Anwendung statischer und dynamischer Verfahren zur Fehlermustererkennung und sicheren Reduktion von Chip-Testplänen

Diplomarbeit, Universität Karlsruhe
Fakultät für Informatik

Maxim Feinleb

28. September 2006

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und ohne unzulässige Hilfsmittel angefertigt habe. Alle verwendeten Quellen sind als solche kenntlich gemacht und im Literaturverzeichnis aufgeführt.

Karlsruhe, den 29. September 2006

Maxim Feinleb

Inhaltsverzeichnis

Zusammenfassung

Diese Arbeit befasst sich mit der Problematik der Testplanung in der Chipherstellungsindustrie. Ein großer Faktor der Gesamtkosten sowie ein häufiger Flaschenhals der Produktion liegt im Testen der hergestellten Schaltkreise. Diese Diplomarbeit zeigt Möglichkeiten auf, die in der Industrie eingesetzten Testpläne in ihrer Laufzeit um mehr als 10 Prozent zu minimieren und somit ca. 4 Prozent der gesamten Herstellungskosten einzusparen.

Die von den Halbleiterherstellern zur Verfügung gestellten Daten werden mit Hilfe von unter anderem aus der Spracherkennung bekannten Algorithmen und Klassifikationsverfahren vorverarbeitet und optimiert. Ein besonderer Augenmerk wird hierbei auf die Modellierung sich zeitlich veränderbarer und nicht direkt beobachtbarer Produktionsprozesse gerichtet, welche durch den Einsatz eines Hidden-Markov-Modells erfasst werden.

Im Rahmen der vorliegenden Diplomarbeit wird gezeigt, dass sich gerade der Einsatz von Fertigungsschwankungen beschreibenden Modellen dazu eignet, um die in der Praxis häufig beobachtete Veranlagung von Schaltkreisen oder ganzen Wafern zu immer wiederkehrenden Fehlern zu erkennen. Weiterhin wird gezeigt, dass gute Testzeitersparnisse auf einem integrierten Schaltkreis auch dann möglich sind, wenn der Zustand, in welchem sich der Herstellungsprozess während dessen Produktion befunden haben mag, unbekannt ist.

Die in dieser Diplomarbeit durchgeführten Untersuchungen basieren auf realen Daten, zur Verfügung gestellt von weltweit agierenden Konzernen, mit Produktionsmengen von bis zu 100 Mio. Chips pro Monat.

Kapitel 1

Überblick

In einem Artikel der Zeitschrift *Electronics*, welcher am 19. April des Jahres 1965 erschien, stellte einer der Mitbegründer des damals jungen Halbleiterunternehmens *Intel* Gordon E. Moore fest, dass durch den technischen Forstschritt bedingt sich die Komplexität integrierter Schaltkreise etwa alle 12 Monate verdoppelt. Im Jahre 1975 korrigierte Moore seine Aussage auf 24 Monate, wohingegen heute üblicherweise ein Zeitraum von 18 Monaten als Verdopplungsschritt angenommen wird. Zusammen mit der steigenden Komplexität integrierter Schaltungen erhöht sich auch die Integrationsdichte, was wiederum zur Folge hat, dass die Prozessgröße unaufhaltsam schrumpft. Zum Entstehungszeitpunkt dieser Diplomarbeit waren Strukturgrößen bis zur 29,5nm erreichbar.

Mit stetiger und ständiger Verringerung der Fertigungsgröße der Schaltkreise nimmt aber auch der Einfluss minimaler Veränderungen der Produktionsumgebung und der Beschaffenheit der verarbeiteten Silizium Platten auf die Leistung der hergestellten Chips rapide zu. Dieses Phänomen wird in vielen Publikationen und Artikeln erwähnt, unter anderem in [Mal86], [BD88] und [LWMa99]. Wie im Artikel [MB89] dargestellt, liegt das Verhältnis von gefertigten Bauteilen zu intakten Bauteilen meistens weit unter der geforderten bzw. vertraglich vereinbarten Mindestqualität, woraus sich die Notwendigkeit für das Testen und Aussortieren fehlerhafter Schaltkreise ergibt.

Im Jahre 1994 machten die Testkosten laut [MSV94] rund 30% der gesamten Herstellungskosten eines Chips aus. Im technischen Report [SKCa99] der *Intel Corporation* des Jahres 1999 wird die Roadmap der *Silicon Industry Association* (SIA) zitiert (siehe Abbildung 1.1). Aus der Roadmap geht hervor, dass im kommenden Jahrzehnt die Testkosten integrierter Schaltungen die restlichen Produktionskosten übersteigen können.

Oftmals wird das Chip-Testen auch als der Flaschenhals der Halbleiterwirtschaft betrachtet. Vor allem die zusätzlichen hohen Personalkosten in Europa schränken die Flexibilität der Industrie im Vergleich zu ihrer asiatischen Konkurrenz ein [ITR03]. Aus genau diesem Grund verfolgt diese Diplomarbeit das Ziel ein Werkzeug zu entwickeln, mit dessen Hilfe sich die immer weiter steigenden Testkosten durch den Einsatz aus der Spracherkennung bekannter

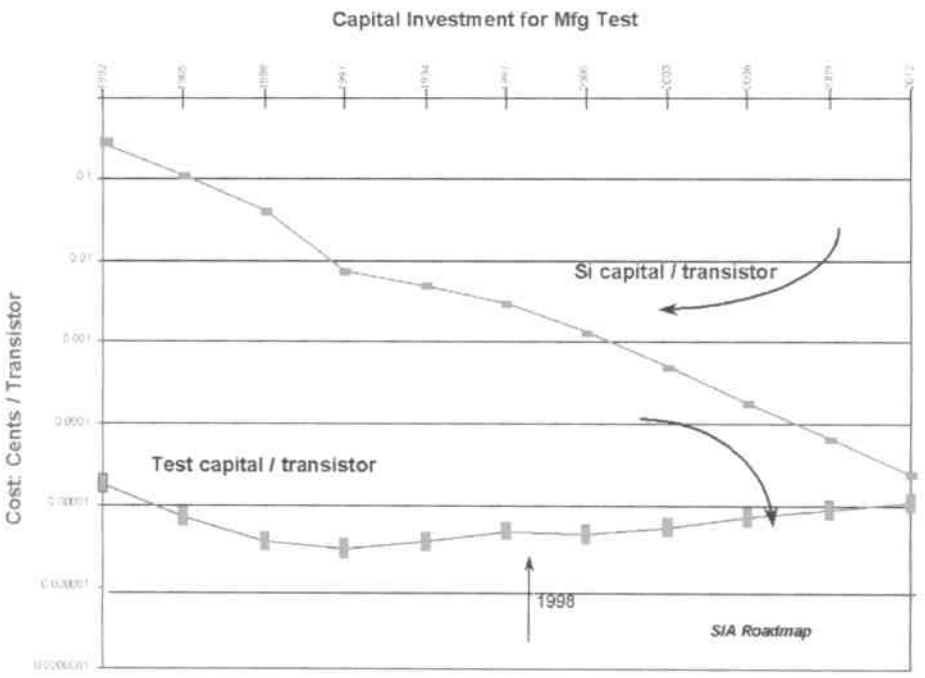


Abbildung 1.1: Produktionskosten im Vergleich zu Testkosten (Quelle: Silicon Industry Association Roadmap, [SKCa99])

Klassifikationsverfahren wieder senken lassen.

Ziele dieser Arbeit sind deshalb:

- Optimierung des gesamten für das Chiptesten benötigten Kostenaufwandes
- Minimierung der Anzahl der auf einem Bauteil ausgeführten Tests
- Bestimmung der effizientesten Methode zur Erfüllung beider vorangehender Ziele

Diese Diplomarbeit geht über die heute gängigen in [MSV90], [MSV94], [SS91a], [SS91b] und [BD88] vorgestellten und weiteren von der Industrien erprobten Verfahren in mehreren Gesichtspunkten hinaus. Die eingesetzten Techniken sind weder auf Produktreihen mit hoher Ausbeute beschränkt, noch ist die Verfahrenskomplexität von der Ausbeute und der Anzahl der Tests mit hoher Fehlerwahrscheinlichkeit abhängig. Die Reihenfolge der Tests muss nicht geändert werden, um so beispielsweise Tests mit höherer Ausfallwahrscheinlichkeit an den Anfang der Testsequenz einzuordnen. Auf Grund des häufigen Auftretens von Produkten mit relativ hoher Ausbeute (zwischen 70% und 95%) werden zur Vergrößerung der Entscheidungsbasis zusätzlich zu Daten von Tests mit Fehlern auch die Messergebnisse fehlerfreier Tests betrachtet. Da über die Ausfallhäufigkeit einzelner Bauteile keine Annahmen *a priori* getroffen werden können, müssen die Entscheidungen über die Auslassbarkeit einzelner Tests für jedes Bauteil zur Laufzeit des Testers innerhalb des Testvorgangs neu getroffen

werden.

Darüber hinaus werden in dieser Diplomarbeit im Gegensatz zu [RvSK05] einzelne Tests zu Testgruppen zusammengefasst. Die Aggregation erfolgt, um zwei in der Praxis beobachteten Phänomenen Rechnung zu tragen. Einerseits bestehen oft Abhängigkeiten zwischen den Tests einer Gruppe und andererseits werden häufig zeitlich ausgedehnte Vorbereitungen der Testapparatur zwischen zwei aufeinander folgenden Gruppen eingelegt, so dass die Auslassung eines Tests innerhalb einer Gruppe zwar zusätzliche Logik aber keine nennenswerte Zeiteinsparung bedeute (siehe auch Abschnitt 3.1.2).

Diese Diplomarbeit entstand in enger Zusammenarbeit mit der Industrie, um die Praxistauglichkeit der Lösung sicherzustellen. Die Entwicklung fand in der Firma *optimiSE GmbH* in Karlsruhe statt, welche seit mehreren Jahren auf dem Gebiet der Chip-Testplanoptimierung tätig ist. Zur Stützung der Allgemeinheit und Wiederverwendbarkeit der vorgestellten Methoden werden die Daten dreier weltweit agierender Konzerne verwendet.

Kapitel 2

Einführung

Dieses Kapitel gibt eine allgemeine Übersicht über die Techniken und den Ablauf des Chip-Testens in der Halbleiterindustrie. Weiterhin werden die Methoden präsentiert, welche den aktuellen Stand der Technik im Bereich der Testzeit-Reduktion und Testplan-Optimierung wiedergeben. Und schließlich, werden die bisher ungenutzten Möglichkeiten zur Optimierung und Minimierung der Testkosten skizziert.

2.1 Chip-Testen

Die Herstellung einer integrierten Schaltung ist ein besonders komplexer Prozess, welcher bereits durch minimale Störungen aus dem Gleichgewicht gebracht werden und damit den Ausfall ganzer Serien eines Produktes bedeuten kann. Zur Minimierung finanzieller Risiken wird deshalb versucht den gesamten Entwicklungsvorgang strengen Kontrollen zu unterziehen. Dies bedeutet häufiges und strenges Überprüfen des sich in Entwicklung befindenden Produktes angefangen bei der Waferherstellung bis hin zum versandfertigen, verpackten Bauteil. Während des gesamten Prozesses fallen Messdaten an, die dazu dienen sollen fehlerhafte Siliziumplatten noch möglichst vor der Bestückung mit Schaltungen oder fehlerhafte Schaltkreise noch vor ihrer Verpackung zu Käferform zu erkennen. In [Fei05] wurde bereits eine Möglichkeit vorgestellt die Testergebnisse einiger Testbausteine, die bereits in einem frühen Entwicklungsstadium auf dem Wafer angebracht werden, einzusetzen, um die Ausbeute des Wafers sowie die Menge auslassbarer Tests auf diesem Wafer zu bestimmen.

Die unterschiedlichen Testphasen während der Herstellung eines bestimmten Produktes sind in der Abbildung 2.1 schemenhaft dargestellt.

Phase 1 Nach der Schmelzung des Silizium-Sandes bei ca. 1400°C wird dieser von Verunreinigungen befreit und zu einem Kristallstab geformt. Dieser besitzt in der Regel eine Reinheit von ca. einem Fremdatom pro 10^{10} Silizium Atomen.

Phase 2 Aus den entstandenen Kristallstäben, welche zurzeit einen Durchmesser von bis zu 300mm erreichen, werden nun Scheiben abgeschnitten. Die

so entstandenen Wafer-Rohlinge bilden die Grundlage für alle Art integrierter Schaltungen. Doch bevor die eigentlichen Schaltkreise auf die Wafer geätzt werden, werden an den späteren Schnittkanten eine Handvoll Testbausteine angebracht, so genannte *Product Control Monitors* (PCMs).

Phase 3 Nach erfolgreicher Vermessung und Überprüfung der PCMs in der Phase 2 bekommen die Wafer ihre eigentliche Funktion. Durch Beschichtung, Lithographie, Ätzung und Dotierung werden die integrierten Schaltkreise (*integrated circuits, ICs*) auf der Waferoberfläche gebildet.

Phase 4 Die fertigen Wafer mit ausgiebig getesteten Chips werden zersägt und die funktionstüchtigen ICs in Käferform verpackt.

Phase 5 Zuletzt werden die Chips zu fertigen Produkten oder Boards montiert, zum letzten Mal als fertiges Komplex auf ihre Funktionalität geprüft und an den Kunden verschickt.

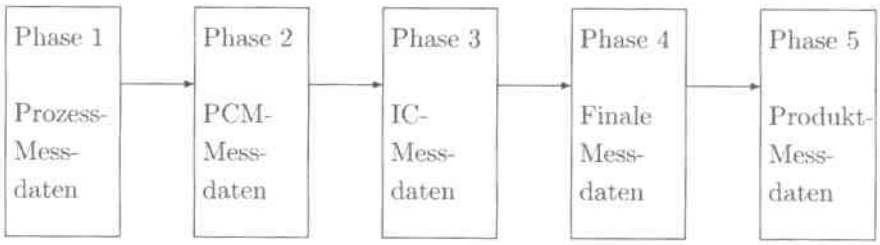


Abbildung 2.1: Chip-Produktion: Prozess in mehreren Phasen

Die im Rahmen dieser Diplomarbeit untersuchten Daten entstammen zwar der dritten Testphase, die im Kapitel 4 aufgeführten Verfahren sind jedoch nicht auf die Daten einer bestimmten Phasen eingeschränkt. Jeder IC durchläuft währen der dritten bzw. vierten Phase im Durchschnitt zwischen einem hundert bis über tausend unterschiedliche Tests. Die Art der durchgeführten Tests richtet sich stark nach der Art der hergestellten Schaltung. Es wird dabei zwischen drei grundsätzlichen Schaltungstypen unterschieden:

Digital Digitale Schaltkreise werden beispielsweise dazu verwendet, um diskrete, binäre Signale zu speichern oder zu verarbeiten.

Analog Die Chips dienen meist zur Verarbeitung beliebiger analoger, kontinuierlicher Signale.

Mixed Signal Diese Art von ICs beinhaltet sowohl analoge als auch digitale Schaltelemente.

Weiterhin gibt es unterschiedliche Testtypen, welche sich in zwei Arten gliedern lassen:

Funktional Funktionale oder digitale Tests setzt man hauptsächlich zur Überprüfung von digitalen Bauelementen ein. Diese Art von Tests liefert ausschließlich eine Bewertung der Funktionsweise eines Elements mit dem binären Wertebereich *bestanden / nicht bestanden* bzw. *go / no go*.

Parametrisch Parametrische oder analoge Messungen werden benötigt, um die Funktionalität von analogen bzw. Mixed-Signal-Bauteilen zu prüfen, sie finden aber auch durchaus auf digitalen Chips Verwendung [Mil98]. Diese Tests bekommen zusätzlich einen festen Wertebereich zugewiesen, innerhalb dessen Grenzen ein Testergebnis als bestanden gilt.

Jede Produktfamilie besitzt selbstverständlich mindestens einen Testplan, der speziell für die Produktfamilie und eventuell eine spezielle Revision des Produktes entwickelt wurde. Im Allgemeinen können die Testsequenzen des Testplans sowohl funktionale als auch parametrische Tests beinhalten. Die durchgeführten Untersuchungen beschränken sich stets zur gleichen Zeit auf eine einzige Testphase und somit eine einzelne Testsequenz.

Wie bereits im Kapitel 1 erwähnt, können Testsequenzen weiter in Testgruppen unterteilt werden. Jede Testgruppe besteht dann aus mindestens einem Test, generell aber aus einer Menge von k Tests. Wie im Abschnitt 3.1.2 genauer erklärt, lässt es sich in der Praxis nicht ausschließen, dass die Tests einer Gruppe auf einander aufbauen, im Sinne, dass ein Test t_j nur dann ausgeführt werden kann, wenn vorher der Test t_i mit $i < j$ ausgeführt wurde.

2.2 Stand der Technik

Digitale Schaltkreise

Statische und dynamische Testsequenz Optimierung

Die meiste existierende Literatur befasst sich mit der statischen bzw. dynamischen Testsequenz Optimierung für digitale Schaltkreise. Es wird stets versucht die Testsequenzlänge zu minimieren ohne dabei die Fehlerabdeckung zu reduzieren. Jeder Test der Testsequenz ist funktional und wird als ein Vektor angesehen. Beide unten aufgeführten Beispielverfahren zeigen Möglichkeiten auf, die Anzahl der Testvektoren zu senken.

Die dynamische Optimierung operiert während der Testgenerierung. Ein Beispiel für einen solchen Ansatz ist [RP99]. Hier wird versucht mit Hilfe eines Fehlersimulators, eines Testgenerators und genetischer Algorithmen eine möglichst kompakte Testvektorenmenge zu kreuzen. Die genetischen Algorithmen übernehmen dabei die Rolle die nur teilspezifizierten Testvektoren derart aufzufüllen, dass zusätzlich zu den Fehlern, auf welche der entsprechende Testvektor ohnehin abzielt, auch noch eine möglichst große Anzahl zusätzlicher Fehler entdeckt wird.

Mit Hilfe der statischen Testsequenzoptimierung wird beispielsweise in [PR02] nach der Testgenerierung eine weitere Möglichkeit für Einsparungen aufgezeigt. Unter Verwendung so genannter *Chronological Order Enumeration*

wird versucht, sukzessive eine Menge von Tests durch einen einzigen Test zu ersetzen. Die Ausführungsreihenfolge wird durch das Verfahren bewahrt, doch bleibt die Schwierigkeit der optimalen Testgenerierung erhalten. Die Fehlerabdeckung der zu ersetzenden Testmenge muss beibehalten werden, was, wie die Autoren zugeben, eines relativ hohen Rechenaufwands bedarf.

Beide Verfahren erzielen zwar eine Reduktion der gesamten Testmenge einer Sequenz, doch bleiben sie gegenüber neuen zum Zeitpunkt der Testgenerierung unbekannten Fehlern blind. Weiterhin bleiben diese Verfahren auf das Testen digitaler Schaltkreise beschränkt, was vor allem im Zeitalter boomender Telekommunikationsmärkte und dem ständig wachsenden Bedarf an Mixed-Signal Chips nicht ausreicht.

Optimierung mit Korrelationsanalyse

Eine weitere in der Industrie eingesetzte Variante basiert auf [BD88]. Hierbei werden im ersten Schritt mit Hilfe einer statistischen Prozess- sowie Schaltkreis-Simulation die Korrelationen zwischen den Messwerten der Tests einer Testsequenz herausgearbeitet. Im zweiten Schritt bestimmt man die Untermenge aller Tests, die zur Vorhersage aller übrigen Tests der Testsequenz ausreichen. Daraufhin werden die augenscheinlich rekonstruierbaren Tests eingespart.

Ein Nachteil dieser Methode liegt in ihrer Einschränkung auf einen relativ rauschfreien Herstellungsprozess. Ein anderer Nachteil ist damit verbunden, dass die Korrelationsbeziehung zweier Tests nicht zwangsläufig auf einen kausalen Ursprung zurückzuführen ist. Aus genau diesem Grund kommt es in der Praxis durchaus häufig vor, dass ein Test eines hoch korrelierten ($\geq$99%) Testpaares einen Fehler liefern kann, wohingegen genau auf dem gleichen Bauteil der Korrelationspartner keine Regung zeigt.

Analoge und Mixed-Signal Chips

Prozessfähigkeitsindex C_{pk}

Ein ebenfalls von der Halbleiterindustrie zurzeit erprobtes Verfahren, welches bisher nur in technischen Reports (z. B. [Swi04]) der Firma *Pintail Technologies, Inc.*, Plano, USA Erwähnung findet, basiert ausschließlich auf der Betrachtung des Prozessfähigkeitsindex C_{pk}. Dieser ist definiert als:

$$C_{pk} = \min\left[\frac{l_u - \mu}{3\sigma}, \frac{\mu - l_l}{3\sigma}\right] \tag{2.1}$$

Dabei bezeichnet l_u die obere und l_l die untere Toleranzgrenze eines Tests, σ repräsentiert seine Standardabweichung und μ steht für dessen Mittelwert. Der Index macht also in Abhängigkeit der Standardabweichung und des Mittelwertes eines Tests von seinen Toleranzgrenzen bestimmte Aussagen darüber, wie stabil sich dieser Test verhält [Cha01]. Das erwähnte Verfahren macht sich diese Eigenschaft des Index zu Nutze, um damit im laufenden

Betrieb zu entscheiden, ob auf den nächsten k Bauteilen ein Test ausgeführt oder eingespart werden kann. Nach k Bauteilen wird der Test wieder für die nächsten l Bauteile ausgeführt, der C_{pk} Wert des Tests aktualisiert und eine neue Entscheidung gefällt.

Vielleicht gerade aufgrund der Einfachheit des Verfahrens besitzt dieses relativ viele Nachteile, ganz abgesehen davon, dass es ausschließlich auf die Betrachtung parametrischer Tests beschränkt ist. Die Entscheidung über die Auslassung einer Messung ausschließlich auf Grund eines Stabilitätsindex erscheint sehr leichtsinnig, wofür gleich mehrere Gründe sprechen:

- Bei der Berechnung des C_{pk} auf Tausenden von Bauteilen erfahren einzelne Werte, die außerhalb der Toleranzgrenzen liegen, keine besondere Behandlung, Gewichtung oder weitere Betrachtung. Daraus folgt, dass eine Messreihe, die zur Identifizierung von vielen Defekten führt, ohne einen möglichen Ersatz ausgelassen werden könnte.
- Die Werte k und l werden mehr oder weniger intuitiv und nicht analytisch bestimmt.
- Die Annahme, dass innerhalb von l Schaltkreisen, auf welchen die Messung voll ausgeführt wird, ein Fehler genau dann auftaucht, wenn dieser auch auf den k ungetesteten Schaltungen auftreten würde, lässt sich bereits mit einem trivialen Experiment widerlegen.

Alle Bautypen

Fehlerverhalten

Für analoge Bauelemente existiert ein in [SS91a] und [SS91b] beschriebenes analytisches Verfahren. Die Basis für das Verfahren liefert die Analyse des Test-Fehlerverhaltens. Diese geschieht durch die QR-Zerlegung[1] einer möglichst großen Menge von Messergebnissen aller Tests der ursprünglichen Testsequenz. Die QR-Faktorisierung liefert einen Satz von Basisvektoren, die genau der gesuchten minimalen Testmenge entsprechen.

Die Schwachstellen dieser Methode liegen zu einem Teil in der Menge der benötigten Messdatensätze, die beachtliche Ausmaße annehmen kann, um die erforderliche statistische Präzision zu erreichen. Zum anderen Teil beschränkt sich dieses Verfahren ausschließlich auf die Betrachtung von fehlerhaften Tests bzw. fehlerhaften Bauteilen. Doch großvolumige, eingefahrene Produkte besitzen nicht selten eine Ausbeute von über 90%, was die Datensammlung und damit verbunden auch die Analyse erheblich erschwert.

Testreihenfolge

Die Technik, welche in [MSV94] aufgezeigt wird, konzentriert sich voll und ganz auf die Umordnung der Testsequenz. Hierbei verfolgt man das Ziel kurze Tests

[1] Die *QR-Zerlegung* oder *QR-Faktorisierung* bezeichnet die Zerlegung einer Matrix A in das Produkt: $A = Q \cdot R$, wobei Q eine *orthogonale* ($QQ^\top = I$) Matrix und R eine obere Dreiecksmatrix ist.

mit hoher Ausfallwahrscheinlichkeit an den Anfang der Testsequenz zu stellen. Die dahinter liegende Idee ist die Einsparung der Ausführungszeit aller übrigen Tests nachdem ein möglichst früher Test einen Fehler meldet. Zur Erstellung der Testsequenz wird *Dijkstras Algorithmus* verwendet. Die Autoren geben die maximale Anzahl möglicher Tests - durch den Algorithmus bedingt - mit 30 an. Zusätzlich zu dieser Einschränkung zeigt auch diese Methode bei Produkten mit hoher Ausbeute wenig Wirkung, da alle fehlerfreien Chips sowieso komplett getestet werden.

Informationsgehalt

Ein allgemeiner Ansatz, welcher sich nicht auf die Analyse der Korrelationen bzw. der Fehlerverhalten beschränkt, wird in [RvSK05] vorgestellt. Mit der Definition eines so genannten *Informationsgehaltes* und einiger weiterer Kriterien wie des C_{pk} oder der Korrelationsanalyse wird hier erreicht, eine Ordnung der Tests einer Sequenz festzustellen. Die Tests niedrigster Ordnung werden aus der Testsequenz entfernt. Der Informationsgehalt basiert auf einer eingehenden Verhaltensanalyse jedes einzelnen Tests und betrachtet zusätzlich zu den als Fehler auch die als *bestanden* markierten Messwerte.

Trotz des Einsatzes dieses Verfahrens in der Industrie besitzt es ebenfalls gewisse Schwachpunkte. Der größte Nachteil ist die fixe bzw. statische Bestimmung der ausgelassenen Tests, denn in der Praxis ist es nicht unüblich, dass die Tests im laufenden Betrieb ihr Verhalten ändern. Diese Änderungen können aufgrund des komplexen Analyseverfahrens des Informationsgehalts nicht im laufenden Betrieb berücksichtigt werden. Ein weiterer Nachteil ist die Außerachtlassung der innerhalb einer Testsequenz bestehenden Abhängigkeiten zwischen den Tests, gerade wenn ein Test den für die Ausführung des nachfolgenden Tests nötigen Zustand des Bauteils oder des Testers aufbaut.

Testphasenübergreifende Optimierung

Die in der Studienarbeit [Fei05] dargestellte Methode verwendet die Messwerte einer vorhergehenden Testphase, um gewisse Aufschlüsse über die möglichen Messergebnisse der darauf folgenden Phase zu gewinnen. Auf diesem Wege ließ sich bereits vor der Bestückung eines Wafers mit integrierten Schaltkreisen dessen Ausbeute und das (Nicht-)Bestehen der Tests der folgenden Testphase mit sehr hoher Genauigkeit vorhersagen. Wie jedoch in der Arbeit bereits erwähnt, besteht eine große Problematik des Verfahrens darin, dass im Normalfall die einzelnen Phasen der Produktion in unterschiedlichen Unternehmen bzw. Standorten eines Unternehmens stattfinden. Diese Aufsplittung führt leider auch dazu, dass die zur Optimierung der Testsequenzen benötigten Daten praktisch nicht mehr rekonstruiert werden können.

2.3 Möglichkeiten der Optimierung

Die stetig steigende Komplexität integrierter Schaltungen sowie fortschreitende Prozessminiaturisierung lassen minimale Prozessfluktuationen immer höheren Einfluss auf das Endprodukt nehmen. Zur Gewährleistung vertraglich geregelter Qualitätsrichtlinien müssen Mechanismen eingeführt werden, welche den gesamten Herstellungsprozess überwachen und unter ständiger Qualitätskontrolle halten [MB89]. Die Qualitätssicherung erfolgt durch die Erhebung für das Endprodukt relevanter Messdaten, angefangen bei der Züchtung des Siliziumkristalls bis hin zur akribischen Überprüfung geforderter Merkmale des fertig montierten Erzeugnisses.

Abhängig von der Organisation des Fertigungsprozesses durchläuft das Produkt mehrere Entwicklungsstadien in unterschiedlichsten Teilen der Welt. Nach dem Abschluss jedes Herstellungsschrittes wird eine neue Testphase durchlaufen (siehe 2.1). Vor allem nach der Anbringung der Logikbausteine auf die Wafer steigt die Anzahl durchgeführter Messungen und mit ihnen auch die zum Testen benötigte Zeit. Diese kann in einer einzigen Phase pro Wafer durchaus im Bereich einer Stunde liegen. Bei der Untersuchung dieses Problems sollte zudem beachtet werden, dass - einfach formuliert - alle durchgeführten Messungen schließlich auf dem gleichen Stück Silizium stattfinden. Unter diesem Gesichtspunkt betrachtet, liegt die Vermutung nah, dass die zu einer Produktionsstufe gehörenden Messwerte sich sowohl untereinander als auch die Messergebnisse späterer Entwicklungsphasen beeinflussen. Die Ausnutzung dieser Erkenntnis sollte zu beträchtlichen Effizienzerhöhungen des gesamten Qualitätssicherungsprozesses führen.

Aus betriebstechnischen bzw. finanziellen Gründen finden die einzelnen Prozessstufen selten in der gleichen Niederlassung eines Unternehmens statt. Viel häufiger ist hingegen der Entwicklungsprozess über mehrere Firmen bzw. Firmenniederlassungen verteilt, und die in den einzelnen Schritten anfallenden Daten lassen sich nur in den seltensten Fällen zusammenführen. Deshalb müssen sich die zur Effizienzsteigerungen verwendeten Informationen zwangsläufig auf einen einzigen Prozessschritt beschränken.

Die verbleibenden Möglichkeiten der Optimierung müssen also

- die während des Testvorgangs einer Fertigungsphase anfallenden Daten analysieren,
- Zusammenhänge zwischen den Messwerten unterschiedlicher Tests auch trotz möglicherweise verrauschter Umgebung erkennen,
- den Beitrag einzelner Tests zur Qualitätssicherung des Produktes in der vorliegenden Phase erfassen,
- die eventuellen Abhängigkeiten folgender Tests von ihren Vorgängern beachten,
- in der Lage sein, sich potentiellen immer wiederkehrenden Schwankungen (wie Testnadelabnutzung) anzupassen,

um für jedes Bauteil eine speziell zugeschnittene Menge von Tests der Testsequenz zu selektieren.

Ein weiteres Detail, welches Beachtung finden sollte, ist die Feststellung, dass unterschiedliche Arten der Fehler sich in ihren Kosten stark unterscheiden können.

Betrachtet man den Fall, dass auf einem intakten Bauteil ein Test bzw. eine Testgruppe ausgeführt wird, obwohl die Ausführung hätte eingespart werden können, so sind die, aufgrund der Fehlentscheidung entstandenen Kosten, verhältnismäßig gering. Sie betragen in diesem Fall die Kosten der unnötig verbrauchten Testerzeit für einen Test oder eine Testgruppe.

