1985), p. 489-497

[Crist 04] Cristianini N., Taylor J. S.: *Kernel Methods for Pattern Classification*, Cambridge, 2004

[DeBerg 97] De Berg M., et. al.: *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 1997

[Duda 01] Duda R., Hart P., Stork D.: *Pattern Classification*, John Wiley and Sons, 2001

[Duin 04] Duin R., et. al.: *PRTools4, a MATLAB Toolbox for Pattern Recognition*, http://www.prtools.org, 2004

[Erlhof 94] Erlhof G., Wagner W.: *Praktische Baustatik*, B.G. Teubner, 1994

[Exeler 88] Exeler H.: *Das homogene Packproblem in der betriebswirtschaftlichen Logistik*, Physica-Verlag Heidelberg, 1988

[Freund 03] Freund E., Stern O.: *Skriptum zur Vorlesung Robotertechnologie I*, Lehrstuhl für Automatisierung und Robotertechnologie, Universität Dortmund, 2003

[Fuku 90] Fukunaga K.: *Introduction to Statistical Pattern Recognition*, Academic Press, 1990

[Graham 72] Graham R.: *An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set*, Information Processing Letters Vol. 1 (1972), p. 132-133

[Hemm 98] Hemminki J., Leipälä T., Nevalainen O.: *On-line packing with boxes of different sizes*, Int. J. Prod. Res., Vol. 36, No. 8 (1998), p. 2225-2245

[Hütte 00] Herausgabe von Czichos H.: *Die Grundlagen der Ingenieurwissenschaft*, Akademischer Verein Hütte e.V., Berlin, Springer Verlag, 2000

[Klein 97] Klein R.: *Algorithmische Geometrie*, Addison-Wesley, 1997

[Knus 02] Knus M.: *Skript der Geometrie-Vorlesung*, Mathematik-Departement, ETH-Zürich, 2002

[Kreng 00] Krengel U.: *Einführung in die Wahrscheinlichkeitstheorie und Statistik*, Braunschweig, 2000

[Kuch 96] Kuchling H.: *Taschenbuch der Physik*, Carl Hanser Verlag, 1996

[Lang 02] Lang H.: *Algorithmen*, Oldenbourg Verlage, 2002, Leseprobe: http://www.iti.fh-flensburg.de/lang/algorithmen/algo.htm, 2005

[Lippe 05] Lippe A.: *Softcomputing*, Elektronische Version: http://wwwmath.unimuenster.de/info/Professoren/Lippe/lehre /skripte/index.html, 2005

[Litz 01] Litz L.: *Wahrscheinlichkeitstheorie für Ingenieure*, Hüthig Verlag, 2001

[Mark 03] Markowetz F.: *Klassifkation mit Support Vector Machines*, Max-Planck-Institut für Molekulare Genetik Berlin, 2003

[Miya 03] Miyazawa F., Wakabayashi Y.: *Parametric on-line algorithms for packing rectangles and boxes*, European Journal of Operational Research Vol. 150 (2003), p. 281-292

[Ott 96] Ottmann T., Widmayer P.: *Algorithmen und Datenstrukturen*, Spektrum Akademischer Verlag, 1996

[Ratc 97] Ratcliff M., Bischoff E.: *Allowing for weight considerations in container loading*, OR Spektrum Vol. 20 (1997), p. 65-71

[Schulze 91] Schulze W.: *Kleine Baustatik*, B.G. Teubner, 1991

[Sixt 96] Sixt M.: *Dreidimensionale Packprobleme*, Europäische Verlag der Wissenschaft, 1996

[Tipler 00] Tipler P.: *Physik*, Spektrum A.V., 2000

[Verf 04] Verfürth R.: *Skriptum zur Vorlesung Numerische Mathematik für Maschinenbauer und Umwelttechniker*, Fakultät für Mathematik, Ruhr-Universität Bochum, 2004

[VDHei 04] Van der Heijden F., et. al.: *Classification, Parameter Estimation and State Estimation*, John Wiley and Sons, 2004

[Zulehn 04] Zulehner W.: *Skriptum zur Vorlesung Numerische Analysis*, Institut für nummerische Mathematik, Johannes Kepler Universität Linz, 2004