Betrachtet man nun den Fall, dass auf einem defekten Bauteil ein Test bzw. eine Testgruppe nicht ausgeführt wird und zudem, genau diese nicht ausgeführte Testgruppe den Fehler des Bauteils erkannt hätte, so sind die Kosten des unterlaufenen Fehlers, im Vergleich zum vorigen Fall, relativ hoch. Das unentdeckte, fehlerhafte Bauteil wird ausgeschnitten, in die Käferform gepackt und entweder durch das Testen in der finalen Phase (siehe 2.1) als defekt erkannt, oder, falls der Klassifikationsfehler erst in der finalen Testphase auftritt, sogar an den Kunden ausgeliefert.

Durch das unnötige Ausschneiden und Verpacken entstehen unnötige, aber bezifferbare Kosten, durch die Auslieferung an den Kunden dagegen noch höher zu bewertende, aber nicht in Zahlen fassbare Imageverluste des Herstellers. Der Kostenunterschied beider vorgestellter Fälle muss bei der Entwicklung des Optimierungsverfahrens unbedingt berücksichtigt werden.

Kapitel 3

Arbeitshypothese

3.1 Problemdefinition

Diese Arbeit präsentiert ein Optimierungsverfahren, mit dessen Hilfe sich die stetig steigenden Testkosten der Halbleiterindustrie, durch den Einsatz aus der Spracherkennung bekannter Klassifikationsverfahren wieder senken lassen. Dabei konzentriert sich die Arbeit auf die Optimierung der Testsequenz einer bestimmten Testphase innerhalb des Fertigungsprozesses eines bestimmten Produktes. Die Art von hergestellten integrierten Schaltkreisen sowie die Art von darauf ausgeführten Tests (siehe Abschnitt 2.1) wird ausdrücklich nicht eingeschränkt, da die allgemeine Einsetzbarkeit des vorgestellten Verfahrens beabsichtigt ist. Dies bedeutet gleichzeitig, dass es *keine* explizite Unterscheidung zwischen dem Testen von ganzen Boards und einzelnen Bauteilen geben wird.

Aufgrund der wahrscheinlichen Verteilung unterschiedlicher Prozessphasen auf unterschiedliche Standorte, darf das Vorhandensein von Messwerten aus vorhergehenden Herstellungsschritten *nicht* vorausgesetzt werden. Insofern ist die herausgearbeitete Methode im Gegensatz zu [Fei05] auf eine einzige, jedoch beliebige Stufe der Herstellung beschränkt. Grundsätzlich muss vor Beginn des Verfahrens sichergestellt sein, dass eine bestimmte Menge an signifikanten Messdaten der zu optimierenden Testsequenz eines Produktes bereits vorliegt.

3.1.1 Annahmen

Zur vollständigen Formulierung der Problemdefinition wurden mehrere Annahmen über den Testablauf getroffen und werden im Folgenden beschrieben.

1. Die zum Testen aller Bauteile eines Produktes verwendete Testsequenz steht in ihrer Reihenfolge fest.
2. Alle Tests der Testsequenz lassen sich zu Gruppen zusammenfassen (siehe auch 3.1.2).
3. Die auf einem Bauteil gemessenen Werte sind zu einem gewissen Maß voneinander abhängig. Dadurch soll es mit Hilfe aus der Spracherkennung

bekannter Algorithmen möglich sein, anhand frühzeitig vorhandener Messergebnisse bestimmte Voraussagen über den weiteren Verlauf der Messung zu treffen.

4. Ein häufig in der Praxis beobachtetes Phänomen führt zu der Annahme, dass unter gleichen Fertigungsbedingungen hergestellte Wafer gewisse Ähnlichkeiten wie bestimmter Fehlermuster oder Anfälligkeiten für Ausfälle aufweisen. So existieren zahlreiche Berichte von Testingenieuren unterschiedlicher Chiphersteller über gewisse Fehlermuster, wie „Schmetterling" oder „Affe" auf Wafern bestimmter Produktionszustände (beispielsweise der Montagsproduktion oder der Abendschicht).

5. Die Fehlentscheidung zur unnötigen Ausführung eines Tests bzw. einer Testgruppe ist um den Faktor hundert *billiger*, als die fehlerhafte Klassifikation eines Tests bzw. einer Gruppe als einsparbar. Die Kosten für eine Testausführung K_T sollen deshalb bei 0.0001 EUR liegen, die Kosten für ein nicht rechtzeitig erkanntes, schlechtes Bauteil (*Escape*) K_E sollen hingegen 0.01 EUR betragen[1]. Weitere Details finden sich unter Abschnitt 2.3.

3.1.2 Gründe für Gruppen

Ein in vielen Publikationen ([Swi04], [MSV94], [RvSK05]) häufig unberücksichtigter Aspekt, ist die Tatsache, dass in der Praxis mehrere Tests einer Sequenz von einander abhängig sein können. Eine derartige Abhängigkeit könnte beispielsweise auftreten, wenn ein Test t_i die Geschwindigkeit messen würde, mit welcher sich ein Kondensator c lädt und ein anderer Test t_j die Geschwindigkeit, mit der sich der gleiche Kondensator c wieder entlädt. Offensichtlich kann der Test t_i nicht ausgelassen werden, wenn der Test t_j als ein wichtiger Bestandteil der Testsequenz angesehen wird.

Es gibt aber auch weitere Gründe, warum eine Menge von Tests zu einer Gruppe zusammengefasst sein könnte, denkbare Beispiele sind:

- Örtliche Lokalität von vermessenen Bauteilen auf einem Board
- Gleiche Art von vermessenen Schaltkreisen
- Gleiche Art der durchgeführten Messung

Hierdurch ergeben sich oft Wartezeiten zwischen den einzelnen Gruppen, die beispielsweise durch die Positionswechsel von Testköpfen bzw. das Aufbauen spezifischer Zustände (wie vorhin: Laden von Kondensatoren) bedingt sein können. Ein einzelner Test einer solchen Gruppe könnte dann Millisekunden dauern, wohingegen die Wartezeit nach der Ausführung der Gruppe im Bereich einer bis mehrerer Sekunden liegen könnte. Würde man in diesem Fall einen Test aus der Gruppe einsparen, brächte das wahrscheinlich keinen echten Testzeitgewinn, zumal die Rechenzeit in welcher die Entscheidung für die

[1] Die Festlegung der Kosten geschah auf Basis echter Werte in Absprache mit den beiden Halbleiterherstellern ATMEL (Heilbronn) und ST (Grenoble, Frankreich)

Einsparung getroffen wird, mitgerechnet werden müsste.

Aus den oben beschrieben Gründen werden in dieser Arbeit Gruppen als kleinste Einheiten einer Testsequenz betrachtet, wobei jede Gruppe aus einer beliebigen Anzahl von Tests bestehen kann, jedoch mindestens einem. Auf diese Weise kann jeder Test, der keine Abhängigkeiten besitzt, von einer eigenen Gruppe repräsentiert werden.

3.2 Problemtransformation

In dieser Arbeit wird gezeigt, wie sich die eingangs dargestellten Engpässe der Halbleiterindustrie mit Hilfe aus der Spracherkennung bekannter Algorithmen optimieren lassen. Zwei im Abschnitt 3.1.1 aufgezählte Annahmen sollen dafür als Schlüssel dienen.

Abhängigkeit von Messergebnissen

Die Annahme, dass die auf einem Bauteil ermittelten Messwerte untereinander unter bestimmten Gesichtspunkten abhängig sind, erlaubt den Einsatz von Klassifikatoren. Die Klassifikatoren werden darauf trainiert, die Zusammenhänge zwischen den einzelnen Messungen auf einem Bauteil eines bestimmten Produktes zu erkennen und darauf aufbauend zu entscheiden, ob eine vorhergesagte Messung ihre Ergebnisse innerhalb oder außerhalb ihrer Toleranzgrenzen haben wird. Selbstverständlich lassen sich auf die gleiche Weise auch ganze Reihen von Messungen (= Testgruppen) vorhersagen.

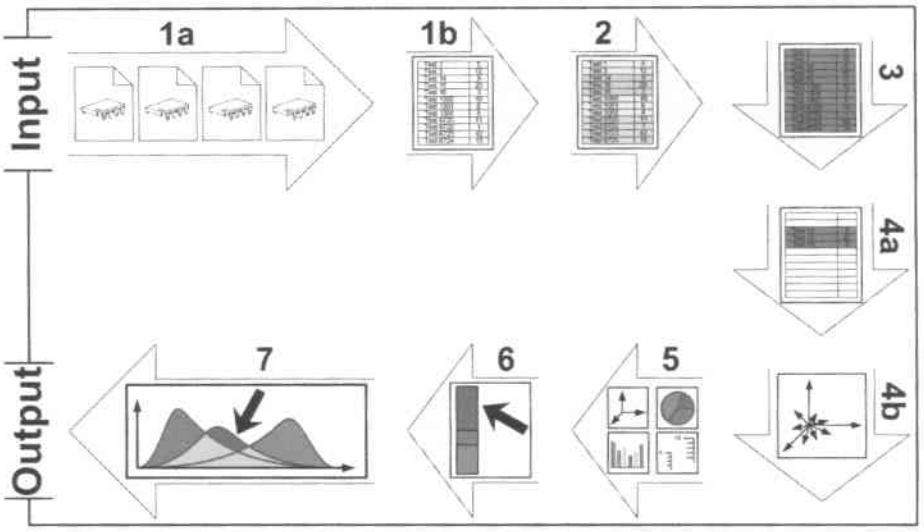
Abhängigkeit vom Herstellungsprozess

Lässt man diese Annahme zu, so ergibt sich die Möglichkeit der Verwendung von zustandsabhängigen Modellen zur Bestimmung von Wahrscheinlichkeitsverteilungen für mögliche Schwankungen des Produktionsprozesses. Als eines der Standardverfahren für eine derartige Fragestellung bieten sich die *Hidden-Markov-Modelle* (HMM) an. Diese werden ebenfalls in der Spracherkennung seit Jahrzehnten zur Modellierung von verborgen Prozessen mit zustandsabhängigen Wahrscheinlichkeitsverteilungen eingesetzt.

In dem vorliegenden Fall wird man für das Hidden-Markov-Modell eine bestimmte Anzahl von möglichen Zuständen definieren, wobei jeder HMM-Zustand einen potenziellen Zustand der Herstellung repräsentiert, beispielsweise:

- Maschinen: Gerade Hochgefahren, Raumtemperatur: Unter Durchschnitt
- Maschinen: Bei regulärer Tätigkeit, Raumtemperatur: Knapp über Durchschnitt
- Maschinen: Nicht ausgelastet, Schichtwechsel

Die Verwendung von Hidden-Markov-Modellen bietet einen weiteren großen Vorteil: Die vom Modell repräsentierten Zustände brauchen nicht bekannt zu



Nr.	Beschreibung
1a,b	Einlesen der Daten
2	Einteilung in Gruppen
3	Analyse des Informationsgehats
4a,b	Sammlung der Messwerte
5	Anwendung der Filter
6	Optimierungsverfahren
7	Klassifikation

Abbildung 3.1: Schematische Darstellung der einzelnen Schritte des Optimierungsverfahrens

sein. Das heißt, dass sich auf diese Weise sämtliche Zustände der Produktion darstellen lassen, ohne dass bestimmte Annahmen über deren Eigenschaften und Parameter gemacht werden müssen.

Der von dieser Arbeit verfolgte Ansatz fordert für jeden Produktionszustand die Existenz eines Klassifikators, der in diesem Zustand die bestmöglichen Klassifikationsergebnisse liefert. Die von jedem Zustand emittierten Beobachtungen entsprechen dabei den auf einem Bauteil vorgefundenen Messergebnissen. Die Übergangswahrscheinlichkeiten sowie die Emissionswahrscheinlichkeiten erlernt das HMM selbständig auf der vorhandenen Lerndatenmenge.

3.3 Vorgehensweise

Die in dieser Arbeit verfolgte Methodik besteht aus mehreren nachfolgend erläuterten und in der Abbildung 3.1 dargestellten Schritten.

3.3.1 Einlesen der Daten

Die ersten beiden in der Abbildung 3.1 dargestellten Schritte widmen sich dem Einlesen der Daten. Im Schritt 1a werden die auf einem Bauteil von der Testerhardware (*bisher*) gemachten Messungen, auf einen Datenträger geschrieben und von der Datenschnittstelle eingelesen. Während des Schrittes 1b werden die eingelesenen Daten in eine Testsequenz und die dazugehörigen Messwerte aufgeteilt.

3.3.2 Gruppenbildung

Im Schritt 2 werden alle Tests der Testsequenz zu Gruppen zusammengefasst, wobei jede Gruppe $k \geq 1$ Tests erhält. Die Zusammenfassung der Tests zu Gruppen ist notwendig, um sicherzustellen, dass alle Abhängigkeiten der Tests untereinander während des gesamten Optimierungsprozesses gewahrt bleiben. Eine Testgruppe stellt dementsprechend das kleinste Element dar, über welches *während der Klassifikation* entschieden wird. Zu weiteren Details siehe auch Abschnitt 4.2.2.

3.3.3 Analyse des Informationsgehalt

Der nächste Schritt der Verarbeitung hängt davon ab, ob es sich bei den eingelesenen Daten um Lerndaten handelt oder, ob es Messdaten von Chips sind, deren Testsequenzen optimiert werden sollen.

Lerndaten: Wurden die Messdaten zu Lernzwecken eingelesen, so muss jetzt bestimmt werden, welche Testgruppen sich als mögliche Kandidaten für die Einsparung eignen. Diese Einteilung erfolgt einmalig für die Dauer des gesamten Experiments. Die Entscheidung wird mit Hilfe der bereits unter Abs. 2.2 vorgestellten Analyse des Informationsgehalts gefällt. Da diese jedoch, wie bereits erwähnt, nur mit einzelnen Tests umgehen kann, wird für die interne Darstellung der Gruppen eine Implementierung verwendet, welche diese als „gewöhnliche" Tests aussehen lässt.

Daten zur Optimierung: Gehören die eingelesenen Daten zu integrierten Schaltkreisen, deren Testsequenzen zur Laufzeit des Testers optimiert werden sollen, so wird davon ausgegangen, dass eine Analyse des Informationsgehalts auf der Testsequenz dieser Bauteile bereits durchgeführt wurde. Sie wird verwendet, um die Gruppen für die weitere Verarbeitung entsprechend zu färben.

Weitere Informationen zur Analyse Informationsgehalts finden sich unter 4.2.3.

3.3.4 Messwertesammlung

Für die weitere Verarbeitung werden, wie im Schritt 4a der Abbildung 3.1 dargestellt, die zur möglichen Einsparung gekennzeichneten Gruppen einzeln betrachtet. Der Schritt 4b stellt die Sammlung aller auf dem untersuchten Bauteil zum jetzigen Zeitpunkt vorhandenen Informationen bzw. Messwerte. Diese Informationen werden für die Klassifikation der betrachteten Gruppe benötigt.

3.3.5 Anwendung der Filter

Die Filter erfüllen innerhalb des gesamten Optimierungsverfahrens mehrere Aufgaben (siehe auch Abschnitt 4.3.1), auch die Entfernung unbedeutender Messungen, wodurch sie auch zu ihrem Namen kommen. Sie sind in der Abbildung im fünften Schritt dargestellt. Im Einzelnen besteht die Hülle und Fülle der von den Filtern erledigten Funktionen aus

- dem Entfernen für die Klassifikation unwichtiger Messungen (4.3.1.1),
- der Anreicherung wichtiger Messungen mit zusätzlichen Informationen (4.3.1.2),
- der Rauschunterdrückung und dem Ausgleich möglicher Prozessschwankungen (4.3.1.4) und
- dem Normierung aller Ergebnisse auf ein einheitliches Intervall (4.3.1.3).

Die genannten Aufgaben werden erledigt, um die Arbeit des Klassifikators zu erleichtern. Anderenfalls müsste der Klassifikationsalgorithmus erst durch das Training lernen, die wichtigen von den weniger wichtigen Messungen zu unterscheiden. Da die Anzahl für das Training zur Verfügung stehender Daten jedoch begrenzt ist, werden die Daten zusätzlich aufbereitet, um den Lerneffekt des Klassifikators zu beschleunigen und die Entscheidungsfindung zu erleichtern.

Die von den Filtern erzeugte Ausgabe hat in vielen Fällen relativ wenige Gemeinsamkeiten mit den ursprünglichen, rohen Daten. Aus diesem Grund werden die Ausgaben der Filter im Folgenden häufig als *bereinigte Messwerte* bezeichnet. Weiterhin kann die Menge M der bereinigten Daten, die zur Entscheidungsfindung an den Klassifikator geleitet wird, als ein *Vektor* eines n-dimensionalen Vektorraums angesehen werden, wobei $n = |M|$.

3.3.6 Optimierung

Im sechsten Schritt der Abbildung 3.1 sieht man den Einsatz unterschiedlicher allgemeiner Optimierungsverfahren. Diese Verfahren werden benötigt, um den Klassifikationsalgorithmen ein weiteres Mal unter die Hände zu greifen, und dieses Mal die Dimensionalität der angereicherten, bereinigten Messwerte wieder zu senken. Dies geschieht ebenfalls, wie im sechsten Schritt, vor allem aufgrund der begrenzten Zahl an Trainingsdaten. Die zur Dimensionsreduzierung eingesetzte Verfahren sind zum Beispiel: Lineare Diskriminanzanalyse, Kohonen Feature Map, Clusterbildung oder Learning Vector Quantization. Details dazu finden sich im Abschnitt 4.3.2.

3.3.7 Klassifikation

Letzter Schritt der Abbildung 3.1 stellt die verschiedenen Klassifikationsverfahren dar. In Abhängigkeit davon, ob es sich um Training oder tatsächliche Vorhersage handelt, verändert sich auch ihre Aufgabenstellung. Die unterschiedlichen eingesetzten Klassifikatoren werden im Abschnitt 4.4 genauer betrachtet.

Training

Während des Trainings müssen die Klassifikatoren einerseits, die Beziehungen zwischen den für eine mögliche Auslassung markierten Kandidaten und den regulären für die ständige Ausführung vorgesehenen Gruppen erlernen. Andererseits, müssen die Klassifikatoren die Intervallgrenzen ermitteln, innerhalb derer sich die ständig gemessenen Werte befinden dürfen, damit die Einsparung der vorhergesagten Gruppen stattfinden kann.

Sollen zudem die zeitlichen Abhängigkeiten und Fluktuationen des Herstellungsprozesses modelliert werden, wird eine zusätzliche, über den Klassifikatoren stehende Instanz implementiert: Das Hidden-Markov-Modell. Das Modell wird ebenfalls mit den Trainingsdaten versorgt und erlernt während ihres Trainings sowohl die Wahrscheinlichkeiten für die Übergänge zwischen unterschiedlichen Prozesszuständen ineinander als auch die Wahrscheinlichkeiten für die Beobachtung einer spezifischen Messung in jedem Zustand.

Vorhersage

Die Vorhersage des Verhaltens der zur möglichen Einsparung markierter Gruppen, wird im Folgenden oft auch als die *Entscheidungsfindung* referenziert. Diese kann zwar auf unterschiedlichste Art und Weise in starker Abhängigkeit vom ausgewählten Klassifikationsalgorithmus erfolgen, es gibt jedoch auch gewisse Gemeinsamkeiten. So ermittelt jeder Klassifikator auf den bereinigten und optimierten Daten die Wahrscheinlichkeit dafür, dass die vorhergesagte Gruppe auf dem aktuell betrachteten Bauteil *keinen* Messwert jenseits der Toleranzgrenzen messen wird. Die errechnete Wahrscheinlichkeit wird mit einem vorher antrainierten Schwellenwert (siehe dazu auch 4.4.5) verglichen und ergibt dadurch die Entscheidung. Diese wird anschließend an die aufrufende Instanz (zum Beispiel: Tester) weitergeleitet.

Wird zusätzlich ein Hidden-Markov-Modell verwendet, so muss dieses zuerst, den zu den überreichten Daten passenden Zustand bestimmen. Dadurch wird auch entschieden welcher Klassifikationsalgorithmus über die vorliegende Daten zu entscheiden hat.

Es besteht außerdem die Möglichkeit, die neu hinzukommenden Messergebnisse aller zur Ausführung bestimmter und vorhergesagter Gruppen der Lerndatenbank hinzuzufügen, um somit die Aktualität der Trainingsdaten zu bewahren. Das Sammeln von Stichproben zur Validierung gemachter Vorhersagen durch die Ausführung kompletter Testsequenz auf jedem k-ten Bauteil ist ebenfalls denkbar.

Kapitel 4

Eingesetzte Verfahren

In diesem Kapitel wird zuerst eine Aufteilung in Stufen, der im Abschnitt 3.3 bereits kurz vorgestellter Schritte des Optimierungsverfahrens, eingeführt. Anschließend werden die in jeder Stufe eingesetzten Verfahren und die Lupe genommen und ausführlich besprochen.

4.1 Strukturierung

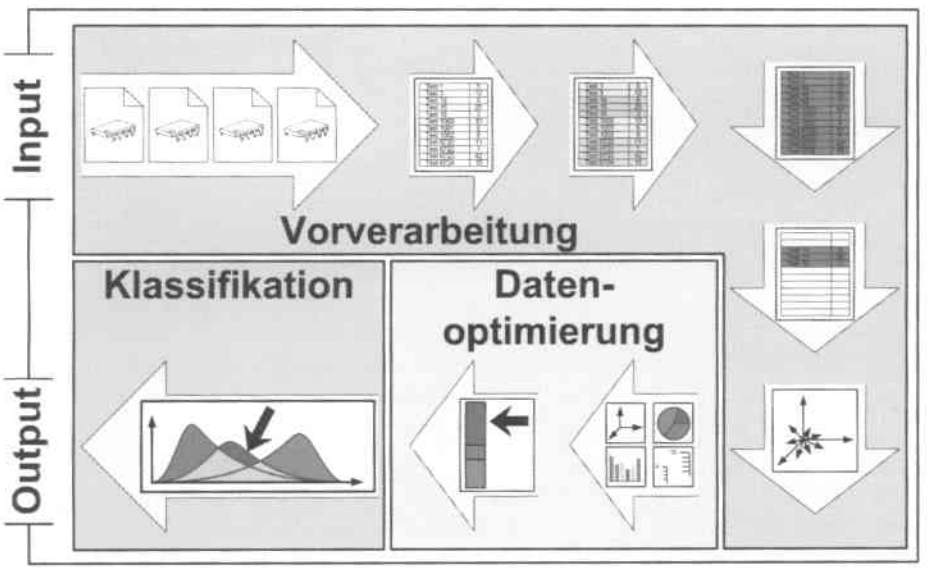


Abbildung 4.1: Gliederung des Optimierungsverfahrens in Stufen

Das in dieser Arbeit vorgestellte Optimierungsverfahren lässt sich, wie auf der Abbildung 4.1 zu sehen ist, in drei Hauptstuffen einteilen:

Vorverarbeitung

Die Hauptaufgabe der Vorverarbeitung (Abschnitt 4.2) ist das zur Verfügung Stellen der eingelesenen Daten für die weiteren Schritte des Optimierungsver-

fahrens. Dafür muss die Einleseschnittstelle in der Lage sein unterschiedliche von den Halbleiterherstellern eingesetzte Datenformate zu erkennen und zu lesen. Die so eingelesenen Testdefinitionen und Testwerte müssen zu Gruppen gebündelt werden. Anschließend werden die Gruppen anhand ihres Informationsgehalts in zwei Kategorien: {zwingend auszuführen} und {eventuell einsparbar} unterteilt. Zum Schluss müssen alle, für das Vorhersagen jeder eventuell einsparbaren Gruppe, vorhandenen Daten der Datenoptimierungsstufe übergeben werden.

Datenoptimierung

Die Datenoptimierung (Abschnitt 4.3) hat das Ziel, die von der Vorverarbeitung überreichten, rohen Messdaten in aufgearbeiteter, „leicht klassifizierbarer" Form den Klassifikatoren bereitzustellen. Die Aufarbeitung erfolgt in mehreren Stadien:

- Befreiung der Daten von unbedeutenden Messungen und Rauschen
- Anreicherung mit Zusatzinformation
- Normierung auf ein einheitliches Intervall
- Zusammenfassung der Daten zu einem Vektor
- Betonung bestimmter Koeffizienten des Vektors durch Transformation (beispielsweise: Clustering oder Lineare Diskriminanzanalyse)

Klassifikation

Die im Abschnitt 4.4 genauer beschriebenen Klassifikatoren erhalten nach der Durchführung der Datenoptimierung bereits bereinigte und optimierte Vektoren. Die eigentlichen Klassifikationsalgorithmen müssen hierbei nicht wissen, welche Bedeutung die klassifizierten Vektoren tragen. Für die Klassifikatoren stellen die Vektoren lediglich die Punkte eines mehrdimensionalen Raumes dar, und es gilt, diese mit einer möglichst geringen Fehlerquote, einer von zwei möglichen Klassen zuzuordnen. Die Zuweisung eines Vektors zu einer Klasse repräsentiert die Entscheidung über die Ausführung bzw. die Einsparung einer Testgruppe auf einem untersuchten Bauteil. Diese Entscheidung wird der Testerhardware, als Ausgabe des gesamten Optimierungsverfahrens mitgeteilt.

4.1.1 Klasseneinteilung

Die zwei weiter oben erwähnten Klassen sollen an dieser Stelle benannt werden, um potenzielle Unklarheiten in folgenden Kapiteln von vornherein auszuschließen. Wie bereits gesehen, bedeutet die Zuweisung eines Vektors zu einer Klasse, die Entscheidung *für* oder *gegen* die Auslassung einer Testgruppe der Testsequenz des gerade analysierten Chips. Abhängig davon, welche semantische Bedeutung eine Klasse trägt, soll auch ihr Name passend gewählt sein.

Klasse C_R: Das im Namen auftauchende $_R$ steht hierbei für *redundant*[1]. Die Zuweisung eines Vektors zu dieser Klasse, stellt also die Einsparung einer Gruppe dar. Indirekt spiegelt diese Klasse auch die Menge aller Bauteile wider, auf welchen die Ausführung einer bestimmten Testgruppe *nicht* notwendig ist.

Klasse C_M: Jede Zuweisung eines Vektors zu dieser Klasse repräsentiert die Ausführung einer Gruppe. Dementsprechend vertritt diese Klasse mittelbar die Menge aller integrierten Schaltkreise, auf welchen die Ausführung einer bestimmten Testgruppe notwendig ist. Der Buchstabe $_M$ im Namen der Klasse steht für *mandatory* und soll die Notwendigkeit der Ausführung zum Ausdruck bringen.

[1] Der Begriff *redundant* wurde deshalb gewählt, weil die zur Klasse C_R klassifizierten Testgruppen keine zusätzliche Information beinhalten, welche über die Information der Klasse C_M zugeordneter Testgruppen hinausgeht.

4.2 Vorverarbeitung

Die Vorverarbeitung ist ein umfassender Prozess, der einen großen Teil zum Erfolg des gesamten Optimierungsverfahrens beiträgt. Während mit Hilfe der im Abschnitt 4.3 beschriebenen, der Vorverarbeitung folgenden Techniken die eingelesenen, rohen Messergebnisse gefiltert und optimiert werden, muss die Vorverarbeitung die dafür nötigen Grundsteine legen. Die hierfür benötigten Schritten wurden bereits im Abschnitt 3.3 umrissen. Im Folgenden sollen sie genauer betrachtet werden.

4.2.1 Dateneinlesen

Wie bereits in der Abbildung 3.1 gesehen und im Abschnitt 4.1 wiederholt, gehört das Dateneinlesen und -interpretieren zu den ersten Pflichten der Vorverarbeitung. Die zu diesem Zweck benötigten Schnittstellen und Algorithmen sind jedoch nicht Gegenstand dieser Arbeit, insofern wird auf sie im Rahmen dieser Ausarbeitung nicht näher eingegangen und ihre Verfügbarkeit als vorausgesetzt angenommen.

4.2.2 Gruppenbildung

Bereits im Abschnitt 3.1.2 wurde auf die Notwendigkeit hingewiesen, die einzelnen Tests der zu untersuchenden Testsequenz zu Gruppen zu vereinen. Die Herleitung der Gruppen aus einer Testsequenz erfolgt einmalig, zu Beginn eines jeden Experiments. Zur Einteilung in Gruppen werden zwei verschieden Techniken verwendet in Abhängigkeit davon, wie vollständig die Beschreibung der Testsequenz in den betrachteten Messdaten ist.

Vollständig beschriebene Testsequenz: Das von den meisten Halbleiterherstellern zum Abspeichern von Messergebnissen verwendete Datenformat trägt den Namen *Standard Test Data Format* (STDF) und wurde von der Firma *Teradyne Inc.*, Boston, USA entwickelt. Im genannten Format finden sich besondere Bereiche, welche speziell für die Definition von Gruppen eingeführt wurden. Weisen die in dieser Arbeit betrachteten Messresultate Gruppenmarkierungen auf, so werden den Kennzeichnungen entsprechend Tests zu Gruppen gebündelt.

Fehlende Gruppenkennzeichnung: Wird ein anderes Datenformat verwendet oder sind die Daten - wie häufig in der Praxis anzutreffen - unvollständig und die Gruppenzugehörigkeiten sind in den Messdaten nicht explizit angegeben, so wird eine einfache Methode zum Bilden von Ersatzgruppen angewandt.

> Man betrachtet dabei die laufenden Testnummern innerhalb der Sequenz und deutet die immer wieder auftretenden Nummerierungsunterbrechungen als Gruppengrenzen.

Die auf diese Weise gewonnenen *virtuellen Gruppen* werden in den folgenden Kapiteln nicht von den echten Gruppen unterschieden und schränken die Allgemeinheit des vorgestellten Verfahrens nicht ein.

Zur Gewährleistung der Vergleichbarkeit aller einer Testgruppe angehörigen Tests, werden alle Tests der Sequenz vor ihrer Zuordnung zu Gruppen auf ein einheitliches Intervall $[0, 1]$ gebracht. Die gebildeten Gruppen sind letztendlich in der Handhabung dank passender Datenstrukturen von einzelnen Tests nicht zu unterscheiden, wenn von ihrer semantischen Bedeutung abgesehen wird. Dies ist auch der Grund dafür, dass im Folgenden nur dann zwischen Tests und Gruppen explizit unterschieden wird, wenn sich dadurch der Sinn der Aussage erheblich ändert.

4.2.3 Informationsgehalt

Der nächste Verarbeitungsschritt des Optimierungsverfahrens befasst sich mit der Auszeichnung mancher Gruppen als Kandidaten für eine mögliche Einsparung. Im Kapitel 2.2 wurde bereits diskutiert, dass sich viele Publikationen einzig auf die Untersuchung des Fehlerverhaltens von Tests und damit auf die Betrachtung ausschließlich fehlerhafter Bauteile konzentrieren. Diese Einschränkung beherbergt jedoch einen erheblichen Schwachpunkt: Wie bereits erwähnt, besitzen viele, vor allem hochvolumige und eingefahrene Produkte eine relativ hohe Ausbeute im Bereich von über 90%. Durch die genannte Einschränkung wird einerseits die Sammlung einer für eine statistische Analyse aussagekräftigen Menge an Daten erschwert und andererseits, dürften auf diese Weise Tests mit ausgesprochen selten bis gar nicht auftretenden Fehlern kaum objektiv untersucht werden können.

Zur Vermeidung des beschriebenen Schwachpunktes verwendet diese Arbeit die Analyse des Informationsgehalts. Dabei werden nicht nur die als Fehler gekennzeichneten Ergebnisse der Tests untersucht, sondern generell alle zur Verfügung stehenden Daten. Die Analyse findet auf allen verfügbaren Lerndaten einmalig, am Anfang eines jeden Experiments statt. Der Erläuterung des Abschnitts 3.3.3 folgend, werden die Gruppen, zur Vorhersage vorgesehener Bauteile, dem Ausgang der Informationsgehaltanalyse entsprechend, eingefärbt.

Das Informationsmaß eines Tests in Abhängigkeit von allen anderen Tests der Testsequenz wird hierbei definiert als:

$$\Delta I(t_x, t_y) = \sum_{d \in D} \min(t'_x(d) - t'_y(d), 0) \tag{4.1}$$

mit $d \in D$ und $t_x, t_y \in S$ und $x \neq y$.

Die Menge D bezeichnet die Gesamtheit aller (zum Lernen) verfügbaren Bauteile, die Menge S steht stellvertretend für die Sequenz aller Tests, $t'_i(d)$ entspricht dem normierten Messwert des Tests t_i auf dem Bauteil d. Die Normierung erfolgt, wie unter Abschnitt 4.2.2 erwähnt auf das Intervall $[0, 1]$. Die Besonderheit ist dabei, dass der Bestwert eines Tests stets auf die Null und die Werte außerhalb des Toleranzintervalls auf die Eins abgebildet werden, alle anderen Werte bekommen einen Wert zwischen den Extremen. Für weitere Details dieser Methode sei auf [RvSK05] verwiesen.

Die Idee hinter diesem Verfahren ist die Ermittlung des Beitrags eines jeden Tests zum Informationsgehalt der gesamten Testsequenz. Ist der Beitrag eines Tests kleiner als ein vorgegebener Schwellenwert, kann dieser zwecks Testzeitreduktion zur Auslassung vorgeschlagen werden. Es ist zu beachten, dass bei zwei identischen Tests nur einer zur Optimierung vorgeschlagen wird - welcher das sein wird, ist mehr oder weniger dem Zufall überlassen und sollte keine weiteren Auswirkungen haben.

4.2.3.1 Stabilitätskriterien

Zusätzlich zu dem in der Gleichung 4.1 definierten Informationsgehalt bestimmen in dieser Arbeit auch noch weitere Stabilitätskriterien eines Tests über dessen mögliche Auslassbarkeit. Diese Kriterien sind unter anderem:

Ausfallwahrscheinlichkeit: Der wahrscheinlich gewichtigste Faktor für die Stabilität eine Tests, abgesehen von dessen Informationsgehalt ist die Häufigkeit, mit der ein bestimmter Test Fehler auf einem Bauteil erkennt. Geschieht dies selten bis gar nicht, ist das ein gutes Indiz dafür, dass der Test ohne einen großen Verlust an Informationen aus einer Testsequenz ausgelassen werden kann.

Korrelationskoeffizient: Der Korrelationskoeffizient (Gleichung 4.8) soll die beobachtbare Beziehung eines Tests mit einem anderen darstellen. Ist ein Test t_i mit einem anderen Test t_j besonders gut korreliert, so bietet es sich an, einen der beiden Korrelationspartner auszulassen, um beispielsweise mit Hilfe von Test t_i die Messwerte des Tests t_j zu rekonstruieren.

Prozessfähigkeitsindex: Der Prozessfähigkeitsindex C_{pk} (definiert in Gleichung 2.1) identifiziert zugleich mehrere für die Einsparung des Tests bedeutende Dinge. Einerseits lässt es sich anhand des Indexes bemessen, wie häufig eine Messung Resultate jenseits der Toleranzgrenzen liefert. Andererseits erkennt man, wie nah die Toleranzgrenzen an dem Mittelwert und der Standardabweichung des untersuchten Tests liegen.

Bei allen genannten Kriterien ist jedoch weiterhin Vorsicht geboten, denn keines der genannten Faktoren spiegelt einen kausalen Zusammenhang wider. Stattdessen beschreiben diese Kriterien die bisher vorliegenden, sichtbaren Ergebnisse eines womöglich unsichtbaren Prozesses, dessen Parameter sich nicht genau bestimmen lassen. Eine Bestätigung dieser Aussage ist die in der Praxis beobachtbare Tatsache, dass einzelne Tests eines hoch korrelierten Verbundes, alle jeweils mit einer sehr kleinen Ausfallwahrscheinlichkeit und damit verbunden einem hohen Prozessfähigkeitsindex auf manchen Bauteilen Ausfälle feststellen, während andere Tests des gleichen Verbundes auf den genannten Bauteilen keine signifikante Veränderung zeigen.

4.3 Datenoptimierung

Nach dem die Messdaten vom Datenträger eingelesen, interpretiert, zu Gruppen gebündelt und gemäß ihres Informationsgehalts eingefärbt wurden, müssen die rohen unverarbeiteten Daten gefiltert und für den Einsatz durch einen Klassifikationsalgorithmus optimiert werden. Die einzelnen zu erfüllenden Aufgaben finden sich in den Abschnitten 3.3.5, 3.3.6 und 4.1. Alle hierbei verwendeten Algorithmen sind der Gegenstand dieses Abschnittes.

4.3.1 Filter

In der vorliegenden Arbeit werden die unter dem Begriff *Filter* zusammengefassten Algorithmen zur Bewältigung mehrerer Aufgaben eingesetzt. Das Ziel und Einsatzgebiet dieser Verfahren ist jedoch stets das gleiche: Filter dienen der Aufbereitung neu eingelesener, roher Daten, um ihre Klassifikation durch den Klassifikationsalgorithmus zu erleichtern. Die Ausgabe der Filter wird in dieser Arbeit als *bereinigte Messwerte* bezeichnet. Ihre Gesamtheit bestehend aus n solchen bereinigten Messergebnissen, wird als ein n-dimensionaler *Vektor* an den Klassifikator übergeben.

> Die Menge aller n-dimensionaler Vektoren spannt einen n-dimensionalen *Merkmalsraum* auf. Die einzelnen Merkmale sind, entweder die bereinigten Messwerte direkt oder weitere, diese Messwerte beschreibenden, Eigenschaften (4.3.1.2). Jeder Vektor stellt einen Punkt des Merkmalsraums dar.

Eine Voraussetzung für manche weiter unten beschriebenen Techniken muss an dieser Stelle festgehalten werden: Während des Lernvorganges findet eine statistische Analyse aller Trainingsdaten statt. Im Rahmen dieser werden unter anderem die Quantile und die Korrelationskoeffizienten der einzelnen Tests bestimmt.

Die in der Abbildung 4.2 dargestellten Aufgaben der Filteralgorithmen werden auf Basis roher Messwerte, mit Hilfe der, aus der statistischen Analyse gewonnenen Informationen, bewältigt. Die von den einzelnen Schritten dargestellten Aufgaben sind im Einzelnen:

1. **Schritt:** Selektion bestimmter Messwerte, anhand verfügbarer Informationen aus der statistischen Analyse (Abschnitt: 4.3.1.1)

2. **Schritt:** Anreicherung roher Messwerte mit Zusatzinformationen aus der statistischen Analyse (Abschnitt: 4.3.1.2)

3. **Schritt:** Normierung der selektierten, rohen Messwerte und der entsprechenden Zusatzinformationen auf ein bestimmtes Intervall (Abschnitt: 4.3.1.3)

4. **Schritt:** Minimierung vom statischen Rauschen und Ausgleich möglicher Prozessschwankungen (Abschnitt: 4.3.1.4)

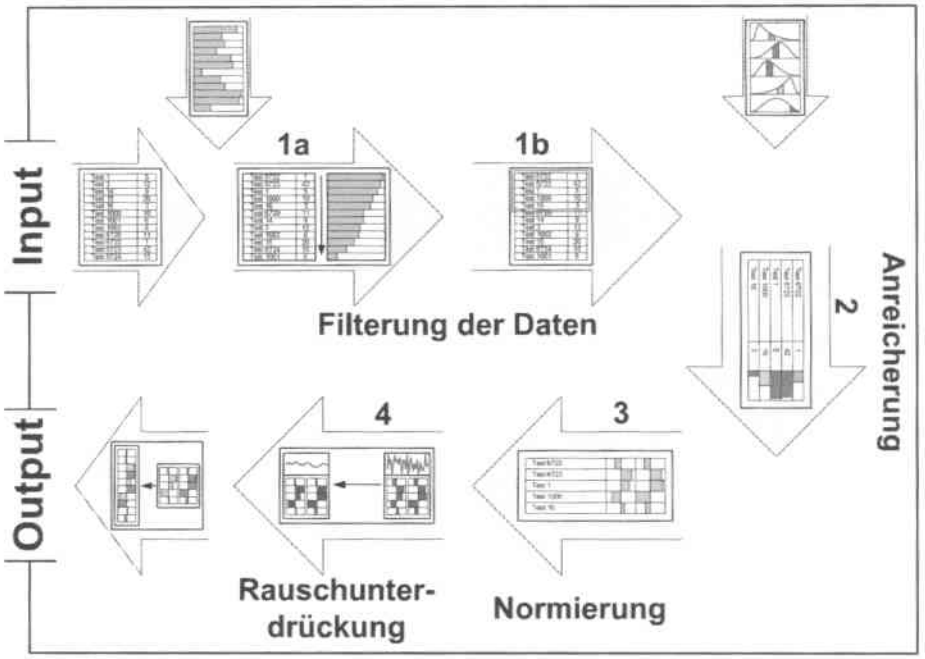


Abbildung 4.2: Skizzenhafte Darstellung der von den Filtern erfüllten Aufgaben

Die Ausgabe der filtrierten und bereinigten Messwerte erfolgt durch ihre Bündelung zu einem Merkmalsvektor.

Im Folgenden findet sich die Auflistung der im Einzelnen von den Filtern erledigter Aufgaben und der zur ihrer Lösung eingesetzter Methoden.

4.3.1.1 Filterung

Die Tätigkeit, durch welche die im Abschnitt 4.3.1 beschriebenen Techniken zu ihrem Namen kamen, ist das Filtern. Die Filterung geschieht auf Basis einer während des Lernvorgangs durchgeführten statistischen Analyse. Während dieser werden unter anderem die Korrelationen der Tests untereinander bestimmt.

Im laufenden Betrieb des Testers bzw. des Experiments werden den Filtern nach dem Abschluss der Vorverarbeitung rohe Daten übergeben. Die übergebenen Messergebnisse sind mit Bezeichnern der Tests markiert, welchen sie entstammen. Weiterhin besitzen die Filter Kenntnis darüber, welche Testgruppe vorhergesagt werden soll und aus welchen Tests diese Gruppe besteht. So ist es den Filtern möglich, aus der Menge der übergebenen Daten, beispielsweise nur die 20 Prozent höchst korrelierter Messwerte zu selektieren. Die restliche Messergebnisse werden vor den Klassifikatoren bzw. der Weiterverarbeitung unterschlagen.

Eine andere Implementierung wählt aus den rohen Messwerten die k *kritischsten* Messergebnisse und verwendet für die Klassifikation ausschließlich die-

se. Wie kritisch die Messung $t(d)$ eines Tests t mit dem Mittelwert μ und dem Toleranzintervall r auf dem Bauteil d ist, lässt sich bestimmen durch

$$\frac{|\mu - t(d)|}{r}$$

also durch die Normierung des Abstandes der Messung zum Mittelwert des Tests, durch die Länge des Intervalls.

Die oben beschriebene, gezielte Auswahl bestimmter Messresultate und die Entfernung restlicher Messwerte, begrenzt die Dimensionalität der gesamten Messung und ermöglicht dem Klassifikator dadurch ein schnelleres und leichteres Erlernen sowie Vorhersagen der Daten.

4.3.1.2 Datenanreicherung

Eine weitere durch die Filter erledigte Aufgabe ist die Anreicherung vorgelegter Daten mit zusätzlichen Informationen. Die zusätzlichen Informationen können dem Klassifikationsalgorithmus helfen, die Bedeutung der ihm zur Klassifikation übergebenen Daten zu erkennen.

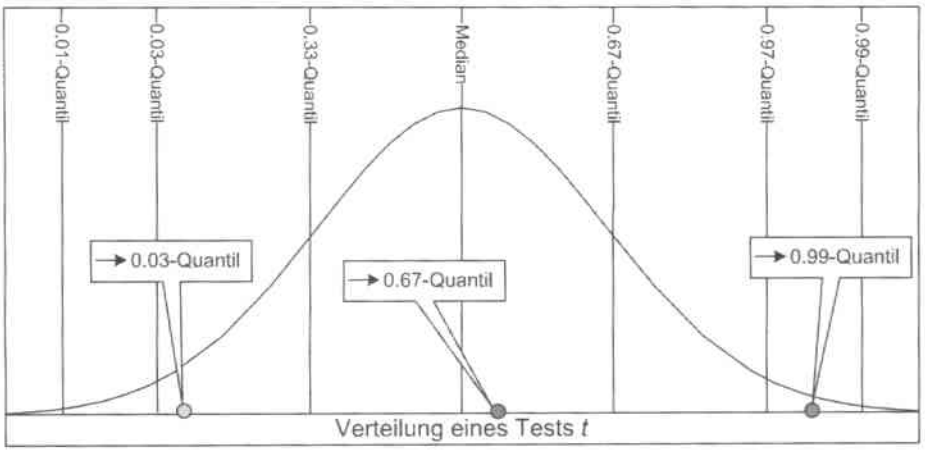


Abbildung 4.3: Beispiel: Zuordnung unterschiedlicher Werte eines Tests t zu ihren Quantilen

Im einfachsten Fall können die den rohen Messwerten *entsprechenden Quantile* hinzugefügt werden. Als Hinzufügen wird hier die Vermengung der bereits vorhandenen n rohen Messwerten mit n passenden Quantilen verstanden, sodass als Ergebnis $2n$ Werte entstehen, dargestellt im dritten Schritt der Abbildung 4.2. Alternativ lassen sich die rohen Messwerte auch ganz durch die entsprechenden Quantile ersetzen. Was mit einem *entsprechenden Quantil* gemeint ist, soll an dem, in der Abbildung 4.3 dargestellten Beispiel, erläutert werden.

Im Beispiel sollen Quantile bestimmt werden, welchen den drei farbig markierten Punkten entsprechen. Zu diesem Zweck wird für jeden Punkt das

nächstgelegene Quantil so gewählt, dass sich der Punkt im Intervall zwischen dem gewählten Quantil und dem Median befindet.

Eine weitere denkbare Möglichkeit besteht in der Anreicherung der Messwerte mit den Korrelationskoeffizienten entsprechender Tests. Der Prozess der Anreicherung kann durch die weiter oben beschriebene Hinzunahme oder das Ausmultiplizieren der Korrelationskoeffizienten mit den vorher normierten (siehe dazu Abschnitt 4.3.1.3) Messwerten erfolgen.

4.3.1.3 Normierung

Bedingt durch das Objekt der Messung können sich die Wertebereiche verschiedener Tests gravierend unterscheiden. Die Annahme, dass der große Wertebereich eines Tests eine größere Rolle, als ein kleinerer Wertebereich eines anderen Tests, spielt, würde das Optimierungsverfahren *ad absurdum* führen. Zur Angleichung jeglicher Wertebereichsunterschiede werden deshalb alle gemessenen Werte auf ein einheitliches offenes Intervall $(-1, 1)$ normiert. Die Normierung einer Messung x erfolgt gemäß einer der folgenden Definitionen:

$$N_1(t(d)) := \frac{2}{\pi} \arctan\left(\frac{t(d) - \mu_t}{\sigma_t}\right) \tag{4.2}$$

$$N_3(t(d)) := \frac{2}{\pi} \arctan\left(\left(\frac{t(d) - \mu_t}{\sigma_t}\right)^3\right) \tag{4.3}$$

$$N_{3\sigma}(t(d)) := \frac{2}{\pi} \arctan\left(\left(\frac{t(d) - \mu_t}{3\sigma_t}\right)^3\right) \tag{4.4}$$

Hierbei repräsentiert $t(d)$ die Messung des Tests t auf dem Bauteil d, μ_t entspricht dem Mittelwert und σ_t der Standardabweichung des Tests t. Der Gedanke hinter dieser Normierung liegt in der Abbildung aller Messwerte nahe dem Testmittelwert auf einen Bildbereich nahe der Null. Die Messwerte, die innerhalb eines bestimmten Definitionsbereiches $i_{0.5}$ liegen, sollen auf den Bildbereich $[-0.5, 0.5]$, alle Werte jenseits des Definitionsbereiches $i_{0.5}$, auf die Ränder des Bildbereichs abgebildet werden.

Der Definitionsbereich $i_{0.5}$ und die Art der Abbildung variieren unter den aufgeführten Funktionen. Die Funktionen N_1 und N_3 (Gleichungen 4.2 und 4.3) besitzen zwar den gleichen Definitionsbereich $i_{0.5}$, welcher den Grenzen der Standardabweichung um den Mittelwert des Tests entspricht, jedoch bewirkt die Art der Abbildung der Funktion N_3 eine zusätzliche Kompression aller nahe des Mittelwerts liegender Punkte. Diese in der Nähe von μ lokalisierten Punkte werden auf einen schmalen Bereich um die Null abgebildet. Die Kompression könnte vor Allem dann sinnvoll sein, wenn damit zu rechnen ist, dass die Messdaten gewisses Rauschen aufweisen. Das Rauschen würde durch die Normierung mit der Funktion N_3 abgeschwächt werden.

Die Funktionen N_3 und $N_{3\sigma}$ (Gleichungen 4.3 und 4.4) unterscheiden sich hingegen in dem Definitionsbereich $i_{0.5}$ und nicht in der Art der Abbildung. Der Definitionsbereich der Funktion $N_{3\sigma}$ wurde gemäß des *Six Sigma* Ansatzes

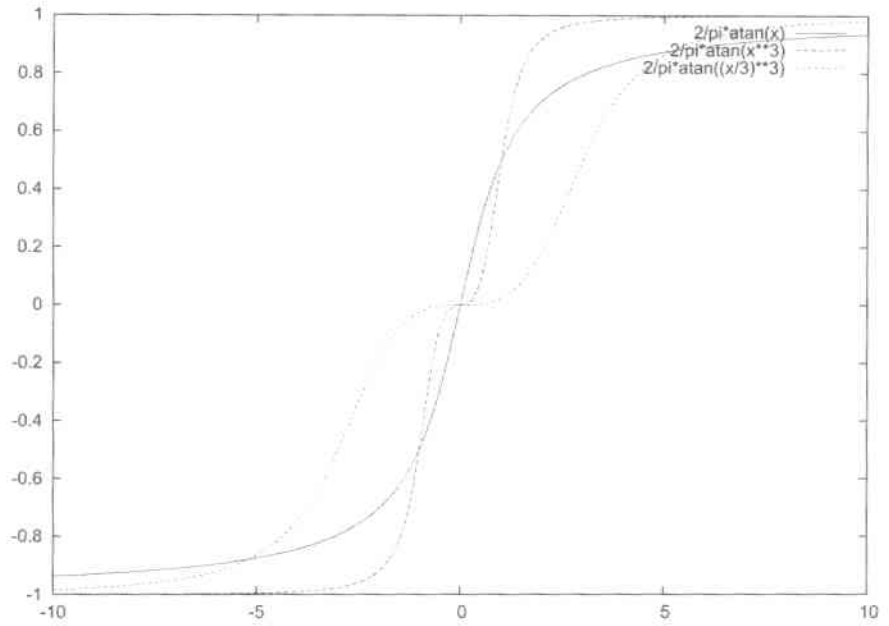


Abbildung 4.4: Die unterschiedlichen Normierungsfunktionen (Annahme: $\mu_t = 0$ und $\sigma_t = 1$)

(siehe dazu [FLJ91]) gewählt und beträgt den Abstand der dreifachen Standardabweichung um den Mittelwert des Tests t. Mit der Wahl dieser Grenzen, wird nicht nur jegliches Rauschen um den Mittelpunkt, sondern auch alle Schwankungen innerhalb der Standardabweichung des Tests stark komprimiert. Damit werden gerade die Messresultate des Tests gut aufgelöst, die nicht zu dem üblichen Verhalten des Tests zu passen scheinen und somit mögliche Fehler des Bauteils d ankündigen können.

Eine weitere Schlussfolgerung der vorgestellten Normierungsverfahren, lässt die auf den Bereich nahe der Null abgebildeten Messwerte, ein gewöhnliches Verhalten des untersuchten Tests vermuten.

4.3.1.4 Rauschunterdrückung & Protzesschwankungen

Für eine stabile Funktionsweise des Optimierungsverfahrens werden Verfahren benötigt, die in der Lage sind, das beim analogen Messen sich einschleichendes Rauschen auf ein Minimum zu reduzieren oder ganz auszublenden. Beim Tester kann das Rauschen als minimale Störung an mehreren Stellen entstehen, denkbar wären:

- ein etwas variierender Anpressdruck von Testerkopfnadeln an die dafür vorgesehenen Kontakte,
- minimale Temperaturschwankungen im Testerraum oder
- eine unzureichende Auflösung der durchgeführten Messung.

Weiterhin unterliegen die Tests beispielsweise bedingt durch die fortwährende Abnutzung der Testköpfe sinoiden Schwankungen. Zwei Mechanismen zur Minimierung der durch das Rauschen und die Schwankungen verursachten Störungen, kommen zum Einsatz.

Normierung: Im Abschnitt 4.3.1.3 finden sich zwei Normierungsfunktionen N_3 und $N_{3\sigma}$ (Gl. 4.3 und 4.4), welche sich zur Filterung von Rauschen eignen würden. Beide Funktionen bewirken durch ihre Abbildungen die Kompression eines bestimmten Definitionsbereichs. Nimmt man an, dass sich das Rauschen genau innerhalb dieses Definitionsbereiches am meisten auswirkt, kann es nach der Normierung als unterdrückt betrachtet werden.

Betrachtungspuffer Zur Entfernung sinoider Messschwankungen kann ein so genannter *Betrachtungspuffer* eingesetzt werden. Die Funktionsweise und der Nutzen dieses Puffers werden nachfolgend diskutiert.

Betrachtungspuffer

Eine in der Praxis nicht unübliche auftretende Erscheinung lässt einen Test im laufenden Betrieb seinen Wertebereich in die eine oder andere Richtung verschieben. Würde man in diesem Fall den Wertebereich des gesamten Tests betrachten, um seinen Mittelwert bzw. seine Standardabweichung zu bestimmen, würde der ermittelte Mittelwert nicht dem Mittelwert der aktuellen Schwankungsphase entsprechen und die Standardabweichung wäre zu groß. Aus diesem Grund erscheint es sinnvoller, zur Berechnung der im Abschnitt 4.3.1.3 vorgestellten Normierungsfunktionen den Mittelwert und die Standardabweichung ausschließlich auf die letzten n Messwerte eines Tests zuzugreifen.

Die Implementierung der Zwischenspeicherung der letzten n Messwerte erfolgt mit Hilfe eines Puffers. Der Puffer (*buffer*) besitzt eine feste Länge, sodass beim Hinzunehmen des $(n+1)$-ten Wertes, der erste Wert des Puffers automatisch entfernt wird. Somit ist die Berechnung der Statistiken ausschließlich auf n aktuellsten Messwerten eines Tests sichergestellt (siehe auch 4.5).

4.3.2 Allgemeine Verfahren

Nachdem die roh eingelesenen Messdaten durch die Filter bereinigt und zu Vektoren zusammengefasst wurden, können diese noch zusätzlich optimiert werden, denn zumeist bestehen die mit Informationen angereicherten Vektoren (4.3.1.2) aus einigen hundert Koeffizienten, womit sie einen hochdimensionalen Merkmalsraum[2] aufspannen.

Auf Grund der hohen Dimensionalität des Merkmalsraumes, kann der eingesetzte Klassifikationsalgorithmus durchaus *sehr lange* brauchen, bis er die Wichtigkeit der einzelnen Merkmale erkennt. Sehr lange bedeutet hierbei nichts anderes, als eine sehr große Menge an Lerndaten. Die Anzahl für das Training zur Verfügung stehender Daten ist aber, wie immer, begrenzt. Hierdurch entsteht der Bedarf an Algorithmen, die den Merkmalsraum insofern

[2]Wie bereits im Abschnitt 4.3.1 definiert, repräsentieren die Merkmale des Raumes entweder die bereinigten Messergebnisse selbst, oder bestimmte statistischen Charakteristiken (wie Korrelationskoeffizienten o. ä.), die diese Messergebnisse beschreiben.

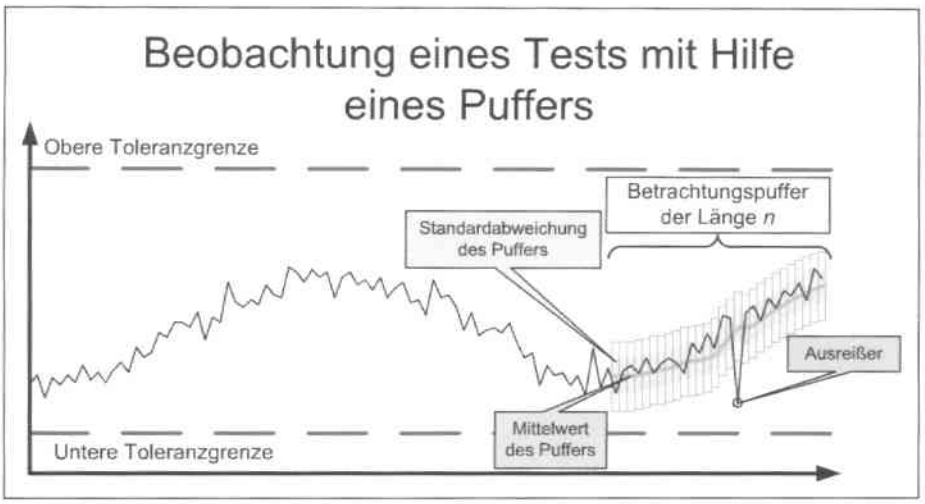


Abbildung 4.5: Schematische Darstellung eines Betrachtungspuffers der Länge n auf einem Test mit sinoider Schwankung

vereinfachen, als dass der Lerneffekt des Klassifikators beschleunigt und die Entscheidungsfindung erleichtert werden können.

Alle in diesem Abschnitt aufgeführten Verfahren lassen sich sowohl einzeln einsetzen, als auch in beliebigen Kombinationen untereinander.

4.3.2.1 *k-means* Clustering

Unter Verwendung des *k-means*-Algorithmus [DHS00a] wird der gesamte Merkmalsraum in eine gewünschte Anzahl von *Clustern* unterteilt. Ein Cluster entspricht einer Ballung von mehreren Mermalsvektoren. Die eingesetzte Implementierung des *k-means*-Algorithmus zeichnet sich durch folgende Vorgehensweise aus:

1. Es werden initial k Clustermittelpunkte im Merkmalsraum verteilt. Das erste Clusterzentrum wird direkt auf einen zufällig gewählten Merkmalsvektor gelegt. Die restlichen $(k-1)$ Clustermittelpunkte werden um einen kleinen Betrag zu dem ersten Punkt versetzt.

2. Alle Vektoren des Merkmalsraumes, welche *näher* am Mittelpunkt μ_i des Clusters c_i, als am Mittelpunkt jedes anderen Clusters im Merkmalsraum liegen, werden dem Cluster c_i zugeordnet.

3. Die Clustermittelpunkte werden auf den Mittelpunktvektor aller, dem jeweiligen Cluster zugeordneten Vektoren, verschoben.

4. Die Schritte 2 und 3 werden solange wiederholt, bis die Clustermittelpunkte nicht mehr bewegt werden oder die maximale Anzahl der Iterationen (zumeist 1000) erreicht ist.

Je nach Art des durchgeführten Experiments, können für die Merkmalsvektoren beider Klassen C_M und C_R dieselben, aber auch jeweils unterschiedliche Cluster gebildet werden. Die unter Punkt 2 erwähnte Nähe soll als die *euklidische Distanz* definiert sein.

Die euklidische Distanz, ist für zwei n-dimensionale Merkmalsvektoren folgendermaßen festgelegt:

> Seien die beiden Vektoren x und y gegeben durch die Koordinaten $x = (x_1, \ldots, x_n)$ und $y = (y_1, \ldots, y_n)$, so entspricht die euklidische Distanz beider Vektoren:
>
> $$d(x,y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (4.5)$$

4.3.2.1.1 Zusatzinformationen Weiterhin kann es für die im nächsten Schritt des Optimierungsverfahrens folgende Klassifikation sinnvoll sein, die Cluster mit zusätzlichen Informationen zu versehen. Falls tatsächlich vorhanden, steht die Information, allen im Abschnitt 4.4 beschriebenen Klassifikatoren zur Verfügung und kann beispielsweise eingesetzt werden, um dicht bevölkerte Cluster in die Entscheidung stärker einfließen zu lassen als dünn besiedelte.

Insgesamt können die gebildeten Cluster folgende Arten von Zusatzinformationen tragen, die den Klassifikatoren als Gewichtung für die Entscheidungsfindung dienen kann:

Klassen: Verwendet man für die Merkmalsvektoren beider Klassen C_M und C_R dieselben Cluster, so kann jeder Cluster ein unterschiedliches Verhältnis von Vektoren der Klasse C_M zu Vektoren der Klasse C_R in sich tragen. Dieses Verhältnis kann für die Klassifikation entscheidend sein und wird deshalb für jeden Cluster abgespeichert.

Clustergröße: Die Anzahl der von einem Clusters gebündelten Merkmalsvektoren kann für die Entscheidungsfindung ebenfalls wichtig sein, weswegen sie gleichfalls in jedem Cluster abgelegt wird.

4.3.2.2 Learning Vector Quantization

Die Learning Vector Quantization (LVQ) [Koh90] kommt in dieser Arbeit zum Einsatz, um eine ganz bestimmte Art von Fehlern zu minimieren. Wie im Abschnitt 2.3 erklärt, können dem Klassifikationsverfahren zwei Arten von Fehlern unterlaufen:

Typ 1: Die Ausführung eines Tests bzw. einer Gruppe, obwohl nicht notwendig

Typ 2: Die Einsparung eines Tests bzw. einer Gruppe, obwohl diese auf dem untersuchten Bauteil einen Fehler feststellt

Die beiden genannten Fehlerarten müssen unterschiedlich bewertet werden, da die durch sie verursachten Kosten unterschiedlich hoch sind. Nach den im

Abschnitt 3.1.1 gemachten Annahmen, betragen die Kosten für den zweiten Fehlertyp das hundertfache der Kosten eines Fehlers vom Typ 1. Aufgrund dieser Differenz, konzentriert sich das LVQ-Verfahren auf die Verringerung der Fehler vom Typ 2 und nimmt ein dadurch bedingtes, häufigeres Auftreten des ersten Fehlertyps bewusst in Kauf. Trotz des dadurch möglichen Anstieges der Gesamtfehlerzahl (vom Typ 1) können auf diese Weise die Gesamtkosten trotzdem gesenkt werden.

Die Idee hinter diesem Verfahren besteht zunächst in der Herausbildung von *Repräsentanten* für die Merkmalsvektoren beider Klassen C_M und C_R, um anschließend die Repräsentanten der Klasse C_M näher an die Grenzen des von den Repräsentanten der Klasse C_R bevölkerten Gebietes zu verschieben. Gerade diese Verschiebung bezweckt die, weiter oben beschriebene, häufigere Klassifikation von Merkmalsvektoren zur Klasse C_M, obwohl diese in der ursprünglichen Konstellation der Klasse C_R zugeordnet worden wären. Das Ziel ist also die Klassifikation aller „unsicheren" Vektoren zur Klasse C_M.

Die Repräsentanten einer Klasse werden in dieser Arbeit, auf die gleiche Weise berechnet, wie die Cluster im Abschnitt 4.3.2.1. Auf die gleiche Art und Weise wie unter 4.3.2.1.1 beschrieben, lassen sich auch die Repräsentanten mit Zusatzinformationen versehen.

4.3.2.3 Lineare Diskriminanzanalyse

Wie bereits in [Fei05] kommt auch in dieser Arbeit die lineare Diskriminanzanalyse [DHS00b] zum Einsatz. Das vom Biologen *Ronald Aylmer Fisher* im Jahre 1936 entwickelte Verfahren versucht, mit Hilfe einer geeigneten Abbildung des Merkmalsraumes auf eine Gerade, die Klassifikation eines Merkmals bzw. Merkmalsvektors zu einer von zwei möglichen Klassen zu erleichtern.

Sei x ein im Merkmalsraum vorhandener Vektor, so wird ein Abbildungsvektor w gesucht, sodass

$$y = w^\top x \tag{4.6}$$

die Abbildungen aller zur Klasse C_M gehörenden x möglichst gut von allen zur Klasse C_R gehörenden x unterschieden werden können. Das dabei zum Einsatz kommende Kriterium ist wie folgt definiert:

$$J(w) = \frac{|\tilde{m}_M - \tilde{m}_R|^2}{\tilde{s}_M^2 + \tilde{s}_R^2} \tag{4.7}$$

$\tilde{m}_M$ und $\tilde{m}_R$ entsprechen hierbei den abgebildeten Mittelpunkten der beiden Klassen C_M und C_R, $(\tilde{s}_M^2 + \tilde{s}_R^2)$ repräsentiert die Intravarianz (*within scatter*) der Projektionen beider Klassen.

Die Implementierung dieses Verfahrens wurde als Bibliothek des, an der Universität Karlsruhe (TH) vom Institut für Theorethische Informatik, Lehrstuhl Professor Dr. A. Waibel entwickelten Spracherkenners JANUS, in diese Arbeit eingebunden.

4.3.2.4 Selbstorganisierende Feature Maps

Ein weiteres eingesetztes Optimierungsverfahren basiert auf den, vom finnischen Ingenieur *Teuvo Kohonen* erforschten, selbstorganisierenden Karten [DHS00c]. Das Ziel dieser Technik ist die Repräsentation aller Punkte des Merkmalsraums durch Punkte in einem Raum mit wenigen Dimensionen, unter der Bedingung, dass die Distanzen und die Nachbarschaftsbeziehungen der Merkmalsvektoren möglichst erhalten werden sollen.

Die in dieser Arbeit implementierte Variante des Verfahrens geht folgendermaßen vor:

1. Ein Netz von $k \times k$ Knoten wird erstellt.
2. Alle Knoten bekommen nach dem Zufallsprinzip bestimmte Koordinaten des Merkmalsraumes zugewiesen.
3. Jeder sich im Merkmalsraum befindender Vektor wird dem (im Merkmalsraum) nächstgelegenen Knoten des Netzes zugeordnet.
4. Mit jeder Zuordnung eines Merkmalsvektors zu dem Knoten n_{ij}, der sich in der i-ten Zeile und j-ten Spalte des $k \times k$ Netzes befindet, werden die Koordinaten dieses Knotens im Merkmalsraum in Richtung des zugeordneten Vektors bewegt.
5. Kennzeichne n^*_{ij} einen Knoten des Netzes, dem gerade ein Merkmalsvektor zugeordnet wurde und sei n_{kl} ein beliebiger, anderer Knoten des Netzes. So vollzieht der Knoten n_{kl} ebenfalls die vom Knoten n^*_{ij} durchgeführte Koordinatenverschiebung im Merkmalsraum, jedoch wird diese mit Hilfe der Fensterfunktion $\Lambda(n^*_{ij}, n_{kl})$ abgeschwächt. Die in dieser Arbeit eingesetzte Fensterfunktion ist wie folgt definiert:

$$\Lambda(n^*_{ij}, n_{kl}) = 2^{-\max[|i-k|,|j-l|]}$$

6. Die Schritte 3 bis 5 werden eine vorgegebene Anzahl an Iterationen wiederholt (zumeist 50).

Die Verschiebung der Koordinaten im Merkmalsraum wird zusätzlich durch eine *imaginäre Temperatur* gedämpft, welche im Laufe des Lernvorganges stetig abnimmt.

Alle Knoten des Netzes erhalten, auf die im Abschnitt 4.3.2.1.1 dargelegte Art, bestimmte Zusatzinformationen. Die Zusatzinformationen können die Anzahl den Knoten zugeordneter Vektoren enthalten, oder auch das Verhältnis der Klassen innerhalb der zugeordneten Merkmalsvektoren.

4.3.2.5 Ausfallmuster

Das letzte in dieser Arbeit vorgestellte allgemeine Optimierungsverfahren, basiert auf einer selbsterstellten Heuristik. Diese Technik entstand aus der Beobachtung heraus, dass in der Praxis der Einsatz von Korrelationen leider allzu oft

an seine Grenzen stößt. Die Ursache dafür kann in der Definition des Korrelationskoeffizienten ϱ vermutet werden. Dieser ist für zwei Tests t_1 und t_2 definiert als:

$$\varrho(t_1, t_2) := \frac{\mathrm{cov}(t_1, t_2)}{\sqrt{\mathrm{var}(t_1)} \cdot \sqrt{\mathrm{var}(t_2)}} = \frac{\mathrm{E}\left[(t_1 - \mathrm{E}(t_1))(t_2 - \mathrm{E}(t_2))\right]}{\sqrt{\mathrm{var}(t_1)} \cdot \sqrt{\mathrm{var}(t_2)}} \tag{4.8}$$

und

$$\begin{aligned} \mathrm{var}(t) &:= \mathrm{cov}(t, t) \\ \mathrm{cov}(t_1, t_2) &:= \mathrm{E}\left((t_1 - \mathrm{E}(t_1))(t_2 - \mathrm{E}(t_2))\right) \end{aligned}$$

$\mathrm{E}(t)$ entspricht zwar dem Erwartungswert eines Tests t, aus praktischen Gründen jedoch, wird in dieser Arbeit dafür der Mittelwert aller verfügbarer Messwerte des Tests t verwendet.

Die Berechnung der Kovarianzen erfolgt auf allen vorhandenen Bauteilen, wobei die Messwerte außerhalb der Toleranzgrenzen keine zusätzliche Gewichtung oder Betonung erhalten. Weiterhin ist der Korrelationskoeffizient nicht in der Lage kausale Zusammenhänge anzuzeigen. Auf Grund des genannten, lässt sich in der Praxis vor allem bei Produkten mit hoher Ausbeute folgende Konstellation beobachten:

> Zwei Tests t_1 und t_2, welche ein sehr ähnliches Verhalten aufzeigen, vermessen mehrere Tausend Chips. Einer der beiden Tests t_1 misst mit einer geringen Wahrscheinlichkeit ($< 0.1\%$) immer wieder Fehler. Der andere Test t_2 hingegen, zeigt auf den vom Test t_1 als fehlerhaft markierten Bauteilen, keine signifikante Veränderung seines Verhaltens.

In diesem Beispiel könnte der Korrelationskoeffizient beider Tests durchaus sehr hohe Werte im Bereich von über 90 Prozent annehmen, und trotzdem ließe sich einer der beiden Tests nicht zum Vorhersagen des Fehlerverhaltens des anderen Tests verwenden. Auf Grund dieser Beobachtung, ist in dieser Arbeit (zusätzlich zu der Korrelationsanalyse) die *Ausfallmusteranalyse* implementiert. Sie bestimmt genau die Tests, die als Identifikatoren verwendet werden können, um fehlerhafte Messungen anderer Tests vorherzusagen. Die Analyse versucht diese Tests aus der Testsequenz herauszufiltern, um sie zur Rekonstruktion von, vor allem außerhalb der Toleranzgrenzen liegender, Messwerte vorherzusagender Tests zu benutzen.

Im Gegensatz zur Berechnung der Korrelationskoeffizienten, werden bei der Ausfallmusteranalyse die Zusammenhänge auf fehlerhaften Bauteilen gegenüber den intakten Bauteilen durch passende Gewichtung deutlich hervorgehoben. Der dafür eingesetzte Algorithmus geht bei der Auswahl der möglichen Kandidaten wie folgt vor:

1. Für einen vorherzusagenden Test t wird zunächst die Menge aller vorhergehenden, bereits ausgeführten Tests bestimmt, die im weiteren Verlauf des Algorithmus beobachtet werden. Die Menge soll im Folgenden als O (*observed*) referenziert werden.

2. In einer Rankingliste R initialisiert man nun für jeden beobachteten Test $t_o \in O$ seinen Rankingwert r_o mit Null.

3. Für jedes trainierte Bauteil x, berechnet man für jeden beobachteten Test t_o die Distanz $d(t_o(x))$. Die Distanz $d(t_o(x))$ ist definiert als:

$$\begin{aligned} d(t_o(x)) &:= |N(t_o(x))| \\ N(t_o(x)) &:= \frac{t_o(x) - \mu'_{t_o}}{\sigma'_{t_o}} \end{aligned}$$

   wobei $t_o(x)$ der Messwert des beobachteten Tests $t_o \in O$ auf dem Bauteil x ist. $N(t_o(x))$ entspricht dem normierten Messwert $t_o(x)$. Für die Normierung werden der Mittelwert μ'_{t_o} und die Standardabweichung σ'_{t_o} des Betrachtungspuffers (siehe dazu Abschnitt 4.3.1.4) des Tests t_o verwendet.

4. Mit der berechneten Abweichung $d(t_o(x))$ wird nun der passende Eintrag r_o der Rankingliste aktualisiert:

$$r_o = r_o + \begin{cases} d(t_o(x)) & \text{falls} \quad L_l > t(x) \text{ oder } L_u < t(x) \\ w \cdot d(t_o(x)) & \text{falls} \quad L_l \leq t(x) \leq L_u \end{cases}$$

   Hierbei bezeichnen L_l bzw. L_u die untere bzw. obere Toleranzgrenze des vorhergesagten Tests t. Penaltykoeffizient $w \in [-1, 0]$ wird verwendet, um das Ranking eines Tests t_o zu verringern. Die Idee dahinter ist die Bestrafung eines beobachteten Tests t_o, falls dieser auf einem Bauteil auf dem der vorhergesagte Test t keinen Fehler liefert, trotzdem einen Wert ungleich dem Mittelwert misst.

5. Nach der Betrachtung aller Lerndaten, lassen sich alle Rankingwerte $r_o \in R$ ihrer Größe nach absteigend sortieren. Je höher der Rankingwert eines beobachteten Tests t_o ist, umso wertvoller ist er für die Vorhersage des Fehlerverhaltens des Tests t.

4.4 Klassifikation

Die Klassifikatoren sind der alles entscheidende Kern dieser Arbeit, schließlich sind sie diejenigen, die darüber entscheiden, wie hoch die erzielten Effizienzsteigerungen tatsächlich sind, wie viel Geld eingespart werden kann und wie hoch der verbleibende Anteil unentdeckter, schlechter Bauteile ausfällt. Zur Bestimmung des geeigneten Optimierungsverfahrens verwendet diese Arbeit folgende Klassifikationsalgorithmen:

- Konstante Entscheider (4.4.1.1)
- Zufallsbasierten Klassifikator (4.4.1.2)
- Bayes Klassifikation (4.4.1.3)
- k Nächste Nachbarn Klassifikation (4.4.2.1)
- Multivariaten Gaußklassifikator (4.4.2.2)
- Multilayer-Perzeptron (4.4.2.3)
- Hidden-Markov-Modell basierten Entscheider (4.4.3.1)
- Prozessfähigkeitsindex Klassifikation (4.4.4.1)

Die genannten Verfahren werden zwecks besserer Übersicht in drei Hauptabschnitte gegliedert: 4.4.1, 4.4.2 und 4.4.3. Da die auf dem Prozessfähigkeitsindex basierende Klassifikation keiner Kategorie eindeutig zugeteilt werden kann, wird diese im anschließenden Abschnitt 4.4.4 besprochen. Der Algorithmus zum Erlernen des von vielen Klassifikatoren für ihre Entscheidung verwendeten Schwellenwerts wird am Ende dieses Kapitels im Abschnitt 4.4.5 vorgestellt.

Im ersten Abschnitt 4.4.1 finden sich drei Klassifikatoren, die als Vergleichsbasis zu den anderen Verfahren dienen sollen. Diese Art von Klassifikatoren wird hier als *triviale Klassifikatoren* bezeichnet. Die unter diese Kategorie fallenden Methoden verwenden für ihre Entscheidungen keine Informationen von dem aktuell vermessenen Bauteil. Sie verwenden außerdem so gut wie kein von der Lernphase gesammeltes Wissen.

Die so genannten *statischen Klassifikatoren* werden im Abschnitt 4.4.2 präsentiert. Alle im diesen Teil aufgeführten Techniken zeichnen sich dadurch aus, dass sie sowohl die während des Lernvorgangs angesammelten Erkenntnisse, als auch die auf dem aktuell untersuchten Bauteil verfügbaren Daten aktiv einsetzen, um zu einer Entscheidung zu kommen. Für jede neue Entscheidung sind jedoch die Voraussetzungen für beide möglichen Klassen gleich - die geschichtliche Entwicklung der letzten Messungen wird hier nicht berücksichtigt.

Erst das letzte, hier als *dynamisches Klassifikator* bezeichnete Verfahren ist in der Lage, alle drei Arten von Informationen in seine Entscheidungen mit einfließen zu lassen. Bei dem, auf einem Hidden-Markov-Modell aufbauenden Klassifikator, wird sowohl das Wissen aus der Lernphase, die Daten vom aktuell analysierten Chip, als auch die Informationen, welche sich der geschichtlichen

Entwicklung der Messungen früherer Bauteile entnehmen lassen, mit einbezogen. Die Modellierung des Algorithmus wird im Abschnitt 4.4.3.1 genauer beschrieben.

Es wird noch eine weitere Klassifikationstechnik in dieser Arbeit eingesetzt, die sich jedoch nicht in eine der vorher beschriebenen Kategorien einordnen lässt. Die Klassifikation auf Grundlage des Prozessfähigkeitsindex verwendet zwar die dem Lernvorgang entstammenden Informationen, als auch die geschichtliche Entwicklung der letzten k Messungen. Die auf dem aktuell analysierten Bauteil gesammelten Informationen werden jedoch nicht, für die Klassifikation dieses Bauteils, eingesetzt. Die Implementierungsdetails dieses Verfahrens finden sich im Abschnitt 4.4.4.1.

Alle vorgestellten Verfahren haben außerdem die im Folgenden aufgeführten Gemeinsamkeiten:

Objekt der Entscheidung: Jeder Klassifikator entscheidet über *eine* als möglicherweise auslassbar markierte Gruppe, die ihrerseits aus $k \geq 1$ Tests besteht.

Verfügbare Daten: Die für die Klassifikation vorliegenden Daten sind n-dimensionale Merkmalsvektoren[3], welche, wie im Abschnitt 4.3 vorgestellt, aus allen, auf dem zur Zeit analysierten Bauteil, verfügbaren Informationen bestehen.

Aufgabenstellung: Die Aufgabenstellung erfordert von jedem Klassifikator, die Zuordnung eines ihm vorgelegten Merkmalsvektors zu einer der zwei möglichen Klassen (siehe auch 4.1.1):

Klasse C_R: Die Menge aller Vektoren, auf welchen die Ausführung der vorherzusagenden Gruppe nicht notwenig war bzw. für nicht nötig befunden wurde.

Klasse C_M: Die Menge aller Vektoren, auf welchen die Ausführung der vorherzusagenden Gruppe notwendig war bzw. für notwendig befunden wurde.

Klassifikation: Die Entscheidungsfindung vollzieht sich bei allen Klassifikatoren durch die Berechnung der Wahrscheinlichkeit für die Zugehörigkeit eines Merkmalsvektors zur Klasse C_R[4].

Zusätzliche Informationen: Sollte in einem Experiment ein Datenoptimierungsverfahren aus dem Abschnitt 4.3.2 eingesetzt werden, welches die Daten mit zusätzlichen Informationen versieht (siehe 4.3.2.1.1), so steht es jedem Klassifikationsalgorithmus frei die zusätzlichen Daten zu berücksichtigen.

[3] Die Klassifikatoren sehen in den Merkmalsvektoren lediglich die Vektoren eines Vektorraumes ohne dabei ihre Bedeutung als Messwerte integrierter Schaltkreisen zu erkennen. Dadurch ist es möglich die Klassifikationsalgorithmen ohne größere Anpassungen direkt aus der Literatur zu übernehmen.

[4] Häufig wird ein Schwellenwert zum Vergleich eingesetzt, der von den Klassifikatoren selbständig erlernt werden kann. Für weitere Details siehe auch Abschnitt 4.4.5

4.4.1 Triviale Klassifikationsverfahren

Die in diesem Abschnitt präsentierten Verfahren zeichnen sich dadurch aus, dass sie für die Klassifikation eines Merkmalsvektors keine Informationen über diesen benötigen. Die von diesen Klassifikationsalgorithmen verwendeten Informationen beinhalten höchstens die aus der vorhergehenden Lernphase gewonnenen Erkenntnisse. Alle hier aufgezählten Methoden eigenen sich jedoch, wie im Abschnitt 6.2 gezeigt wird, nicht für den Einsatz in einer produktiven Umgebung, sondern viel eher für eine Gegenüberstellung mit anderen Verfahren zur Bemessung des erzielbaren Fortschrittes.

4.4.1.1 Konstante Entscheidung

Die Entscheidung dieser Klassifikationstechnik wird vor deren Ausführung extern festgelegt. Dieser Algorithmus fällt, ungeachtet jeglicher Umstände, während des gesamten Experimentes stets die gleiche, vorgegebene Entscheidung.

Würde man durch diesen Klassifikator einen Tester veranlassen alle potenziell auslassbaren Gruppen einzusparen, erhielte man dadurch die maximale Anzahl aller übersehbaren Fehler. Nach diesem Experiment ließe es sich ebenso abwägen, wie hoch das Verhältnis der Kosten eines übersehenen, schlechten Bauteils gegenüber den Ersparnissen eines ausgelassenen Tests sein müsste, damit der Break-Even-Point erreicht ist. Durch die Ausführung *aller* Testgruppen, lässt es sich andererseits feststellen, wie hoch die aktuellen Testkosten der untersuchten Testssequenz auf den vorliegenden Daten sind.

4.4.1.2 Zufallsbasierte Klassifikator

Die Aufgabe dieses Algorithmus ist die Simulation eines unwissenden Entscheiders, der seine Entscheidungen auf gut Glück trifft. Jede neue Klassifikation wird auf Basis einer Zufallszahl vollzogen. Die Ergebnisse werden ebenfalls ausschließlich für Vergleichszwecke verwendet.

4.4.1.3 Bayes Klassifikation

Dieser nach dem englischen Mathematiker *Thomas Bayes* benannte Klassifikator [DHS00d], verwendet ein gewisses Maß an Wissen. Der in dieser Arbeit implementierte Fall sieht jedoch vor, dass alle eingesetzten Informationen sich ausschließlich über die Menge der trainierten Lerndaten und der daraus berechneten *a priori* Wahrscheinlichkeiten für die Entdeckung eines defekten Bauteils durch jede potenziell weglassbare Gruppe erstrecken.

Zur Laufzeit wird für jede vorherzusagende Gruppe überprüft, wie hoch die *a priori* Wahrscheinlichkeit für die Entdeckung eines Ausfalls durch diese Gruppe ist. Liegt die Wahrscheinlichkeit für einen Ausfall über der Wahrscheinlichkeit, dass kein Fehler erkannt wird, wird die Gruppe stets ausgeführt. Anderenfalls wird die Testgruppe auf jedem Bauteil ausgelassen und ihre Ausführungszeit wird eingespart.

Dieser Klassifikationsalgorithmus verdeutlicht eine nachvollziehbare Entscheidung eines Testingenieurs, der nach der Betrachtung eines voll ausgetesteten Beispieldatensatzes vor der Frage steht, welche Testgruppen er auf den folgenden Bauteilen einsparen könnte.

4.4.2 Statische Klassifikationsverfahren

Zu der in dieser Arbeit bezeichneten Kategorie der statischen Klassifikationsverfahren, werden, wie im vorigen Abschnitt mehrere Algorithmen gezählt. Diese Eingliederung umfasst Verfahren, welche sowohl die aus dem Lernvorgang gewonnenen Erkenntnisse, als auch die auf dem aktuell untersuchten Bauteil verfügbaren Daten aktiv einsetzen, um eine Klassifikation vorzunehmen. Die aufgezählten Techniken sind jedoch nicht in der Lage, die geschichtliche Entwicklung der letzten Messungen in ihren Entscheidungen zu berücksichtigen.

4.4.2.1 k Nächste Nachbarn Klassifikation

Das in dieser Arbeit eingesetzte k Nächste Nachbarn (KNN) Klassifikationsverfahren [Fuk90] zeichnet durch keine besonderen Abweichungen von der in der Literatur verwendeten Implementierung aus, nichtsdestotrotz werden im Folgenden einige Details der Umsetzung hervorgehoben.

Dank der im Abschnitt 4.3.1 vorgestellten Techniken, sind die zu klassifizierenden Merkmalsvektoren und die sich im Merkmalsraum befindenden Beispielvektoren bereits vom Rauschen und sinoiden Protzesschwankungen befreit. Aus diesem Grund wird versucht, möglichst klare Entscheidungen durch eine relativ klein gewählte Anzahl nächster Nachbarn herbeizuführen.

Die eingesetzte Abstandsmetrik ist stets das euklidische Abstandsmaß (siehe Gleichung 4.5), welches jedoch *drei* zusätzliche Gewichtungen erhalten kann.

Am besten lässt sich der Zweck solcher Gewichtungen am folgenden Beispiel erläutern: Im Beispiel soll angenommen werden, dass die drei zu einem Merkmalsvektor x gefundenen Nachbarn jeweils ganze Cluster repräsentieren, nämlich die Cluster c_1, c_2 und c_3.

- Der Vorteil durch die zusätzlichen Angabe des Verhältnisses beider Klassen C_M und C_R innerhalb eines Cluster zu einander (4.3.2.1.1) wird deutlich, wenn man annimmt, dass die beiden Cluster c_1 und c_2 zu 51 Prozent mit Vektoren der Klasse C_R gefüllt sind, während der Cluster c_3 zu 100 Prozent Vektoren der Klasse C_M beinhaltet. Würde man in diesem Fall eine oberflächliche Entscheidung (ohne Beachtung der Zusatzinformationen) fällen, so würde der Merkmalsvektor x zur Klasse C_R klassifiziert werden, denn zwei der drei Nachbarn gehören mehrheitlich der Klasse C_R an. Bei genauer Betrachtung der Zusatzdaten, würde die Entscheidung jedoch zu Gunsten der Klasse C_M kippen, denn die gefundenen Nachbarn sind im Durchschnitt zu 66 Prozent mit Vektoren der Klasse C_M gefüllt.

- Die Angabe der Anzahl der, von den Clustern abgedeckten Merkmalsvektoren (siehe ebenfalls 4.3.2.1.1), kann gleichermaßen die Klassifikation entscheidend beeinflussen. Würde man hier annehmen, dass c_1 1000 Vektoren des Merkmalsraums umfasst, c_2 100 Vektoren und c_3 nur 10, so wäre schnell klar, dass bei der Entscheidungsfindung der Hauptaugenmerk auf dem Cluster c_1 mit 1000 Vektoren lasten sollte.
- Der Abstand eines zu klassifizierenden Merkmalsraumvektors zu seinem Nachbarn kann ein Hinweis auf die Ähnlichkeit bzw. Vergleichbarkeit beider Vektoren sein. Ist also der Abstand von x zu seinen Nachbarn c_1 und c_2 groß, zu seinem Nachbarn c_3 jedoch klein, so soll sich die Klassifikation eher an dem näheren Nachbarn c_3 orientieren[5].

4.4.2.2 Multivariate Gaußklassifikation

Das nächste in dieser Arbeit angewandte Klassifikationsverfahren, setzt zwar die Normalverteilung der Messwerte voraus, erlaubt aber dadurch den Aufbau eines multivariaten Gaußklassifikators [DHS00d].

Der Klassifikator berechnet während der Lernphase, die für die Berechnung der Dichtefunktion p notwendigen Parameter: Den Mittelwertvektor μ und die $n \times n$ Kovarianzmatrix Σ. Mit n ist hier die Dimensionalität der Merkmalsvektoren angegeben. Die beiden statistischen Größen μ und Σ werden auf den zum Training verfügbaren Bauteilen für beide Klassen C_M und C_R separat bestimmt.

Die Dichtefunktion p ist definiert als:

$$p(x) := \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right) \tag{4.9}$$

Zur Laufzeit werden die erlernten Parameter verwendet um mit Hilfe der Dichtefunktion die Wahrscheinlichkeiten p_M und p_R für die Klassifikation eines analysierten Merkmalsvektors x zu den beiden Klassen C_M und C_R zu bestimmen.

Die Genauigkeit der Dichtefunktion könnte aber unter der zumeist hohen Dimensionalität der Merkmalsvektoren leiden. Aus diesem Grund wird die in dieser Arbeit verwendete Dichtefunktion etwas abgewandelt. Durch die Transformation des Wertebereiches der Funktion in den logarithmischen Bereich erreicht man ein auf dem Computer besser darstellbares numerisches Segment. Die eingesetzte, transformierte Dichtefunktion p' ist somit definiert durch:

$$p'(x) := -\frac{1}{2}\ln\left((2\pi)^n |\Sigma|\right) - \frac{1}{2}\left((x-\mu)^\top \Sigma^{-1}(x-\mu)\right) \tag{4.10}$$

Werden während der Datenoptimierung 4.3 Verfahren eingesetzt, die die Merkmalsvektoren mit Zusatzinformationen (4.3.2.1.1) kennzeichnen, so lassen sich diese Informationen bei der Berechnung des Mittelwertvektors und

[5] Diese Art der Zusatzinformation ist in dieser Arbeit nur für den k Nächste Nachbarn Algorithmus verfügbar. Sie betrifft allerdings alle mögliche Nachbarn eines zu klassifizierenden Merkmalsvektors - nicht nur Cluster.

der Kovarianzmatrix berücksichtigen. Auf diese Weise kann beispielsweise erreicht werden, dass ein Cluster, der viele Merkmalsvektoren bündelt, bei der Bestimmung des Mittelwertvektors stärker ins Gewicht fällt, als ein Cluster mit wenigen Merkmalsvektoren. Somit lässt sich die Präzision der Dichtefunktion noch weiter steigern.

4.4.2.3 Multilayer-Perzeptron

Das letzte in diesem Abschnitt vorgestellte Verfahren ist der Multilayer-Perzeptron [DHS00e]. Dieses vereinfachte neuronale Netz, welches in seinen Grundzügen bereits 1958 vom Psychologen und Informatiker *Frank Rosenblatt* vorgestellt wurde, erlaubt ein selbstständiges Erlernen der Lösung eines bestimmten Problems durch ausgiebiges Training.

Der Einsatz neuronaler Netzwerke wurde bereits in [Fei05] erprobt, scheiterte jedoch an einer unverhältnismäßig langen Laufzeit der eingesetzten Implementierung. Dies war der Grund, warum für diese Arbeit eine eigene Implementierung des Netzes entwickelt wurde. Die Entwicklung umfasst ein schnell arbeitendes Feed-Forward Netz ohne Abkürzungen, mit beliebiger Anzahl und Größe von versteckten Layern.

Der Aufbau der tatsächlich eingesetzten Netzwerke bestand zumeist aus einer versteckten Neuronenschicht. Die Anzahl der Eingangsneuronen entsprach jeweils der Dimensionalität der Merkmalsvektoren. Die Anzahl der Neuronen innerhalb der versteckten Schicht belief sich auf ein Fünftel der Eingabeschicht, jedoch auf mindestens drei. Die Ausgabeschicht, bestehend aus genau zwei Neuronen, war dazu konzipiert, bei einer klaren Entscheidung, auf einem der Neuronen eine 1 und auf dem anderen Neuron jeweils eine -1 zurückgeben, abhängig von der Klasse des angelegten Merkmalsvektors. Zur Vereinfachung der Implementierung wurde zu jeder Schicht außer der Ausgabeschicht ein zusätzliches *bias* Neuron hinzugefügt. Auf diese Weise soll das Erlernen von Schwellwerten auf das Erlernen von Verbindungsgewichtungen transformiert werden.

Das Kernstück eines jeden neuronalen Netzes dürfte zweifelsohne das Erlernen der Lösung der dem Netzwerk gestellten Aufgabe sein. Die Details der dafür eingesetzten Implementierung werden nachfolgend aufgelistet.

1. Das Training der Lerndaten erfolgt in mindestens 10000 Iterationen[6] unter Einsatz der Backpropagation-Methode. Es werden keine Beschleunigungsverfahren, wie das *Quickprop* [DHS00e] verwendet.

2. Als Transferfunktion dient die, in der Literatur übliche, differenzierbare Sigmoidfunktion *Tangens Hyperbolicus* $f(x) = \tanh(x)$.

3. Zur Herbeiführung eines stabilen Endzustandes wurde die Backpropagation mit einer Prozesstemperatur versehen. Die Temperatur wird im Laufe

[6] Zum einfacheren Verständnis, verwendet diese Arbeit den Begriff „Iteration" anstatt des sonst für neuronale Netze üblichen Begriffs „Epoche". Die beiden Bezeichnungen sind in diesem Abschnitt gleichzusetzen.

des Trainings kontinuierlich, logarithmisch gesenkt und mit ihr auch die Lernrate Δw_{ij}.

4. Beträgt die Anzahl der Merkmalsvektoren einer Klasse mehr als das Vierfache der anderen Klasse, so werden die Vektoren der kleineren Klasse vervielfacht, bis ihre Menge mindestens ein Viertel der größeren Klasse erreicht.

5. Um das Einlernen einer bestimmten Reihenfolge durch den Perzeptron zu vermeiden, werden die Trainingsdaten zwischen zwei aufeinander folgenden Iterationen zufällig sortiert.

6. Der Lernfortschritt wird mit Hilfe vorher aussortierter, nicht gelernter Daten nach jeder Iteration kontrolliert.

Zur Beschleunigung des Lernprozesses wurde eine zuschaltbare Heuristik entwickelt. Die Idee hinter der Heuristik, ist die Beobachtung des neuronales Netzes über einen bestimmten Zeitraum von k Iterationen. Nach jeder k-ten Iteration wird der Netzzustand[7] mit der geringsten Fehlerquote ausgesucht und im neuronalen Netz wiederhergestellt. Von diesem optimalen Zustand ausgehend werden die nächsten k Iterationen beobachtet, etc.

In der Abbildung 4.6 findet sich ein Vergleich des Lernfortschrittes mit und ohne der vorgestellten Heuristik.

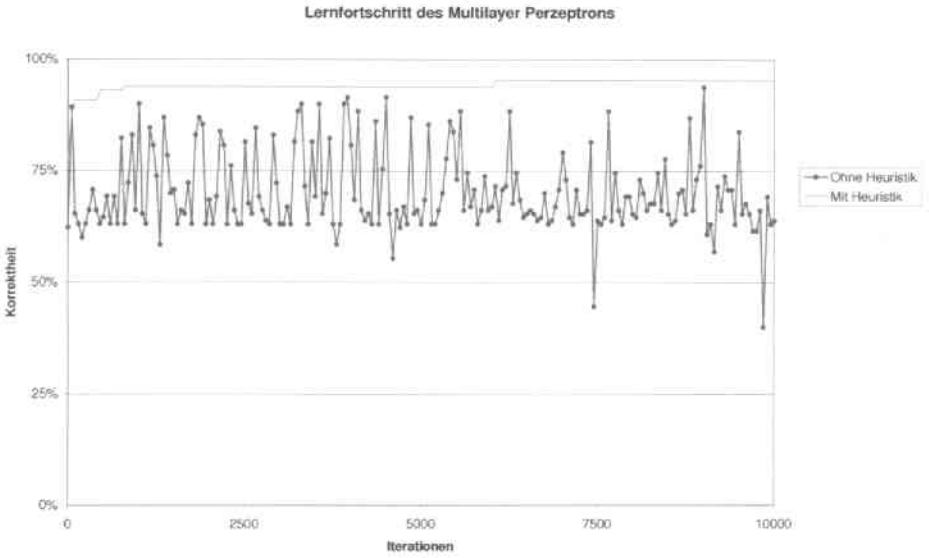


Abbildung 4.6: Vergleich des Lernfortschrittes eines Multilayer-Perzeptrons mit und ohne Einsatz der vorgestellten Heuristik

Zur Laufzeit wird mit Hilfe des Feed-Forward Algorithmus, für einen angelegten Merkmalsvektor, die Ausgabe der beiden Neuronen der Ausgabeschicht bestimmt. Die gemessene Differenz gilt als Vorhersage einer bestimmten Klassenzugehörigkeit.

[7] Der Netzzustand soll durch die Matrix der Verbindungsgewichtungen zwischen den einzelnen Neuronen eindeutig definiert sein.

4.4.3 Dynamische Klassifikationsverfahren

In diesem Kapitel wird die einzige Technik vorgestellt, welche im Sinne dieser Arbeit als dynamisch bezeichnet werden kann. Der auf einem Hidden-Markov-Modell aufbauende Klassifikator, ist in der Lage, sowohl die Erkenntnisse der Lernphase, die Messwerte des aktuell vermessenen Bauteils, als auch die zeitliche Entwicklung der Messungen früherer Bauteile in seine Entscheidungen mit einzubeziehen. Die Beobachtung der zeitlichen Entwicklung früherer Messungen ist der Schlüssel zur Modellierung der sich ständig verändernden Produktionszustände, wie sie im Abschnitt 3.1.1 angenommen werden.

4.4.3.1 Hidden-Markov-Modell basierter Entscheider

Ein Hidden-Markov-Modell (HMM) [DHS00f] ist ein Konzept, welches in der Lage ist, zwei stochastische Zufallsprozesse zu beschreiben, von welchen sich aber nur ein Prozess beobachten lässt. Anhand der ausschließlichen Beobachtung der Emissionen des sichtbaren Prozesses, ist es möglich den wahrscheinlichsten Zustand des verborgenen Prozesses zu ermitteln.

Transformiert auf die in dieser Arbeit untersuchte Problematik bedeutet dies gleichzeitig, dass mit Hilfe eines Hidden-Markov-Modells und der Beobachtung der Messwerte analysierter Chips, sich, die im Hintergrund der Herstellung ablaufenden Veränderungen (Zustände und Zustandsübergänge) modellieren und als zusätzliche Information für die Klassifikation verwenden lassen.

Ein weiterer großer Vorteil von Hidden-Markov-Modellen ist die Tatsache, dass die einzelnen verborgenen Zustände nicht näher bekannt zu sein brauchen. Festgelegt werden muss lediglich ihre Anzahl. Unter Einsatz des Forwad-Backward Algorithmus sowie eines bestimmten Zustandsübergangsschemas, können so die Übergangswahrscheinlichkeiten zwischen den einzelnen verborgenen Zuständen, sowie die Emissionswahrscheinlichkeiten gemachter Beobachtungen für jeden Zustand berechnet werden.

Die Konstruktion eines Hidden-Markov-Modells wird in dieser Arbeit folgendermaßen vorgenommen:

1. Ein verborgener Zustand des Hidden-Markov-Modells soll einen möglichen *nicht näher spezifizierten* Zustand der Produktion darstellen. Wie bereits im Abschnitt 3.2 gezeigt, kann ein solcher Zustand alle möglichen Bedingungen repräsentieren.

   - Zustand der Siliziumkristall-Schmelzanlagen, Belichtungmaschinen: {*Frisch eingeschalten, Normaler Betrieb, Hohe Belastung*}
   - Zustand der Produktionshalle: {*Temperatur, Feuchtigkeit, Luftreinheit*}
   - Zustand der Mitarbeiter: {*Arbeitszeit, Schicht, Wochentag*}

   Alle aufgezählten Bedingungen, ebenso, wie viele weitere, können einen gewissen Einfluss auf die Qualität oder bestimmte Eigenschaften des hergestellten Bauteils haben. Zu unterschiedlichen Produktionsphasen können

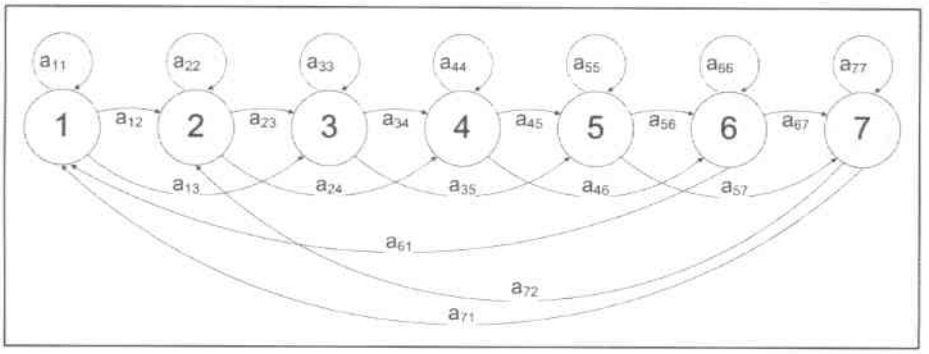


Abbildung 4.7: Das implementierte Bakis-Übergangsschema des Hidden-Markov-Modells

verschiedene Produktstadien betroffen sein: ganze Kristallstäbe, einzelne Wafer oder auch nur einzelne Chips.

2. Die beobachtbaren Emissionen eines verborgenen Zustandes sind Messwerte eines analysierten Bauteils, bzw. der daraus entstandenen Merkmalsvektoren.

3. Jeder der HMM-Zustände besitzt einen für ihn optimalen Klassifikator. Der optimale Klassifikator K^* eines Zustands q sei definiert, als der Klassifikator K, welcher den, vom Zustand q am wahrscheinlichsten emittierten Merkmalsvektor x, mit der geringsten Fehlerwahrscheinlichkeit klassifizieren kann. Siehe dazu auch Abbildungen 4.8 und 4.9.

4. Die Gestaltung der Zustandsübergänge folgt, der in der Vorlesung *Sprachliche Mensch-Maschine-Kommunikation*[8] ausgesprochenen Empfehlung[9]. Es wird das Bakis-Übergangsschema verwendet, die Topologie wird allerdings zu einem Ring verbunden, um damit eine beliebige Anzahl von Zustandsübergängen und Emissionsbeobachtungen zu ermöglichen (siehe Abbildung 4.7).

Wie im Kapitel 6 zu sehen sein wird, eignet sich von allen vorgestellten Verfahren ein auf Hidden-Markov-Modellen basierende Klassifikator am besten, um die Chiptestpläne zu optimieren.

4.4.4 Sonstige Verfahren

Diese Arbeit setzt eine weitere Klassifikationstechnik ein, die jedoch nicht in eine, der vorher beschriebenen Kategorien zugeordnet werden kann: *Prozessfähigkeitsindex basierte Klassifikation*. Bei diesem Verfahren werden zwar,

[8] Die Vorlesung „Sprachliche Mensch-Maschine-Kommunikation" wurde gehalten vom Lehrstuhl Professor Dr. A. Waibel am Institut für Theoretische Informatik, an der Universität Karlsruhe (TH).

[9] Es wurden zwar auch weitere Topologien des Hidden-Markov-Modells ausprobiert, aber es konnten keine nennenswerten Unterschiede in den Klassifikationsergebnissen festgestellt werden.

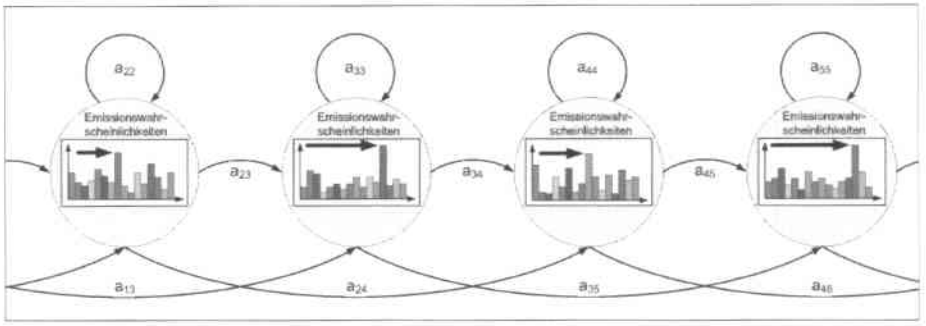


Abbildung 4.8: Das implementierte Bakis-Übergangsschema des Hidden-Markov-Modells im Detail mit sichtbaren Emissionswahrscheinlichkeiten der Merkmalsvektoren x_i

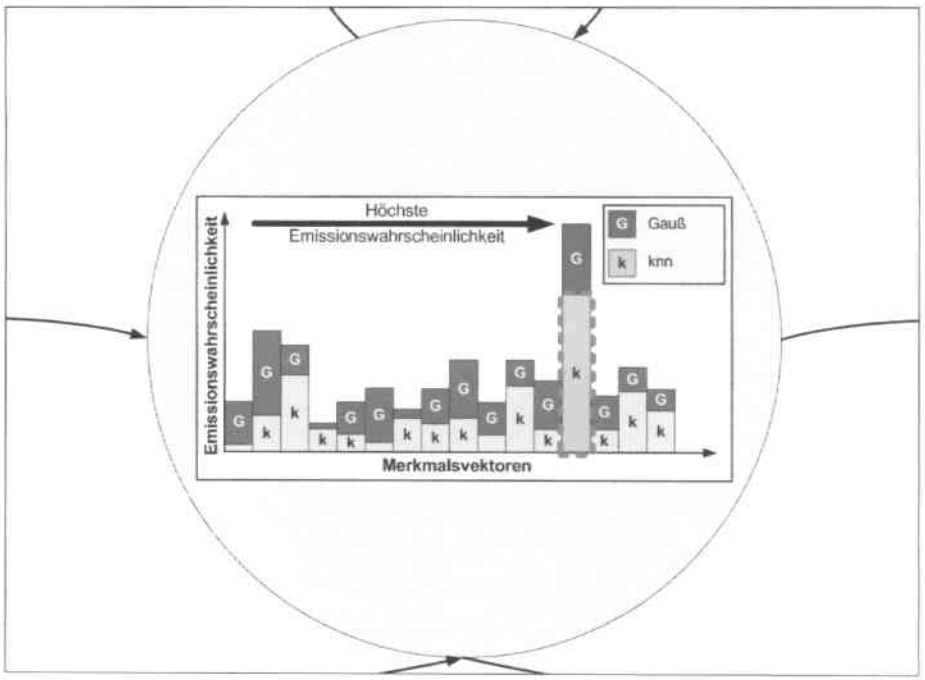


Abbildung 4.9: Der in der Arbeit implementierte Aufbau eines verborgenen Zustandes des Hidden-Markov-Modells: Die Emissionswahrscheinlichkeiten der Merkmalsvektoren x_i mit eingezeichneter Vorhersagequalität zweier Klassifikationsverfahren sichtbar dargestellt

sowohl die dem Lernvorgang entnommenen Informationen, als auch die Entwicklung der letzten l Messungen betrachtet. Die, auf dem aktuell analysierten Bauteil gesammelten Informationen kommen jedoch während der Klassifikation dieses Bauteils bzw. seines Merkmalsvektors nicht zum Einsatz.

4.4.4.1 Prozessfähigkeitsindex C_{pk} basierte Klassifikation

Das in diesem Abschnitt beschriebene, auf dem Prozessfähigkeitsindex C_{pk} aufbauende Klassifikationsverfahren repräsentiert den aktuellen Stand der Technik (Abs. 2.2). Diese Klassifikationstechnik ist gleichzeitig auch die *einzige* Methode der Testkostenoptimierung zur Laufzeit des Testers, welche von der Halbleiterindustrie tatsächlich erprobt wird [Swi04]. Genau aus diesem Grund wird die Technik von der vorliegenden Arbeit nachgebaut, um einen Vergleich mit den aus der Spracherkennung bekannten Klassifikationsalgorithmen durchführen zu können.

Die auf dem Prozessfähigkeitsindex C_{pk} aufbauende Klassifikatoren sind die bisher Dies ist auch der Grund, weshalb trotz mehrerer im Abschnitt 2.2 aufgeführter Schwachstellen dieses Klassifikationsverfahren in dieser Arbeit eingesetzt wird.

Das Grundprinzip dieser Technik setzt auf dem Prozessfähigkeitsindex auf. Bei der Berechnung des Index wird die Standardabweichung und der Abstand des Mittelwerts eines Tests zu seinen Toleranzgrenzen in Relation gesetzt (siehe Gleichung 2.1). Als direkte Folgerung der Definition dieses Index, ergibt sich die Feststellung, dass bei einem normalverteilten Test mit einem steigenden C_{pk} Wert die Fehlerwahrscheinlichkeit sinkt [FLJ91].

Die Implementierung des Algorithmus soll im Folgenden kurz erläutert werden:

1. Der Prozessfähigkeitsindex aller Tests bzw. Gruppen wird auf allen gelernten Daten ermittelt.
2. Alle Testgruppen bzw. Tests mit einem C_{pk} Wert höher oder gleich 2.0 werden sogleich für die Auslassung markiert. Die Begründung dafür ist in der statistisch gesehen, geringen Wahrscheinlichkeit für einen Ausfall dieses Tests zu suchen. Laut [FLJ91] liegt diese Wahrscheinlichkeit bei 0.0000099 Prozent.
3. Die zur Einsparung markierten Gruppen werden auf den folgenden k Bauteilen nicht ausgeführt.
4. Nach diesen k Bauteilen werden wieder alle Testgruppen für die nächsten l Bauteile zugeschaltet (bzw. ausgeführt).
5. Während der Ausführung aller Testgruppen auf den l Bauteilen werden die C_{pk} Werte aller Gruppen aktualisiert.
6. Das Verfahren wird ab Schritt 2 wiederholt, bis alle Bauteile abgearbeitet sind.

Die in dieser Arbeit eingesetzten Werte für k und l entsprechen den in der Praxis häufig erprobten Werten $k = 50$ und $l = 50$.

Es soll noch einmal darauf hingewiesen werden, dass es zahlreiche Gründe (siehe 2.2) existieren, die dafür sprechen, diese Technik nicht in der Praxis einzusetzen. Die beiden größten Schwachstellen des Verfahrens sind im Weiteren genannt:

1. Der Prozessfähigkeitsindex beschreibt lediglich die beobachteten Daten. Treten auf den k Bauteilen, auf welchen ein Test eingespart wird, Fehler auf, so gibt es keine Gesetzmäßigkeit, die das Auftreten von Fehlern auch auf den l vermessenen Chips erzwingt.

2. Der C_{pk} Index stellt keine Analyse des Fehlerverhaltens dar. Eine Analyse des Fehlerverhaltens des einzusparenden Tests ist jedoch eine Mindestvoraussetzung, um Schätzungen über den Zeitpunkt bzw. die Umstände, in welchen der Test einen Fehler feststellt, machen zu können.

4.4.5 Schwellenwerte

Alle im Abschnitt 4.4 vorgestellten Klassifikatoren entscheiden über die Zugehörigkeit eines Merkmalsvektors zu einer der beiden möglichen Klassen C_M oder C_R. Die Entscheidung wird in den meisten Fällen[10] auf die unten beschriebene Art ermittelt:

1. Als erstes werden die beiden Koeffizienten e_M und e_R berechnet, die als Schätzungen (*estimations*) für die Zugehörigkeit des analysierten Merkmalsvektors x zu den Klassen C_M und C_R verwendet werden. Hierbei sind die Koeffizienten e_M und e_R, in Abhängigkeit von den ermittelten Wahrscheinlichkeiten p_M und p_R für die Zugehörigkeit des Merkmalsvektors x zu den beiden Klassen C_M und C_R, wie folgt definiert:

$$e_i = w(x)p_i(x) \text{ und } i \in \{M, R\}$$

   Der Gewichtungskoeffizient w ist abhängig von x und stellt die, aus den im Abschnitt 4.3.2.1.1 beschriebenen Zusatzinformationen, gewonnene Gewichtung des Merkmalsvektors x dar. Sind die Zusatzinformationen bedingt durch das verwendete Datenoptimierungsverfahren nicht vorhanden, wird w stets der Wert 1.0 zugewiesen.

2. Anschließend wird der Quotient q_R berechnet. Dieser ist definiert durch:

$$q_R = \frac{e_R}{e_M + e_R}$$

   Er entspricht also der Schätzung c_R für Zugehörigkeit des Merkmalsvektors x zu der Klasse C_R, normiert durch die Summe der Schätzungen, dass der Merkmalsvektor zur Klasse C_R *oder* Klasse C_M zugeordnet werden kann.

[10] Triviale Klassifikatoren sowie der auf dem Prozessfähigkeitsindex C_{pk} basierender Klassifikator berechnen ihre Vorhersagen auf anderen Wegen.

3. Anschließend wird der Quotient q_R mit einem Schwellenwert s verglichen.

   Fall $q_R > s$**:** Ist die für die Klasse C_R sprechende Wahrscheinlichkeit größer als der Schwellenwert s, so entscheidet sich der Klassifikator für die Klassifikation von x zur Klasse C_R.

   Fall $q_R \leq s$**:** Ist der ermittelte Quotient kleiner als oder gleich dem Schwellenwert s, so entscheidet sich das Klassifikationsverfahren für die Zuordnung des Merkmalsvektors x zur Klasse C_M.

Die offen verbleibende Frage, ist die nach der Festlegung bzw. Bestimmung des geeigneten Grenzwertes s. Die vorliegende Arbeit erlaubt hierbei zwei Alternativen:

- Die manuelle Festsetzung des Schwellenwertes auf einen beliebigen Wert innerhalb des Intervalls $[0, 1]$[11].
- Die Wahl der geeigneten Position des Grenzwertes durch selbständiges, automatisches Training.

Automatische Bestimmung des Schwellenwerts

Zur Minimierung der Anzahl möglicher Fehlentscheidung besteht für jeden Klassifikator die Möglichkeit den für seinen Merkmalsraum besonders geeigneten Grenzwert automatisch berechnen zu lassen. Dafür werden alle zum Lernen verfügbaren Trainingsdaten mit dem nachfolgend beschriebenen Algorithmus untersucht:

1. Der Ausgangswert für den Grenzwert s wird auf 1.0 gesetzt.
2. Für jeden Merkmalsvektor x wird, wie oben beschrieben, mit Hilfe des Klassifikationsalgorithmus der Quotient q_R ermittelt.
3. Korrektur:

   Fall $q_R > 0.5$ **und** $x \in C_M$**:** Der Schwellenwert s wird nach *oben* korrigiert.

   Fall $q_R \leq 0.5$ **und** $x \in C_R$**:** Der Schwellenwert s muss nach *unten* korrigiert werden.

   Die Korrekturschrittweite ist direkt von der Distanz $|s - q_R|$ abhängig und kann zusätzlich mit Hilfe einer Temperatur beschränkt werden.

Um eine sichere Stabilität dieser Technik zu gewährleisten, erhält sie eine Temperatur, die während des Voranschreitens des Lernprozesses gesenkt wird und damit auch die maximale Korrekturschrittweite begrenzt.

Auf Grund der Kostendifferenz unterschiedlicher Typen von Fehlern, welche bereits im Abschnitt 2.3 und 3.1.1 beschrieben wurde, lassen sich zudem, die

[11] Werte nahe der Null lassen den Klassifikator häufiger Merkmalsvektoren der Klasse C_R zuordnen, das heißt es werden mehr Testgruppen eingespart. Werte in der Nähe der Eins bewirken hingegen eine häufigere Zuordnung zur Klasse C_M. Dadurch würden mehr Testgruppen ausgeführt werden.

Auswirkungen des Temperaturrückgangs für die verschiedenen Korrekturrichtungen unterschiedlich handhaben. So kann die Temperatur für die Korrekturen des Schwellenwertes nach unten, beispielsweise nur halb so schnell fallen, wie für die Korrektur des Schwellenwertes nach oben. Diese Konfigurationsmöglichkeit des Temperaturrückgangs ermöglicht vor allem bei Merkmalsräumen, in welchen die Vektoren beider Klassen stark vermischt sind, eine häufigere Ausführung einer bestimmten Testgruppe, um damit die teureren Fehler gezielt zu vermeiden.

Kapitel 5

Arbeitsumgebung

Dieses Kapitel gibt einen kurzen Überblick über die Arbeitsumgebung dieser Arbeit, sowie die für die Durchführung der Experimente im Kapitel 6 verwendeten Daten.

5.1 Analysierte Daten

Das Ziel dieser Arbeit ist die Präsentation eines Optimierungsverfahrens, mit dessen Hilfe sich die stetig ansteigenden Testkosten der Halbleiterherstellung wieder senken lassen. Um die tatsächliche Anwendbarkeit des Verfahrens in der Praxis zu belegen, basieren die im Kapitel 6 durchgeführten Experimenten auf wirklichen, *nicht* synthetisch generierten Messdaten integrierter Schaltkreise oder ganzen Boards. Die original Messungen sind beim Testen von Chips oder Boards, während der dritten Testphase (siehe 2.1) angefallen.

Die Daten entstammen drei verschiedenen, für Mixed-Signal Chips repräsentativen Produkten namhafter weltweit agierender Konzerne, welche sich entweder exklusiv, oder partiell als Halbleiterhersteller engagieren. Zur besseren Unterscheidung der Daten, wurden diese in drei Datensätze, entsprechend ihren Quellen unterteilt. Daraus ergaben sich drei folgende Datensätze:

Datensatz A: Ein vollständiges Los, bestehend aus 22 Wafern mit jeweils ca. 3200 Bauteilen und 95 Tests pro Bauteil. Insgesamt wurden also die Daten von 70670 Chips in etwa 6.7 Mio. Einzelmessungen gesammelt. Die Anzahl fehlerhafter Bauteile lag bei 14753 Stück, was einer Ausbeute von 79.1 Prozent gleichkommt.

Datensatz B: Mehrere Lose, insgesamt 173 Wafer mit jeweils durchschnittlich 930 Chips und jeweils 97 Tests pro Chip. Zusammengenommen befinden sich im Datensatz 161606 Bauteile, davon nur 7627 defekt. Dies entspricht einer Ausbeute von 95.2 Prozent und 15.7 Mio. Einzelmessungen.

Datensatz C: Ein Satz von 17647 nacheinander vermessenen integrierten Schaltkreisen auf einer unbekannten Menge von Wafern. Jeder Chip wurde 454 verschiedenen Tests unterzogen, womit sich etwas mehr als 8 Mio. Einzelmessungen ergeben. Die Ausbeute dieser Bauteile betrug 78.1 Prozent und entspricht 3870 defekten Bauteilen.

Die Daten wurden von der Firma optimiSE GmbH in Karlsruhe für diese Arbeit zur Verfügung gestellt.

Zur Durchführung der im Kapitel 6 folgenden Experimente, werden die Datensätze in eine Trainings- und eine Evaluationsdatenmenge gemäß eines bestimmten Stichtages unterteilt. Die Trainingsdaten wurden vor dem gewählten Stichtag gemessen, die Evaluationsdaten danach. Kein Bauteil darf gleichzeitig in beiden Mengen vorkommen. Die Aufteilung der Datensätze ist in der Tabelle 5.1 beschrieben.

Datensatz	Trainingsdaten	Evaluationsdaten	Gesamt
A	51 402 (72.7%)	19 268 (27.3%)	70 670 (100%)
B	91 859 (56.8%)	69 747 (43.2%)	161 606 (100%)
C	9 356 (53.0%)	8 291 (47.0%)	17 647 (100%)

Tabelle 5.1: Aufteilung der verfügbaren Messdaten in Trainings- und Evaluationsdaten

Wie gerade dargelegt, ist die Ausbeute der verfügbaren Daten relativ hoch (> 78 Prozent), beim Datensatz B liegt diese sogar über 95 Prozent. Somit ist auch die Wahrscheinlichkeit für einen Ausfall auf diesen Daten relativ gering. Dennoch ist es zu riskant einen Test erst dann als kritisch einzustufen, wenn dessen Messung auf einem Bauteil bereits jenseits des Toleranzintervalls liegt. Aus diesem Grund werden in dieser Arbeit alle Tests schon früher als *ausführenswert* betrachtet. Da sich die Toleranzgrenzen verschiedener Tests innerhalb der Testsequenz sehr stark unterscheiden können, werden die für die Entscheidung ausschlaggebenden Grenzen für jeden Test einzeln angepasst. Zur Vereinheitlichung wird das Toleranzintervall derart gewählt, dass jeder Test einen C_{pk} Wert (Gleichung 2.1) von 0.5 anstrebt[1].

Durch die durchgeführte *Verschärfung* der Toleranzgrenzen entstehen somit keine neuen Escapes.

Die verschärfte Betrachtung der Toleranzgrenzen gilt selbstverständlich für jeden Test, für alle Datensätze und sowohl während der Lernphase, als *auch in der darauf folgenden Evaluationsphase.*

5.2 Programmumgebung

Die Umgebung dieser Arbeit bestand hauptsächlich aus mehreren von der Firma optimiSE GmbH, Karlsruhe zur Verfügung gestellten Schnittstellen zum Einlesen unterschiedlicher Messdaten-Formate, unter anderem auch des bereits erwähnten Standard-Test-Data-Formats.

[1] Bedingt durch die zum Teil sehr unterschiedlichen Größen der Standardabweichung und der Feststellung, dass nicht alle Tests - beispielsweise funktionsbedingt - normalverteilte Messungen ermitteln können, kann der von einem Test erreichte C_{pk} Wert, von dem angestrebten Wert 0.5 im Einzelfall erheblich abweichen.

Kapitel 6

Experimente

Dieses Kapitel gibt eine Übersicht über die in dieser Arbeit durchgeführten Experimentreihen. Wie im Kapitel 4 bereits gesehen, implementiert die vorliegende Arbeit eine Vielzahl von Klassifikatoren und weiteren Datenoptimierungsverfahren, die in zahlreichen Kombination erprobt wurden. Aufgrund der großen Anzahl an untersuchten Kombinationsmöglichkeiten, präsentiert dieses Kapitel nur eine Auswahl aller durchgeführten Versuche.

Nach der Skizzierung des grundlegenden Experimentaufbaus im Abschnitt 6.1, werden im restlichen Teil dieses Kapitels die in der Grafik 6.1 abgebildeten Experimente vorgestellt.

6.1 Experimentaufbau

In diesem Abschnitt wird die Vorgehensweise kurz erläutert, mit welcher die in diesem Kapitel vorgestellten Experimente durchgeführt wurden. Für weitere Informationen sei auf den Abschnitt 3.3 verwiesen.

Lernphase

Vor dem Beginn der Lernphase werden alle zur Verfügung stehenden Trainingsdaten eingelesen (4.2.1). Die eingelesene Testsequenz wird in einzelne Tests unterteilt, welche wiederum zu Gruppen gebündelt werden (4.2.2). Danach erfolgt eine eingehende statistische Analyse, sowie eine Analyse des Informationsgehalts (4.2.3). Aufbauend auf den Ergebnissen der Informationsgehaltsanalyse, angereichert mit weiteren statistischen Daten (4.2.3.1), werden bestimmte Gruppen als mögliche Kandidaten zum Einsparen vorgeschlagen.

Im nächsten Schritt, wird für jede möglicherweise auslassbare Gruppe g_R (*redundant*) ein Klassifikator (4.4) zur Verfügung gestellt. Jeder Klassifikator wird mit den bereits auf dem aktuellen Bauteil ausgeführten Messungen[1] und

[1] Die Messungen sind durch eines oder mehrere im Abschnitt 4.3 vorgestellte Verfahren vom statischen Rauschen bereinigt und normiert.

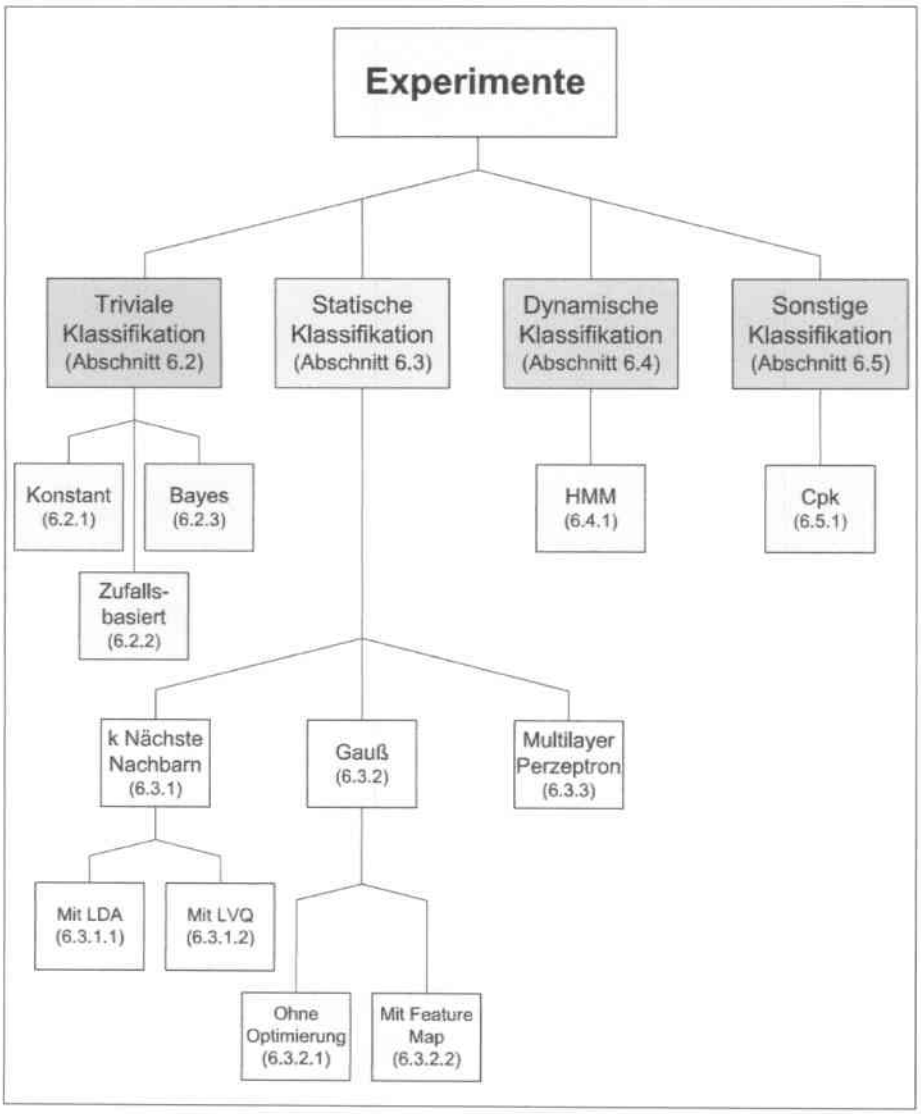


Abbildung 6.1: Alle im Kapitel 6 vorgestellten Experimente.

dem Ergebnis der vorherzusagenden Gruppe g_R trainiert.

Evaluationsphase

Während dieser Phase werden die Evaluationsdaten einzeln eingelesen. Die Testsequenz jedes Bauteils wird der Reihe nach abgearbeitet, wobei die Gruppenzugehörigkeit der Tests strikt beachtet wird. Sobald die Abarbeitung eine Gruppe g_R erreicht, welche als ein möglicher Kandidat zum Auslassen markiert ist, wird für diese der entsprechende Klassifikator ausgesucht.

Dem spezifischen Klassifikator werden alle, auf dem Bauteil bisher angesammelten Daten in Form eines n-dimensionalen Merkmalsvektors überreicht und eine Entscheidung über das weitere Vorgehen gefordert.

Der Klassifikator wertet intern den ihm übergebenen Merkmalsvektor aus und versucht diesen einer der beiden Klassen C_M und C_R zuzuordnen. Das Ergebnis dieser Zuordnung ist auch gleichzeitig die Entscheidung für das Einsparen oder Ausführen der vorherzusagenden Gruppe g_R. Wird diese Gruppe als ausführenswert klassifiziert, so stehen die Messdaten dieser Gruppe auf dem aktuellen Bauteil für andere, in der Testsequenz folgenden Entscheidungen zur Verfügung. Die getroffene Entscheidung wird von einer Überwachungsinstanz auf ihre Richtigkeit überprüft und vermerkt.

Das Einlesen der Testsequenz wird anschließend fortgesetzt, bis entweder ihr Ende erreicht ist, oder wieder über die Ausführung einer anderen Gruppe g'_R entschieden werden muss. Ist das Ende der Testsequenz erreicht, wird ein neues Bauteil eingelesen, bis schließlich alle verfügbaren Bauteile abgearbeitet sind. Daraufhin wird ein Report mit Diagrammen erstellt, welche sich in den folgenden Auswertungen einzelner Klassifikatoren finden.

6.2 Triviale Klassifikatoren

Dieser Abschnitt präsentiert einige ausgewählte Ergebnisse von Klassifikationsverfahren, die in dieser Arbeit als *triviale Klassifikatoren* bezeichnet werden. Diese Art von Klassifikatoren ist unter Abschnitt 4.4.1 beschrieben und unterscheidet sich von den restlichen Klassifikationstechniken dadurch, dass sie mit sehr wenig bis gar keinem Wissen auskommt. Falls diese Klassifikatorenart überhaupt Informationen verwendet, dann entstammen diese dem Lernvorgang. Damit wird die Entscheidung über das Ausführen oder Einsparen einer Testgruppe vor der eigentlichen Evaluationsphase gefällt. Aus diesem Grund kommen bei diesen Verfahren auch *keine* Datenoptimierungsverfahren aus dem Abschnitt 4.3 zum Einsatz, da diese keinen Einfluss auf die Vorhersage der Klassifikationsalgorithmen üben können.

Wie den folgenden Experimenten zu entnehmen sein wird, eignen sich die in diesem Abschnitt aufgezählten Klassifikationsverfahren ausdrücklich *nicht* für einen Einsatz in der Praxis und werden hier lediglich für Vergleichszwecke

hinzugezogen. Nach einer bestimmten Festlegung vor der Evaluationsphase ist es den Klassifikatoren während der Vorhersage nicht mehr möglich neu auftauchende Probleme zu erkennen.

6.2.1 Konstante Entscheidung

Ein konstanter Entscheider (siehe 4.4.1.1) wurde auf den Datensätzen eingesetzt, um zu klären, wie viele schlechte Messwerte unentdeckt bleiben, wenn man alle Testgruppen auf allen Bauteilen stets einspart. Als geeignete Datenbasis wurde in diesem Fall der Datensatz C gewählt. Die Ergebnisse der Klassifikation sind in den folgenden Diagrammen 6.2 und 6.3 zu sehen.

Im Diagramm 6.2 lässt sich die Gesamtzahl aller eingesparten Testausführungen mit der Anzahl unentdeckter Fehler in Relation setzen. Es wurde bereits festgestellt, dass dieses Verfahren so konzipiert wurde, dass es alle potenziellen Testausführungen stets einspart. Es verwundert daher nicht, dass die Anzahl eingesparter Testausführungen bei diesem Verfahren maximal ist. Keine andere Klassifikationstechnik kann mehr Tests einsparen, aber auch keine Technik kann einen derart hohen Anteil an unentdeckten, schlechten Bauteilen (*Escapes*) aufweisen.

Es ist zu beachten, dass die in diesem Diagrammtyp[2] eingesetzte Skala logarithmisch ist. Die Idee dahinter, liegt in der Betonung der Veränderungen der, vor allem in der Praxis besonders relevanten Anzahl an unentdeckten, schlechten Bauteilen.

Der folgende Diagrammtyp (Abbildung 6.3) verwendet für jeden vorhergesagten Test einen eigenen Balken. Die Reihenfolge der Tests entspricht genau der zeitlichen Ausführungsreihenfolge der Tests innerhalb der Testsequenz. Die *gesamte* (rot und grün) Höhe des Balkens gibt für jeden Test den *gesamten* Anteil der, bei seiner Vorhersage gemachten Fehler wider. Zusätzlich wird bei diesem Diagramtyp nach der Art der gemachten Fehler unterschieden. Der *rote* Anteil des Balkens beschreibt die Quote der übersehenen, schlechten Bauteile. Der Anteil des *grünen* Balkens hingegen, stellt die nicht zwingend notwendigen Testausführungen dar.

Der Grafik 6.3 lässt sich demnach entnehmen, dass die durchschnittliche Fehlerquote bei etwas über 17 Prozent liegt, obwohl die in Einzelfällen erreichte Fehlerquote davon merklich abweichen kann. Unnötige Testausführungen lassen sich in dieser Abbildung keine feststellen, da sie bereits definitionsbedingt ausgeschlossen sind.

In der Grafik 6.4 findet sich ein Überblick über den Aufbau aller in dieser Arbeit verwendeten *Precision-Recall* Statistiken [Rij79]. Die Abbildung zeigt ein Beispiel für die Klasse C_M. Zu erkennen ist, dass das Zielgebiet der Statistik bei dem Precisionwert 1 und dem Recallwert 1 liegt. Alle Abweichungen von diesen Koordinaten verursachen Kosten, doch unterscheiden sich diese in

[2] Der gleiche Diagrammtyp wird innerhalb der folgenden Seiten mehrmals verwendet.

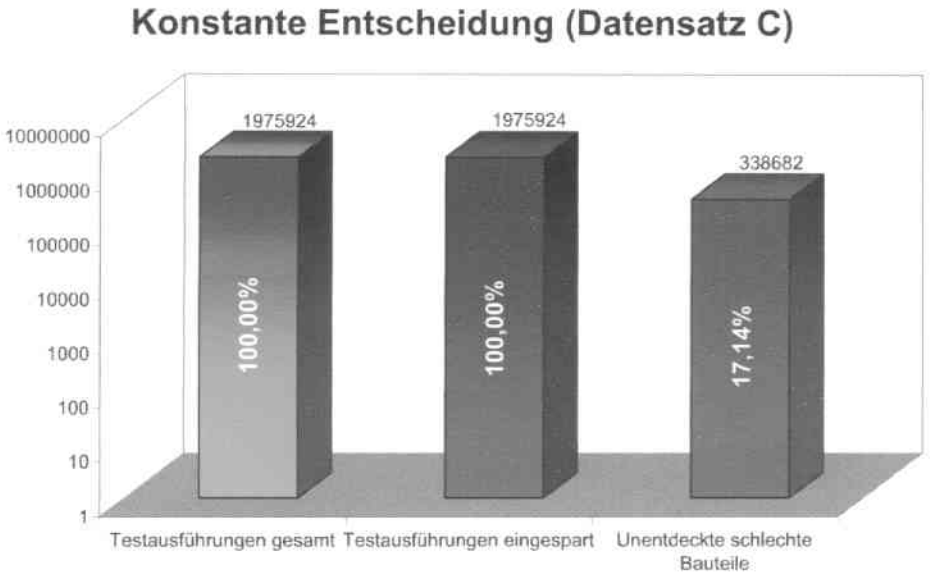


Abbildung 6.2: Konstante Entscheidung: Gesamtüberblick der Klassifikationsergebnisse

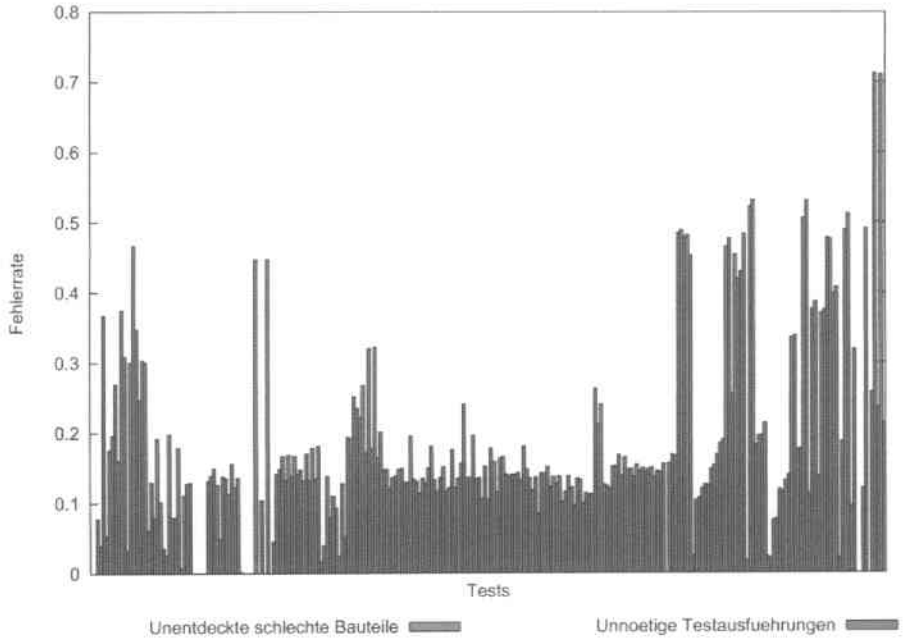


Abbildung 6.3: Konstante Entscheidung: Fehlerraten einzelner vorhergesagter Tests

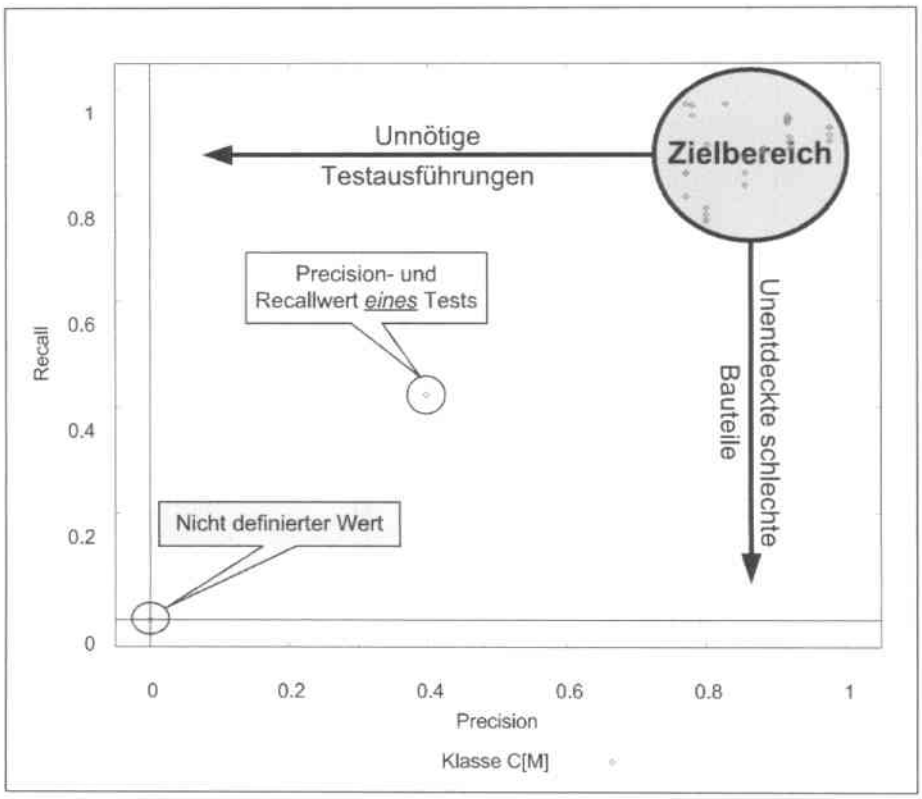


Abbildung 6.4: Beispiel einer Precision-Recall Statistik der Klasse C_M

ihrer Höhe in Abhängigkeit davon, ob eher die Precision (zusätzliche unnötige Testausführungen) oder der Recall (zusätzliche unentdeckte, schlechte Bauteile) sinkt[3]. Das gleiche Beispiel lässt sich ebenfalls für die Klasse C_R verwenden, jedoch müssten die beiden Pfeilüberschriften *Unnötige Testausführungen* und *Unentdeckte schlechte Bauteile* für die Klasse C_R getauscht werden.

Werden nun die Precision-Recall Graphen dieser Klassifikationstechnik in den Abbildungen 6.5, 6.6 und 6.7 betrachtet, so lässt sich beobachten, dass der Precisionwert der Klasse C_M sowohl in der Einzelübersicht, als auch in der Gesamtübersicht den Wert Null besitzt. Dies lässt sich durch die Definition der Precision für eine Klasse C erklären:

$$\text{Precision}_C = \frac{|\{\text{relevante Objekte} \in C\} \cap \{\text{gefundene Objekte}\}|}{|\{\text{gefundene Objekte}\}|}$$

Aus dieser Definition ist ersichtlich, dass im Falle, dass die Menge gefundener Objekte für die Klasse C leer ist, kein Precisionwert bestimmt werden kann. Um eine Illustration der fehlenden Werte dennoch zu ermöglichen, verwendet diese Arbeit für die Darstellung nicht existenter Precision- bzw. Recallwerte

[3]Wie bereits in den Annahmen im Abschnitt 3.1.1 festgelegt, sind die Kosten für übersehene Escapes um den Faktor 100 teuerer als die (unnötigen) .

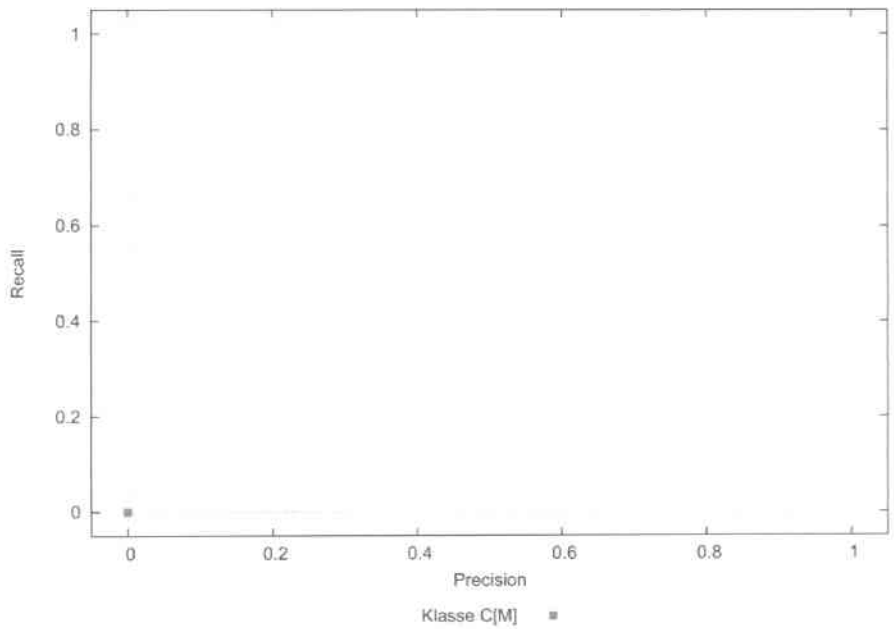


Abbildung 6.5: Konstante Entscheidung: Precision-Recall Statistik der Klasse C_M

für diese ersatzweise den Wert 0.0.

In der im Folgenden eingesetzter Darstellung der Precision-Recall Diagramme repräsentiert jeder Punkt in den Einzelklassenübersichten (Abb. 6.5 und 6.6) einen Test einer vorhergesagten Gruppe. In der Gesamtübersicht (Abb. 6.7) stellen die beiden Punkte jeweils die Mittelwerte aller Precision- und Recallwerte beider Klassen C_M und C_R dar.

Auf den Precision-Recall Graphen ist deutlich zu erkennen, dass die Precision der Klasse C_R über einen relativ breiten Bereich gestreut ist. Dies hängt natürlich damit zusammen, dass der erzielte Precisionwert das Gegenstück des Anteils unentdeckt verbliebener, schlechter Bauteile darstellt. Somit sieht man in der Gesamtstatistik den Precisionwert der Klasse C_R bei ca. 83 Prozent liegen. Der Recallwert der Klasse C_M ist durchgehend 0.0, der Precisionwert ist, wie oben bereits erklärt, nicht definiert.

Auswertung

Tatsächlich sieht man, dass die Anwendung dieser Methode nicht profitabel ist:

> Bei Kosten K_E von 0.01 EUR für einen Escape und Kosten K_T von 0.0001 EUR für einen ausgeführten Test (siehe auch Annahmen im Abschnitt 3.1.1), würde das Testen aller 8291 evaluierter Bauteile ohne der Anwendung dieses Klassifikationsverfahrens 376.41 EUR kosten. Durch den Einsatz des konstanten Entscheiders könnten

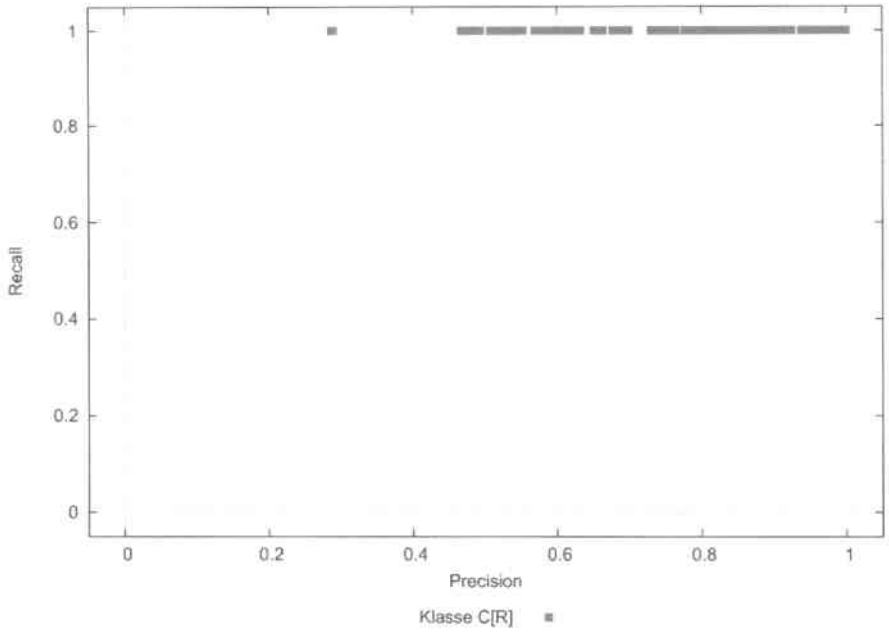


Abbildung 6.6: Konstante Entscheidung: Precision-Recall Statistik der Klasse C_R

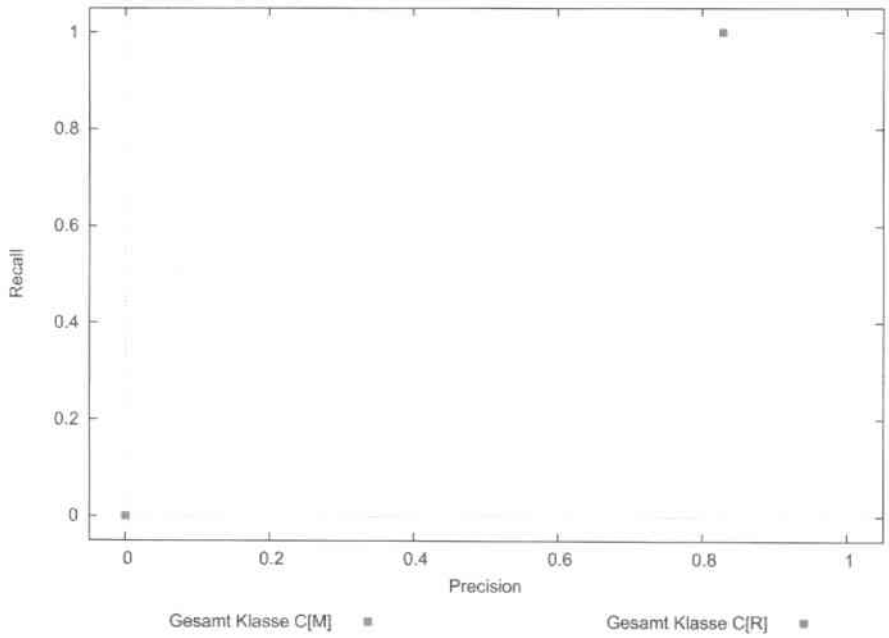


Abbildung 6.7: Konstante Entscheidung: Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

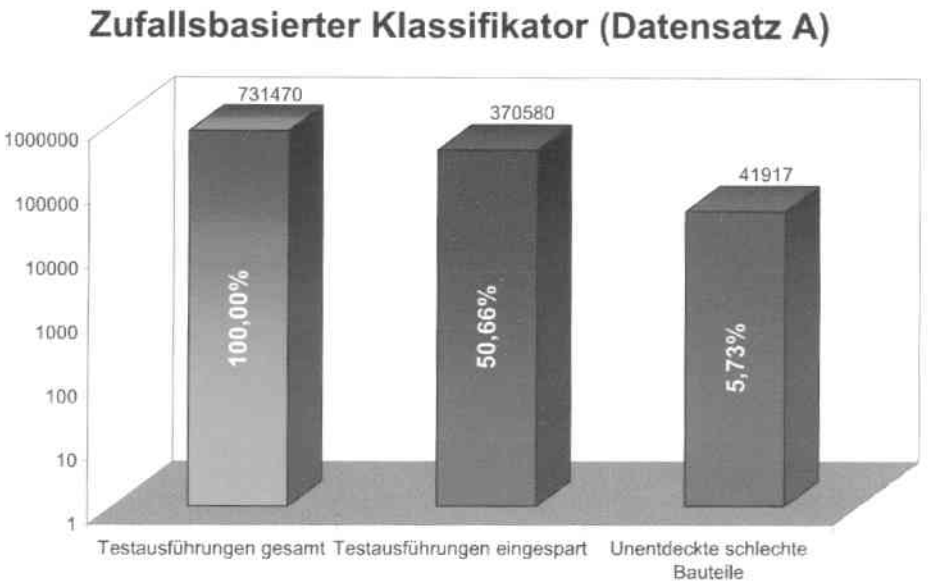


Abbildung 6.8: Zufallsbasierter Klassifikator: Gesamtüberblick der Klassifikationsergebnisse

zwar 221.37 EUR durch die Auslassung der Testgruppen eingespart werden, doch würden dadurch andererseits, bedingt durch die große Anzahl unentdeckter, schlechter Bauteile, Mehrkosten in Höhe von 3386.82 EUR entstehen. Somit würde die Anwendung dieser Optimierungstechnik beachtliche Zusatzkosten verursachen, anstatt die Testkosten zu senken.

6.2.2 Zufallsbasierte Klassifikation

Das nächste vorgestellte Klassifikationsverfahren basiert auf dem Zufallsprinzip (siehe auch Abschnitt 4.4.1.2). Diese Technik soll einen unwissenden Entscheider simulieren und einen Überblick über die Qualität auf diese Art gefällter Entscheidungen geben. Als Beispieldaten werden hier die Messwerte aus dem Datensatz A gewählt.

Der in der Grafik 6.8 präsentierte Gesamtüberblick der Klassifikationsergebnisse lässt erkennen, dass dieser Entscheider etwa nur ein Drittel der Escapes des konstanten Entscheiders hinterlässt, nämlich nur 5.73 Prozent. Die Anzahl eingesparter Tests sinkt hingegen *nur* um die Hälfte auf etwas mehr als 50 Prozent. Insgesamt scheint also diese Klassifikationstechnik erfolgreicher zu sein.

Bei der Betrachtung der Ergebnisse einzelner Tests (6.9) ist zu erkennen, dass im Gegensatz zum konstanten Entscheider, ein guter Teil aller Testausführungen unnötig war. Diese Feststellung wird auch von den folgenden Precision-Recall Statistiken 6.10, 6.11 und 6.12 gestützt.

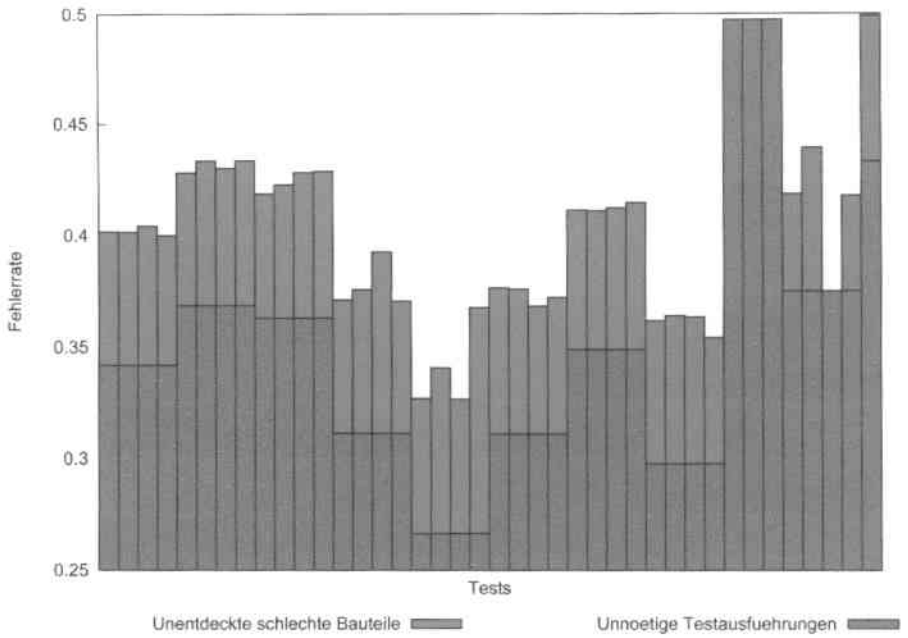


Abbildung 6.9: Zufallsbasierter Klassifikator: Fehlerraten einzelner Tests vorhergesagter Testgruppen

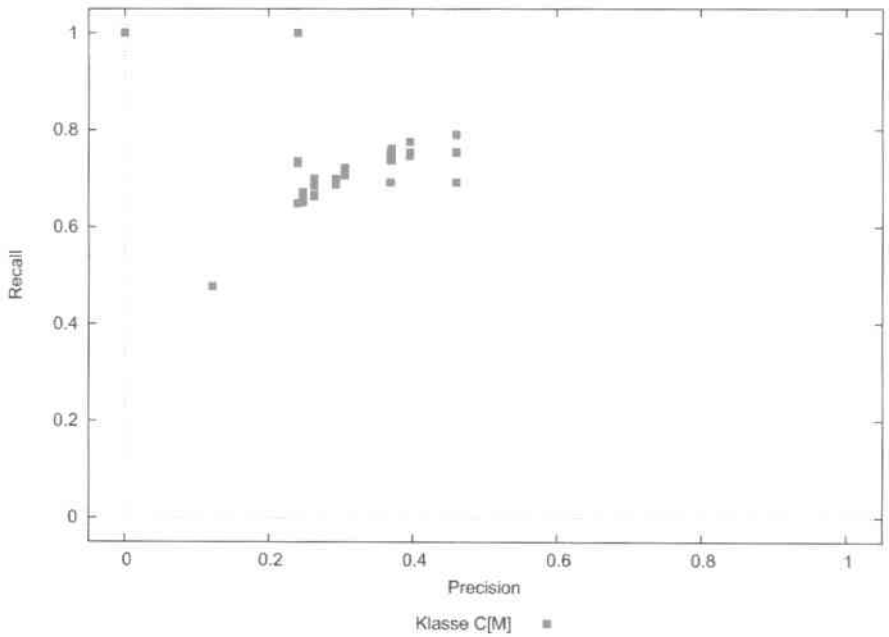


Abbildung 6.10: Zufallsbasierter Klassifikator: Precision-Recall Statistik der Klasse C_M

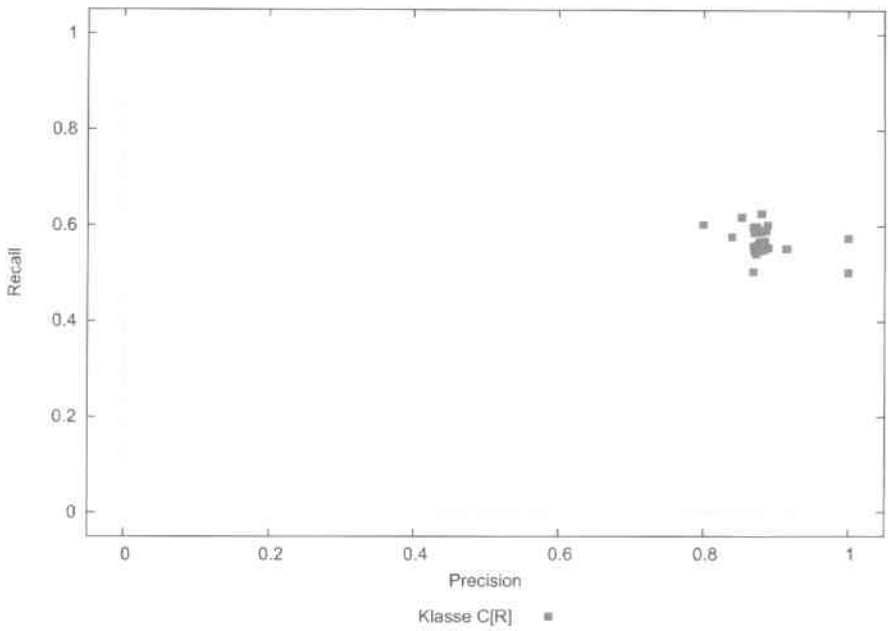


Abbildung 6.11: Zufallsbasierter Klassifikator: Precision-Recall Statistik der Klasse C_R

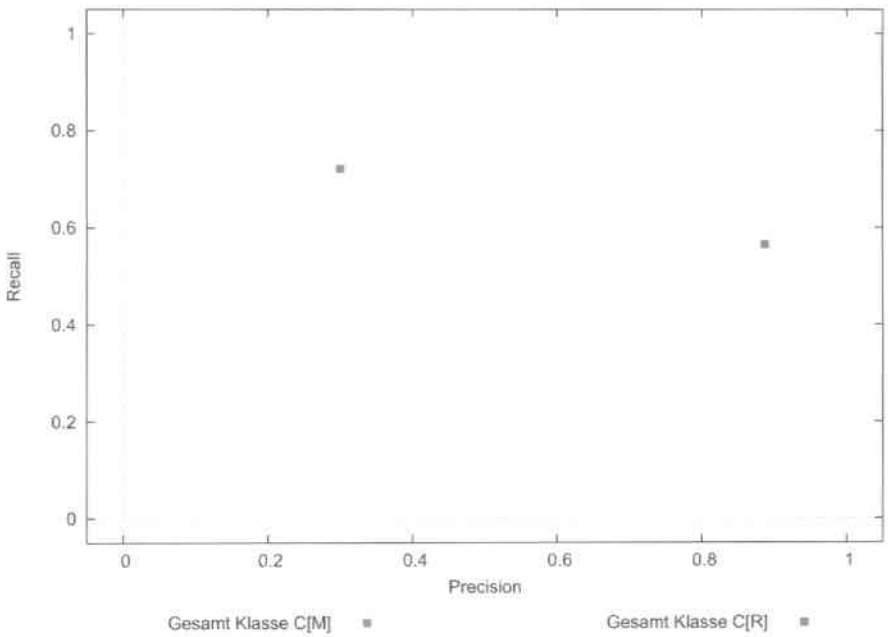


Abbildung 6.12: Zufallsbasierter Klassifikator: Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

Vor allem an der geringen Precision der Klasse C_M $(0.20 - 0.45)$ erkennt man die der Klasse unnötigerweise zugeordneten Merkmalsvektoren, welche das unnötige Ausführen entsprechender Gruppen nach sich zogen. Die Recallwerte dieser Klasse häufen sich hauptsächlich im oberen Mittelfeld $(0.65 - 0.80)$ und deuten damit an, dass im Durchschnitt pro Test „nur" etwa 27% aller schlechten Bauteile unentdeckt blieben.

An den relativ hohen Precisionwerten $(0.80 - 0.90)$ der Klasse C_R lässt sich darüber hinaus gut erkennen, dass die als einsparbar klassifizierten Merkmalsvektoren sich meistens auch tatsächlich als auslassbar erwiesen haben. Dies lässt sich dadurch erklären, dass - pro Test betrachtet - sich die Mehrheit der Messwerte innerhalb der kritischen Grenzen befindet. Die eher geringen Recallwerte $(0.50 - 0.60)$ dieser Klasse lassen weiterhin darauf schließen, dass circa die Hälfte aller in der Tat einsparbaren Tests als solche klassifiziert werden konnten.

Auswertung

Mit einer einfachen Analyse kommt man auch bei diesem Verfahren schnell zum Schluss, dass von einem ernsthaften Einsatz in der Praxis dringend abzuraten ist:

> Der Datensatz A beinhaltet 19268 zu evaluierende Bauteile. Würden diese mit allen 95 Tests pro Bauteil getestet werden, so würden die Gesamtkosten dafür 183.05 EUR betragen. Unter Einsatz dieser Klassifikationstechnik ließen sich durch die Testauslassungen 37.06 EUR einsparen. Die Zusatzkosten, welche durch Escapes verursacht werden würden, wären in diesem Fall jedoch deutlich höher und würden bei 419.17 EUR liegen. Auch dieses Verfahren würde am Ende mehr Kosten verursachen als einsparen.

6.2.3 Bayes Klassifikation

Das letzte Verfahren, welches in dieser Arbeit zu den trivialen Klassifikatoren zugeordnet wird, ist die Bayes Klassifikation (siehe Abschnitt 4.4.1.3). Das Klassifikationsverfahren erhält dabei ausschließlich das während der Lernphase angesammelte Wissen. Unter dieser Voraussetzung soll die Optimierung der Testskosten der Bauteile des Datensatzes B erprobt werden.

Dieses Vorgehen entspricht der Entscheidung eines Testingenieurs, der nach dem Betrachten einer begrenzten Anzahl von getesteten Bauteilen für die restliche Bauteilmenge *a priori* festlegen muss, welche Tests ausgeführt und welche eingespart werden sollen.

Die Gesamtübersicht über die Klassifikationsresultate dieser Methode finden sich in der Abbildung 6.13. Es fällt gleich auf, dass der Anteil an Escapes bei diesem, wie auch bei den beiden vorhergehenden Verfahren mit 8.78% zwar immer noch recht hoch, aber andererseits auch nur halb so groß, wie bei dem konstanten Entscheider ist. Die Anzahl der eingesparten Testausführungen ist

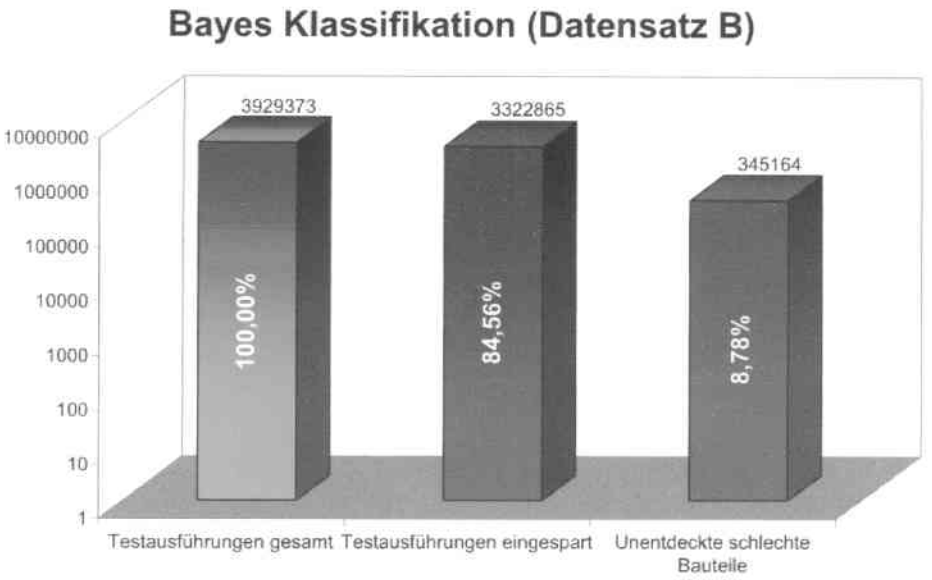


Abbildung 6.13: Bayes Klassifikator: Gesamtüberblick der Klassifikationsergebnisse

ebenfalls sehr hoch, liegt aber deutlich unter 100%.

Die Einzeltestübersicht in dem Diagramm 6.14 zeigt, dass das Klassifikationsverfahren *nicht* alle Tests permanent eingespart, sondern einzelne Tests auch ausgeführt hat. Per Definition muss das Verfahren einen Tests entweder *ausnahmslos* ausführen oder einsparen. Für alle Tests mit mindestens einer (unnötigen) Ausführung bedeutet die Feststellung gleichzeitig, dass diese Tests durchgehend auf allen Bauteilen ausgeführt wurden. Weiterhin lässt sich anhand dieses Diagramms und der Precision-Recall Statistiken in den Grafiken 6.15 bis 6.17 ermitteln, dass einige Tests besonders gut vorhergesagt werden konnten bzw. beinahe keine Fehlerraten aufweisen.

Interpretiert man die Precision- und Recallwerte der Klasse C_R, so erkennt man, dass einzelne Tests beinahe fehlerfrei der Klasse C_R zugeordnet werden konnten. Mit dem verwendeten Verfahren ist eine derart gute Zuordnung zur Klasse C_R nur dann möglich, wenn ein Test quasi fehlerfrei ist. Die weiteren Datenpunkte in diesem Diagramm deuten an, dass annähernd alle anderen Tests zwar vollständig zur Klasse C_R klassifiziert wurden (Recall = 1.0), dabei aber auch Escapes hingenommen werden mussten (Precision: 0.65 - 0.95).

Die Precision-Recall Statistik der Klasse C_M[4] unterstreicht hingegen die Tatsache, dass einzelne Tests durchgehend ausgeführt wurden (Recall = 1.0), was jedoch auf vielen Bauteilen nicht notwendig war (Precision: 0.55 - 0.65). Da einige Precisionwerte nicht definiert sind, kann weiterhin die Schlussfolgerung gezogen werden, dass auch Tests existieren, die trotz bestehender Notwendigkeit

[4]In der Precision-Recall Statistik der Klasse C_M in Abbildung 6.15 sind zwar nur drei unterschiedliche Punkte zu erkennen, doch wird dieses Phänomen dadurch hervorgerufen, dass mehrere Tests sehr ähnliche oder sogar gleiche Precision- und Recallwerte haben.

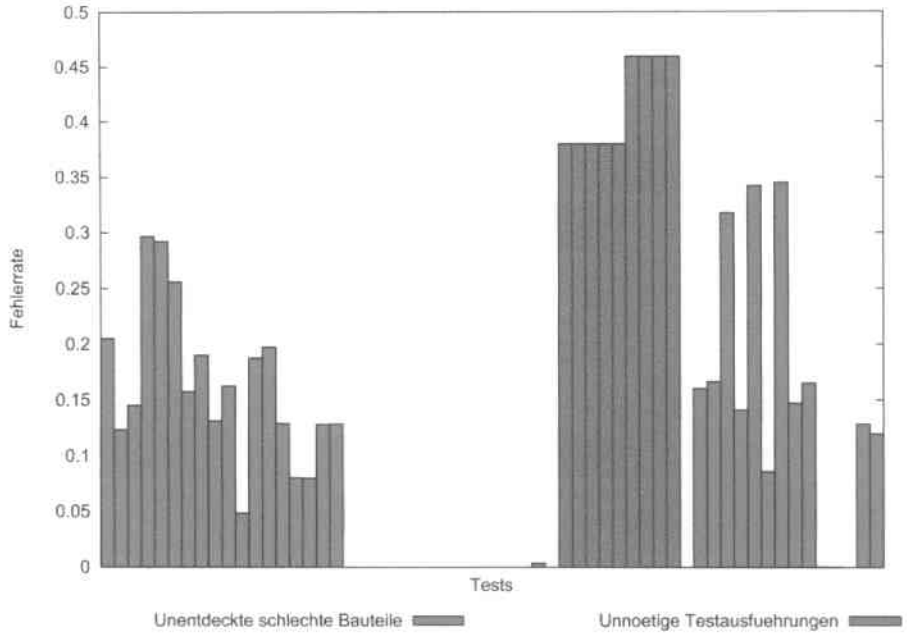


Abbildung 6.14: Bayes Klassifikator: Fehlerraten einzelner Tests vorhergesagter Testgruppen

kein Mal der Klasse C_M zugeordnet wurden, was auch durch die Existenz von Escapes (siehe Abbildung 6.14) und die Definition des eingesetzten Verfahrens bestätigt werden kann.

Auswertung

Eine Analyse dieses Verfahrens beweist ein weiteres Mal, dass sich die trivialen Klassifikatoren für einen echten Einsatz in der Praxis nicht eignen:

> Würden alle 69747 zur Evaluation bestimmten Bauteile des Datensatzes B mit Hilfe der Bayes Klassifikation optimiert werden, könnten 332.29 EUR eingespart werden. Da jedoch diesem Verfahren, bei etwa jeder elften Vorhersage ein Escape unterläuft, belaufen sich die dadurch verursachten Mehrkosten auf 3451.64 EUR. Ohne den Einsatz einer Optimierung, bei ständiger Ausführung aller Tests der Testsequenz, würden hingegen Testkosten von lediglich 676.54 EUR entstehen.

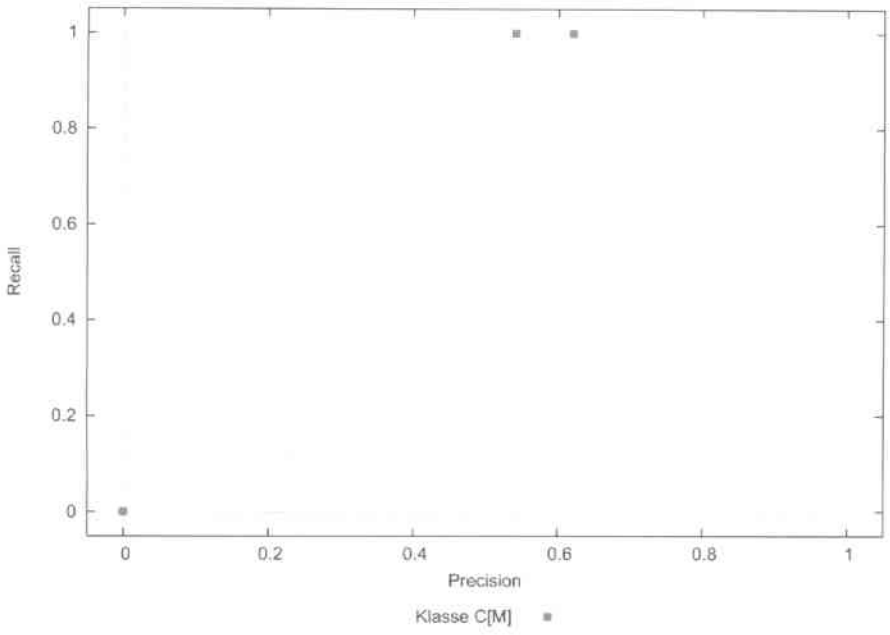


Abbildung 6.15: Bayes Klassifikator: Precision-Recall Statistik der Klasse C_M

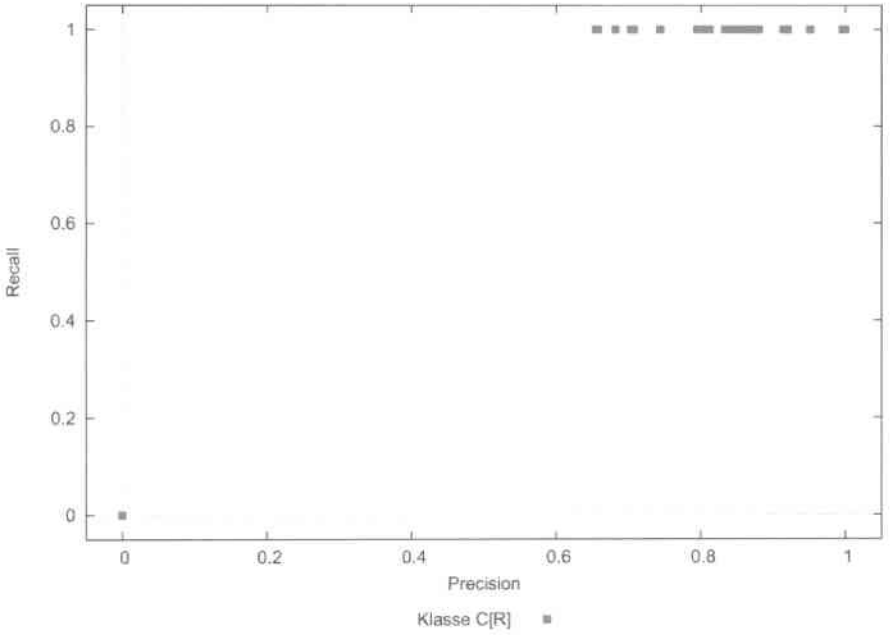


Abbildung 6.16: Bayes Klassifikator: Precision-Recall Statistik der Klasse C_R

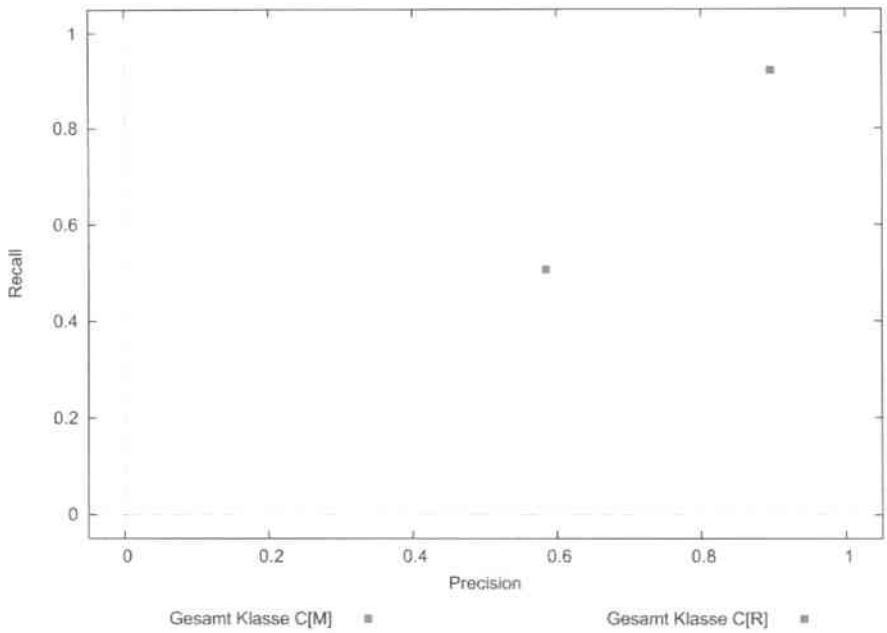


Abbildung 6.17: Bayes Klassifikator: Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

6.2.4 Resümee

Ein besonderes Problem aller trivialen Klassifikatoren lässt sich durch das in der Abbildung 6.18 dargestellte Problem verdeutlichen.

Auf der Grafik lässt sich vor allem gut erkennen, dass der dargestellte Test seinen Wertebereich während der laufenden Messung ändert. Mögliche Ursachen dafür können die im Abschnitt 2.1 und 3 beschriebenen Schwankungen des Herstellungsprozesses sein.

Derartige Schwankungen würden sich erwartungsgemäß gleich auf mehrere Eigenschaften eines Bauteils auswirken, sodass ein intelligenter Klassifikator in der Lage wäre, einen Unterschied festzustellen und den vormals zur Einsparung freigegebenen Test wieder auszuführen.

Für Klassifikatoren, welche ausschließlich auf dem *a priori* Wissen aufbauen, stellen gerade solche Tests bzw. Prozessschwankungen ein besonderes Risiko dar. Ein Test, wie auf der Abbildung 6.18 dargestellt, würde während der Lernphase als einsparbar gekennzeichnet werden, und würde dies auch nach der Änderung seines Wertebereichs bleiben. Bestimmte, gefährliche Produktionsschwankungen können somit völlig unbemerkt bleiben und einen großen finanziellen Schaden verursachen.

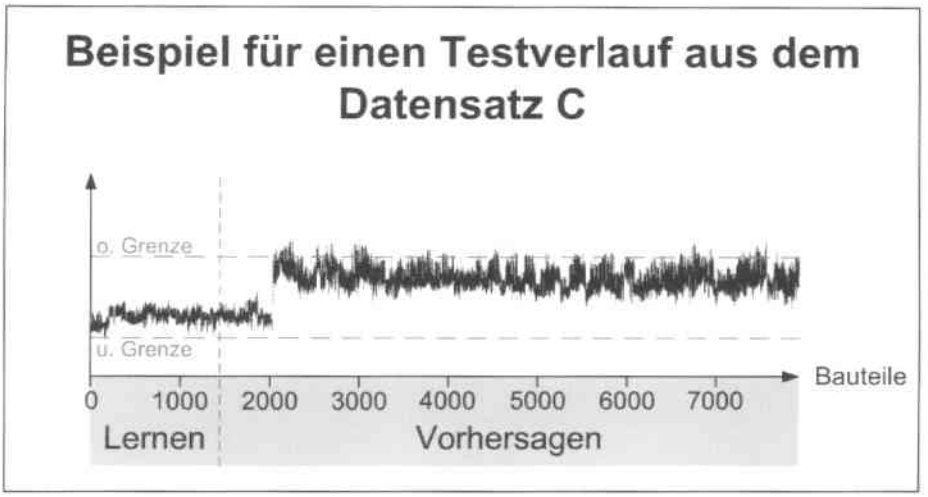


Abbildung 6.18: Beispiel: Schwierigkeiten bei der Optimierung eines Tests ausschließlich mit Hilfe von *a priori* Wissen.

6.3 Statische Klassifikatoren

In diesem Abschnitt werden einige Ergebnisse von Klassifikationsalgorithmen präsentiert, welche in der vorliegenden Arbeit als *statische Verfahren* genannt werden (siehe auch Abschnitt 4.4.2). Die vorgestellten Klassifikationstechniken zeichnen sich dadurch aus, dass sie für ihre Entscheidungsfindung nicht nur die Informationen aus der Lernphase verarbeiten können, sondern auch die auf dem zu klassifizierenden Bauteil gesammelten Daten. Zur Optimierung der Daten werden die im Abschnitt 4.3 vorgestellten Verfahren eingesetzt, diese werden vor jedem Experiment explizit benannt.

6.3.1 k Nächste Nachbarn Klassifikation

Als erstes statisches Klassifikationsverfahren, wird das k Nächste Nachbarn (*KNN*) Verfahren eingesetz. Im Folgenden wird dieses Verfahren in zwei unterschiedlichen Kombinationen auf den Daten des Datensatzes C vorgestellt.

6.3.1.1 KNN - Lineare Diskriminanzanalyse

Das erste Experiment mit dem k Nächste Nachbarn Algorithmus verwendet zwar mehrere unterschiedliche Datenoptimierungsverfahren, vor allem wird jedoch zur Dimensionsreduktion die lineare Diskriminanzanalyse (*LDA*) eingesetzt.

Die Anzahl der betrachteten nächsten Nachbarn wurde auf drei festgelegt[5].

[5]Die ausprobierte Anzahl an nächsten Nachbarn umfasste einen Bereich zwischen 3 und 31. Der Wert 3 erwies sich jedoch in mehreren Hinsichten als günstig: Es traten einerseits keine signifikanten Qualitätsunterschiede der Klassifikationsergebnisse auf und andererseits, sprachen die Performancegründe für einen eher kleineren Wert. Die Ähnlichkeit der Vorhersa-

Für weitere Implementierungsdetails bezüglich des implementierten k Nächste Nachbarn Algorithmus sei auf den Abschnitt 4.4.2.1 verwiesen. Die einzelnen eingesetzten Datenoptimierungsverfahren sind im Folgenden benannt:

Filterung: Für die Filterung werden die Korrelationskoeffizienten (Gleichung 4.8) verwendet. Es werden für jeden vorherzusagenden Test einer Testgruppe 10% seiner bestkorrelierten, in der Testsequenz vorhergehenden Partner ausgewählt. Bei jedem ausgewählten Korrelationspartner t_κ wird die Abweichung des auf dem aktuellen Bauteil ermittelten Messwertes zum Bestwert des Korrelationspartners t_κ mit Hilfe des Betrachtungspuffers (4.3.1.4) verfolgt[6].

Datenanreicherung: Die Anreicherung der Daten geschieht mit Hilfe der gerade erwähnten Korrelationskoeffizienten.

Normierung: Als Normierungsfunktion kommt die Funktion N_1 (siehe Gleichung 4.2) zum Einsatz.

Rauschunterdrückung: Das statische Rauschen wird mit Hilfe eines Betrachtungspuffers über die letzten 80 Bauteile minimiert[7].

Eine Übersicht der Ergebnisse, die mit dieser Kombination von Datenoptimierungsverfahren erzielt werden konnten, findet sich in dem Diagramm 6.19. Bemerkenswert hoch ist in dieser Grafik der Prozentsatz unentdeckter, schlechter Bauteile. Dieser erreicht mit 5.47% beinahe die Ausmaße des zufallsbasierten Klassifikators (5.73%).

Die hohe Anzahl der Escapes ist jedoch ganz typisch für die lineare Diskriminanzanalyse. Dieses Verfahren konnte leider in keinem Experiment mit keinem verwendeten Datensatz überzeugen. Ein möglicher Grund dafür, könnte beispielsweise die Komplexität der zugrunde liegenden Daten sein, sodass eine Reduktion des n-dimensionalen Merkmalsraumes auf eine Dimension wahrscheinlich zu viele wertvolle Informationen verloren gehen lässt. Eine andere denkbare Alternative ist, dass die lineare Diskriminanzanalyse zwar die Gesamtfehlerquote zu reduzieren versucht, dabei allerdings außer Acht lässt, dass bestimmte Fehler *teuerer* sind als bestimmte andere.

Die detaillierten Fehlerraten einzelner Tests sowie die Precision-Recall Statistiken dieses Experimentes folgen in den Abbildungen 6.20 bis 6.23.

Die Analyse der Precision-Recall Statistiken beider Klasse fördert relativ große Punktwolken und somit große Unterschiede zwischen den einzelnen Tests zu Tage. Bei der Betrachtung der Statistik für die Klasse C_M, lassen sich mehrere Tests mit besonders hohen, aber auch einige mit besonders

gequalität kann möglicherweise auf die eingesetzten Datenoptimierungsalgorithmen (aus dem Abschnitt 4.3) zurückgeführt werden, welche die vorhergesagten Daten bereits vom statischen Rauschen und möglichen Protzesschwankungen befreiten.

[6] Diese Filtermethode erwies sich bereits in den ersten durchgeführten, kleineren Versuchsreihen als äußerst effektiv und wird deshalb in den meisten folgenden Experimenten ebenfalls eingesetzt.

[7] Diese Größe des Betrachtungspuffers hat sich in mehreren Versuchen als optimal erwiesen.

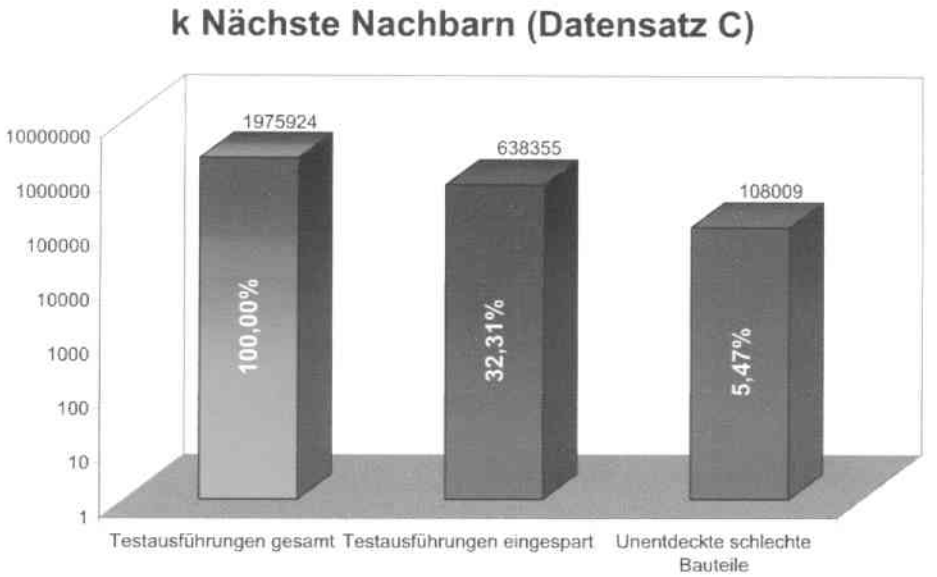


Abbildung 6.19: k Nächste Nachbarn Klassifikator (mit LDA): Gesamtüberblick der Klassifikationsergebnisse

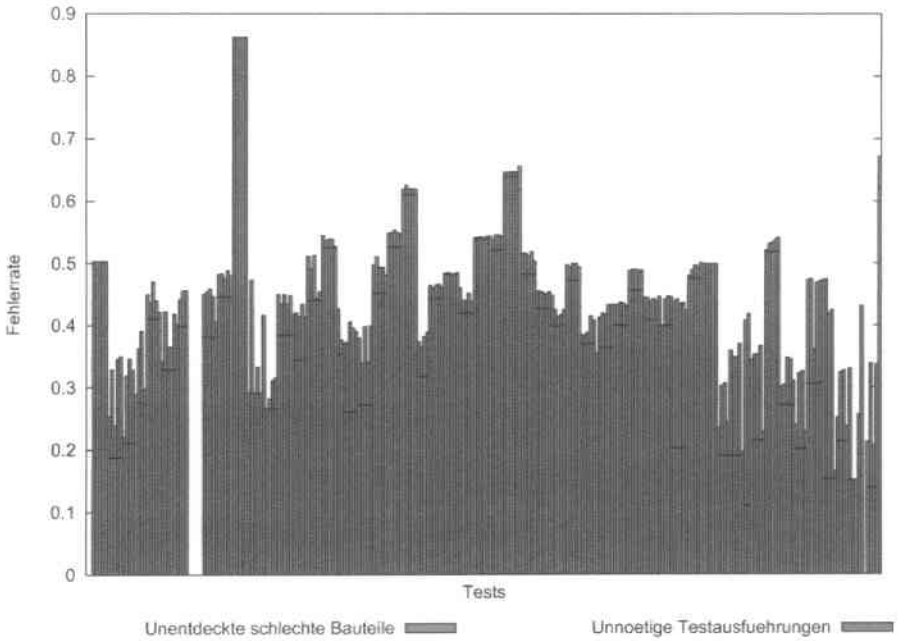


Abbildung 6.20: k Nächste Nachbarn Klassifikator (mit LDA): Fehlerraten einzelner Tests vorhergesagter Testgruppen

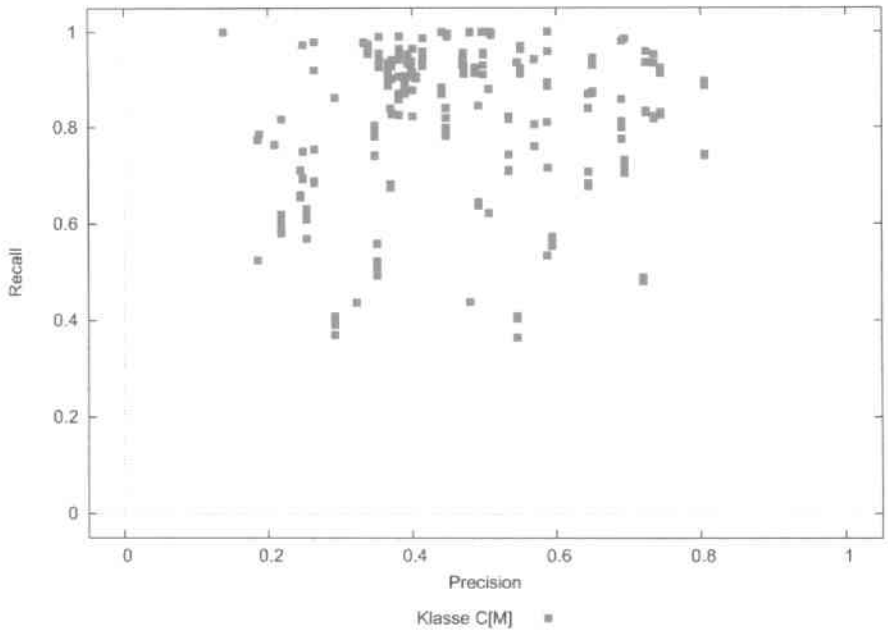


Abbildung 6.21: k Nächste Nachbarn Klassifikator (mit LDA): Precision-Recall Statistik der Klasse C_M

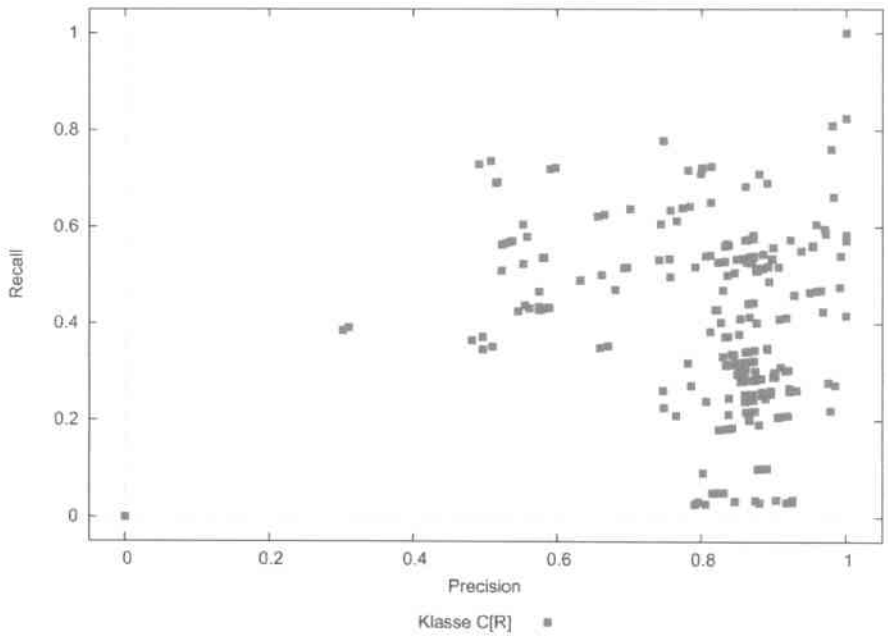


Abbildung 6.22: k Nächste Nachbarn Klassifikator (mit LDA): Precision-Recall Statistik der Klasse C_R

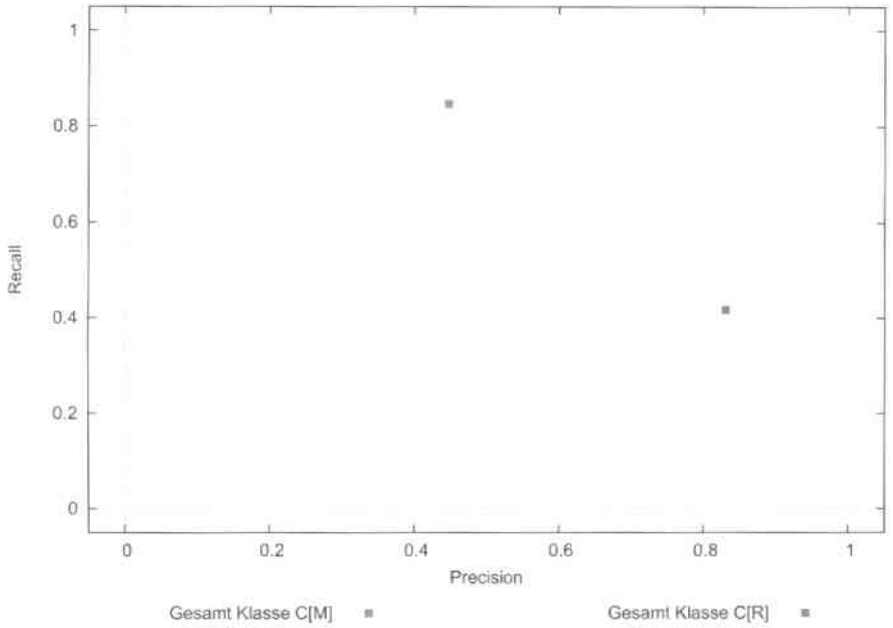


Abbildung 6.23: k Nächste Nachbarn Klassifikator (mit LDA): Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

niedrigen Recallwerten $(0.4 - 1.0)$ erkennen. Das lässt darauf schließen, dass dem eingesetzten Klassifikationsverfahren bei manchen Tests beinahe keine und bei diversen anderen, relativ viele Escapes unterlaufen sind. Der maximale Precisionwert der Statistik liegt bei 0.8 und deutet an, dass das Verfahren nicht präzise bei der Zuordnung der Tests zur Klasse C_M vorgegangen ist, viele Tests wurden entgegen der Notwendigkeit als ausführbar klassifiziert.

Die Auswertung der Precisionwerte der Klasse C_R offenbart, dass manche Tests stets ausgeführt wurden (Precision: < 0) und einige andere keine Escapes hinterliessen (Precision: 1.0). Einige der Tests ohne Escapes wurden vollständig korrekt der Klasse C_R zugeordnet, was die Betrachtung der Recallwerte in der Nähe von 1.0 bestätigt.

Auswertung

Wie bereits bei den trivialen Klassifikatoren aus dem Abschnitt 6.2 gesehen, spielt die Quote unentdeckter, schlechter Bauteile eine maßgebliche Rolle bei der Bewertung eines Klassifikationsverfahrens. Die Fehlerquote des in diesem Abschnitt vorgestellten Verfahrens lag bei 5.47 Prozent, und somit annährend so hoch, wie beim zufallsbasierten Klassifikator. Rechnet man die Einsparnisse und die durch die Escapes verursachten Zusatzkosten zusammen, ergibt sich auf den Daten des verwendeten Datensatzes C folgendes Bild:

Die Gesamtzahl aller Testausführungen auf den Evaluationsdaten beträgt circa 3.8 Mio. Die Testkosten ohne den Einsatz eines Optimierungsverfahrens würden demnach 376.41 EUR betragen. Mit dem aktuellen Verfahren könnten über 0.6 Mio. Testausführungen (63.84 EUR) eingespart werden, doch die durch die knapp 110 Tausend Escapes verursachten Kosten würden mit 1080.09 EUR die Ersparnisse und sogar die normalen Testkosten bei weitem übertreffen.

6.3.1.2 KNN - Learning Vector Quantization

Wie bereits im Abschnitt gesehen, ist die Fehleranfälligkeit des k Nächste Nachbarn Algorithmus in Verbindung mit der linearen Diskriminanzanalyse sehr hoch und für den Einsatz in der Praxis ungeeignet. Aus diesem Grund wurde ein anderes Datenoptimierungsverfahren ausprobiert, nämlich die Learning Vector Quantization (*LVQ*) (siehe auch 4.3.2.2). Die Auswahl und die Einstellungen weiterer Datenoptimierungsverfahren wurde gegenüber dem vorhergehenden Experiment (Abs. 6.3.1.1) nicht verändert.

Für diesen Versuch, sowie für weitere Experimente, wurden die Parameter des LVQ-Verfahrens und des zu Grunde liegenden k-means Clustering Algorithmus wie folgt gewählt[8]:

LVQ Fenster: Das LVQ Fenster [Koh90] wurde auf den Wert 0.25 bestimmt.

LVQ Lernrate: Die Lernrate der Learning Vector Quantization wurde auf 0.5 festgelegt.

***k*-means Cluster:** Die Anzahl der zu bildenden Cluster wurde mit 16 spezifiziert.

***k*-means Iterationen:** Die maximale Anzahl der Iterationen des Clusteringverfahrens wurde mit 100 angegeben.

Als zusätzliche Maßnahme zur Senkung der Anzahl der unentdeckt verbleibenden schlechten Bauteile wurde der im Abschnitt 4.4.5 beschriebene Schwellenwert auf 1.0 gesetzt, um die Entscheidung zu Gunsten der Einsparung einer Testgruppe nur dann zu ermöglichen, wenn sich der Klassifikationsalgorithmus der Zuordnung ganz sicher ist.

Die Abbildung 6.24 gibt einen Überblick über die Ergebnisse des Experiments. Zwei Sachverhalte stechen aus diesem Diagramm besonders hervor. Einerseits ist die Anzahl der eingesparten Testausführungen relativ gering, doch andererseits gibt es keine Escapes. Damit stellt dieser Klassifikationsalgorithmus das erste in der Praxis tatsächlich einsetzbare Optimierungsverfahren dar. Weitere Ergebnisse dieses Verfahrens, sowie die Precision-Recall Statistiken finden sich in den Grafiken 6.25, 6.26, 6.27 und 6.28.

[8] Die Einstellungen basieren dabei auf vorhergehenden Versuchen, während welcher sich die optimalen Werte der aufgezählten Einstellungen empirisch ermittelt ließen.

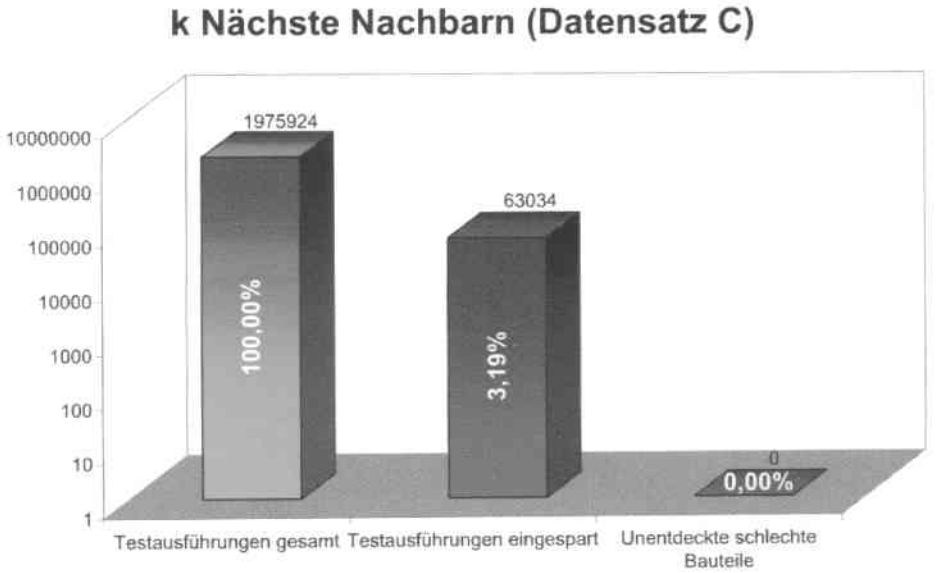


Abbildung 6.24: k Nächste Nachbarn Klassifikator (mit LVQ): Gesamtüberblick der Klassifikationsergebnisse

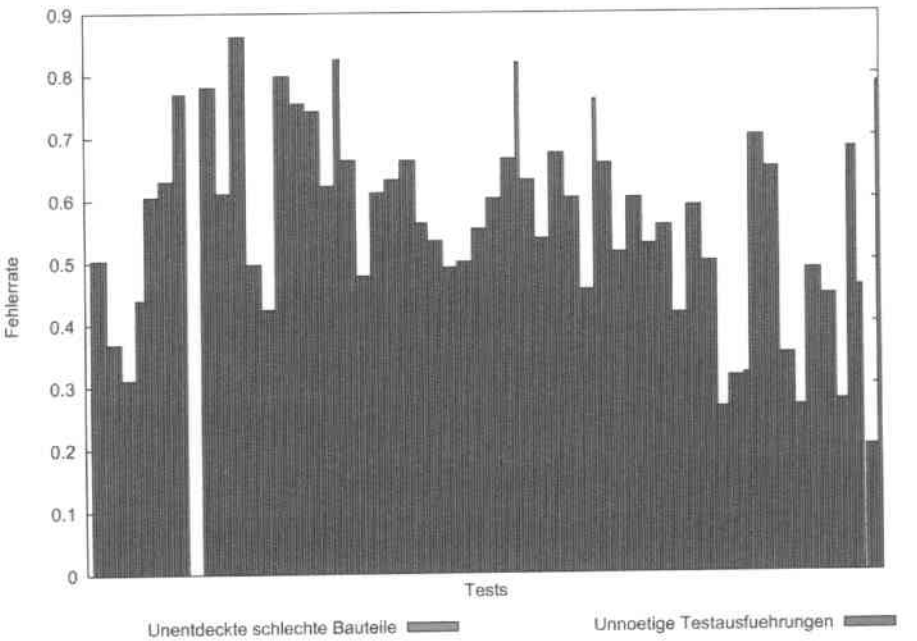


Abbildung 6.25: k Nächste Nachbarn Klassifikator (mit LVQ): Fehlerraten einzelner Tests vorhergesagter Testgruppen

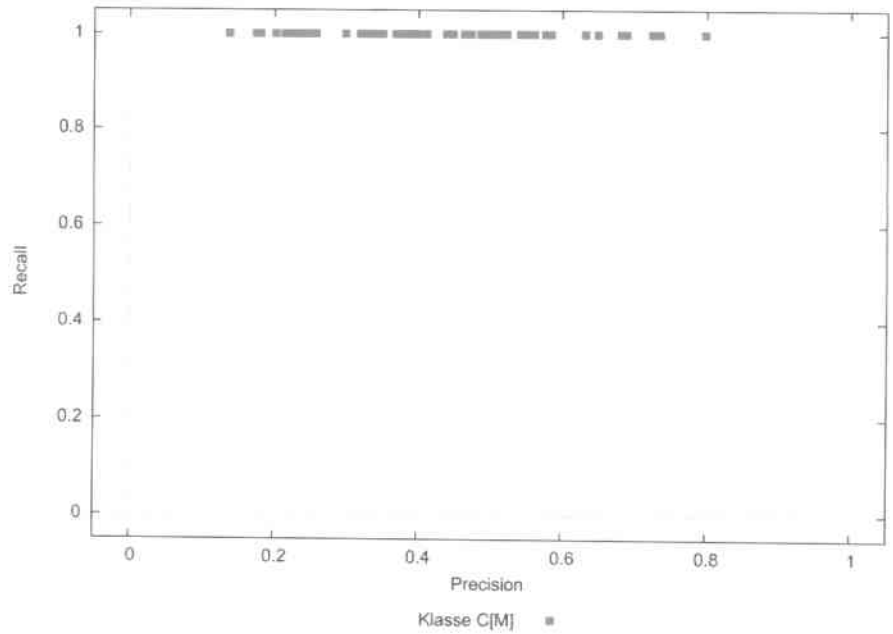


Abbildung 6.26: k Nächste Nachbarn Klassifikator (mit LVQ): Precision-Recall Statistik der Klasse C_M

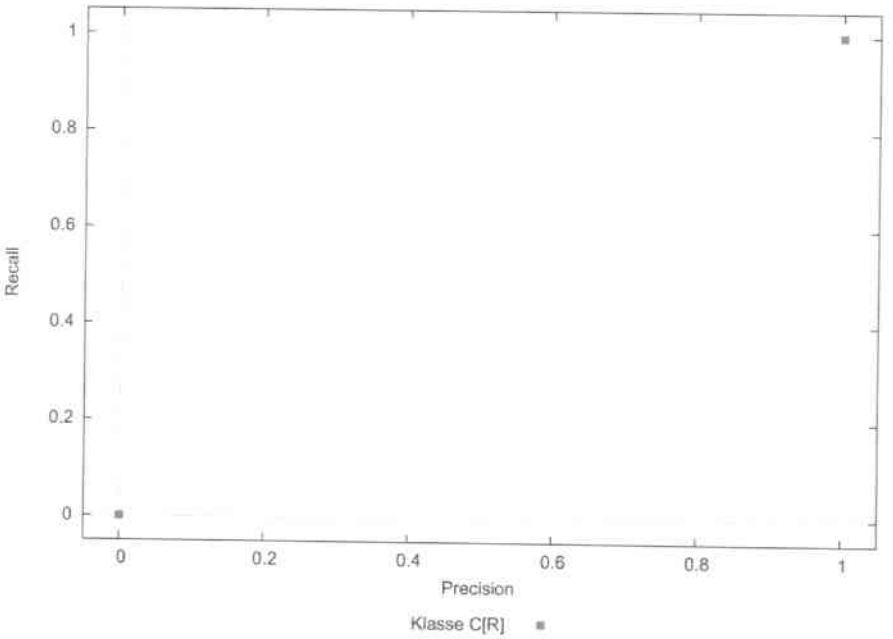


Abbildung 6.27: k Nächste Nachbarn Klassifikator (mit LVQ): Precision-Recall Statistik der Klasse C_R

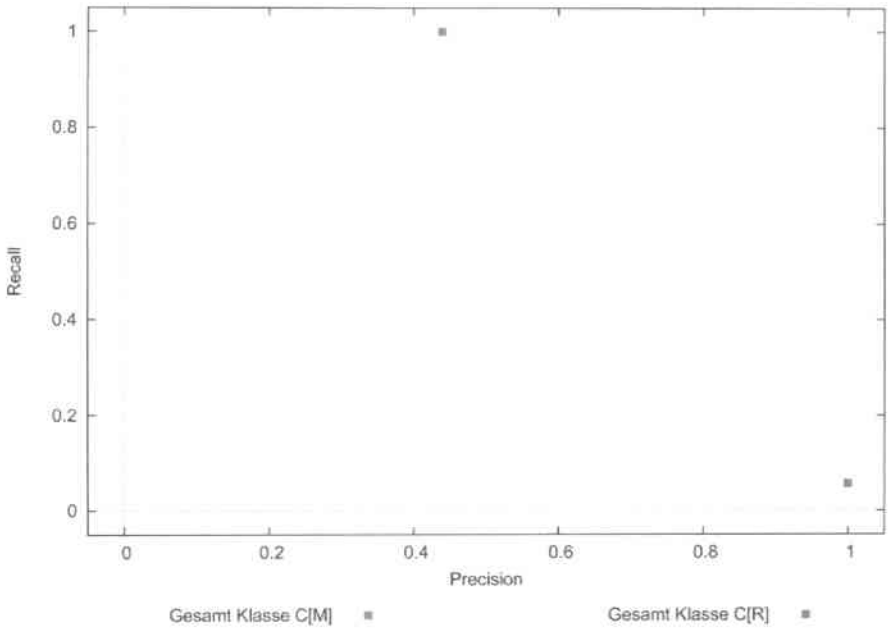


Abbildung 6.28: k Nächste Nachbarn Klassifikator (mit LVQ): Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

Der Einzelübersicht der Tests lässt sich entnehmen, dass viele Tests ausgeführt wurden, obwohl keine Notwendigkeit dazu bestand. Zum gleichen Schluss führen auch die Ergebnisse der Precision-Recall Statistik der Klasse C_M. Der Recallwert aller Tests liegt bei 1.0, die Precision streut hingegen zwischen 0.20 und 0.75. Die teilweise sehr geringe Precision verdeutlicht ein weiteres Mal, dass diverse Testausführungen durch den Algorithmus hätten eingespart werden können.

Der Precision-Recall Graph der Klasse C_R weist lediglich darauf hin, dass manche Tests einwandfrei klassifiziert werden konnten, wohingegen andere Tests wiederum unnötigerweise stets ausgeführt wurden. Der Recallwert dieser Klasse in der Precision-Recall Übersichtsgrafik betont die geringe Beachtung der möglichen Einsparung vorhergesagter Testgruppen.

Auswertung

Der Einsatz der Learning Vector Quantization scheint einen sehr positiven Einfluss auf die Klassifikation zu besitzen. Die Ergebnisse des angewandten Klassifikationsalgorithmus sind bedeutend besser, als die des Verfahrens, aus dem zuletzt vorgestellten Experiment (Abschnitt 6.3.1.1). Zusammengefasst lässt sich sagen, dass das in diesem Experiment vorgestellte Optimierungsverfahren das erste bisher aufgezählte Verfahren ist, welches in der Industrie eingesetzt werden könnte, um die Testkosten tatsächlich zu senken.

Ohne den Einsatz eines Optimierungsverfahrens betragen die Testkosten der evaluierten 8291 Bauteile aus dem Datensatz C, wie bereits gesehen 376.41 EUR. Mit Hilfe des vorgestellten Optimierungsverfahrens könnten auf den zu vorhersagenden Bauteilen 6.30 EUR oder 1.67 Prozent der Gesamttestkosten eingespart werden.

6.3.2 Multivariate Gaußklassifikation

Das zweite präsentierte Klassifikationsverfahren aus der Kategorie statischer Klassifikatoren, ist das auf multivariaten Normalverteilungen aufbauende Klassifikationsverfahren (siehe Abschnitt 4.4.2.2). In diesem Abschnitt werden zwei Konfigurationen eines Optimierungsverfahrens auf Basis dieses Klassifikators vorgestellt. Beide Experimente wurden auf den Daten des Datensatzes A durchgeführt.

6.3.2.1 Multivariate Gaußklassifikation - Nur Filter

In diesem Abschnitt wird ein Klassifikationsverfahren beschrieben, welches auf multivariaten Normalverteilungen basiert und zusätzlich *keine* Datenoptimierungsverfahren aus dem Abschnitt 4.3.2 benötigt. Für die Datenaufbereitung kommen lediglich die Techniken aus dem Abschnitt 4.3.1 zum Einsatz.

Die Konfiguration der Filter ist im Folgenden beschrieben:

Filterung: Es findet keine Filterung der Messdaten statt. Die Differenz auf dem zu klassifizierenden Bauteil ermittelter Messwerte zu dem Mittelwert der letzten 80 Messungen werden mit Hilfe des Betrachtungspuffers (Abschnitt 4.3.1.4) beobachtet.

Datenanreicherung: Die Daten werden nicht angereichert.

Normierung: Die Normierung der Daten erfolgt mit der Funktion $N_{3\sigma}$ (siehe auch Gl. 4.4).

Rauschunterdrückung: Die Rauschunterdrückung geschieht wieder mit Hilfe der Normierung und des Betrachtungspuffers der letzten 80 Bauteile.

Die Klassifikationsergebnisse des auf die obige Art konfigurierten Optimierungsverfahrens sind im Diagramm 6.29 zu finden. Diese Ergebnisübersicht zeigt, dass in dieser Konfiguration das Optimierungsverfahren sowohl moderate Einsparungen erlaubt, als auch die Quote der unentdeckt verbleibenden schlechten Bauteile gemäßigt bleibt. In der Grafik 6.30 kann die Einzeltestübersicht begutachtet werden. Die Grafiken 6.31 bis 6.33 geben einen Überblick über die Precision-Recall Statistiken dieses Verfahrens wider.

Die gemachten Einsparungen (11.00%) dieses Verfahrens liegen zwar oberhalb des im Abschnitt 6.3.1.2 vorgestellten Experiments (3.19%), doch stieg mit den ausgelassenen Testausführungen auch die Quote der Escapes. So beträgt diese, beim aktuellen Verfahren bereits 0.27 Prozent.

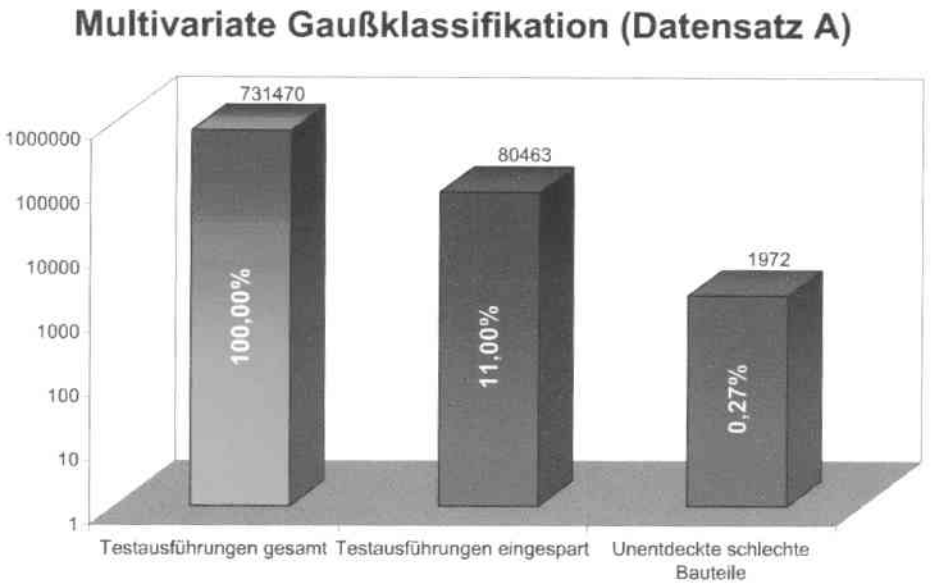


Abbildung 6.29: Multivariate Gaußklassifikation: Gesamtüberblick der Klassifikationsergebnisse

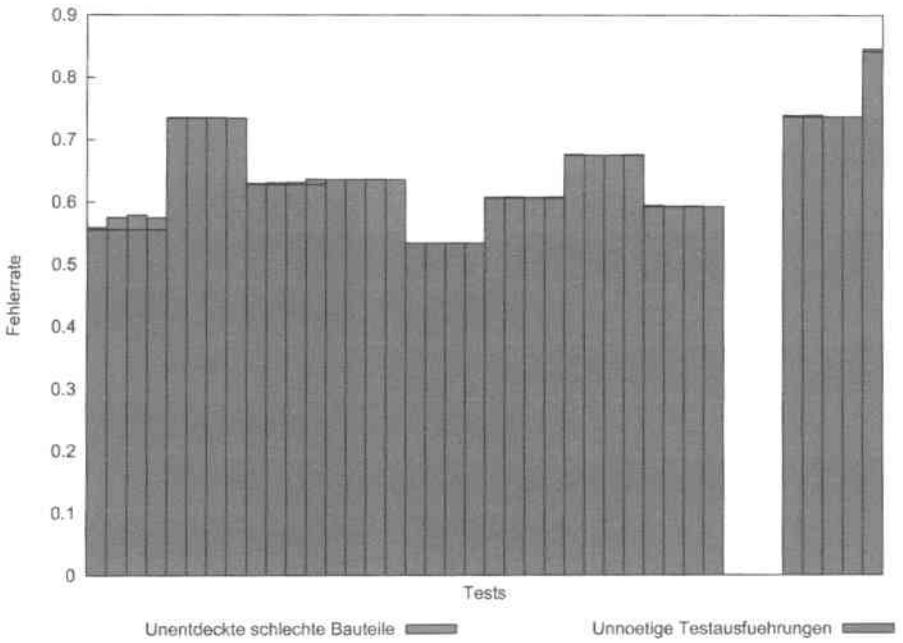


Abbildung 6.30: Multivariate Gaußklassifikation: Fehlerraten einzelner Tests vorhergesagter Testgruppen

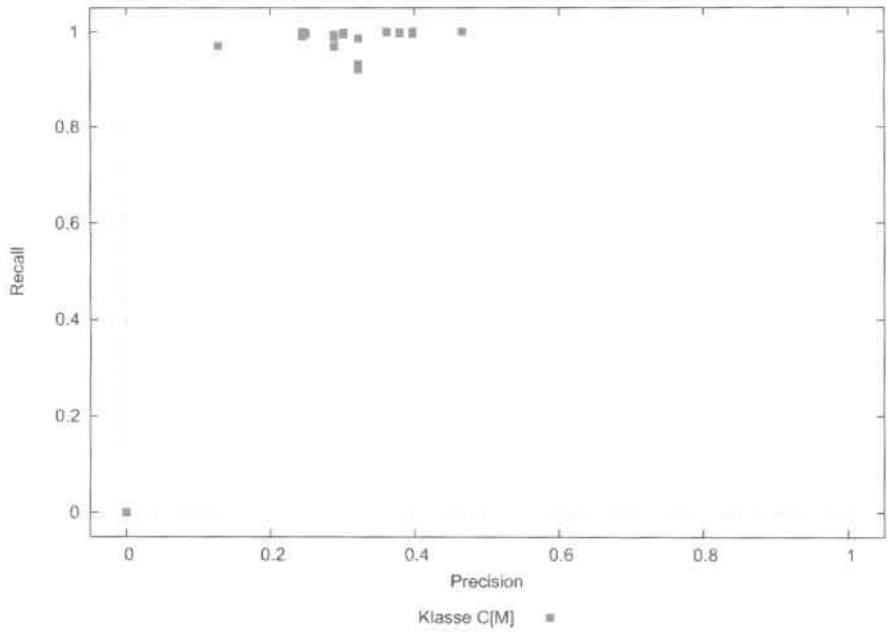


Abbildung 6.31: Multivariate Gaußklassifikation: Precision-Recall Statistik der Klasse C_M

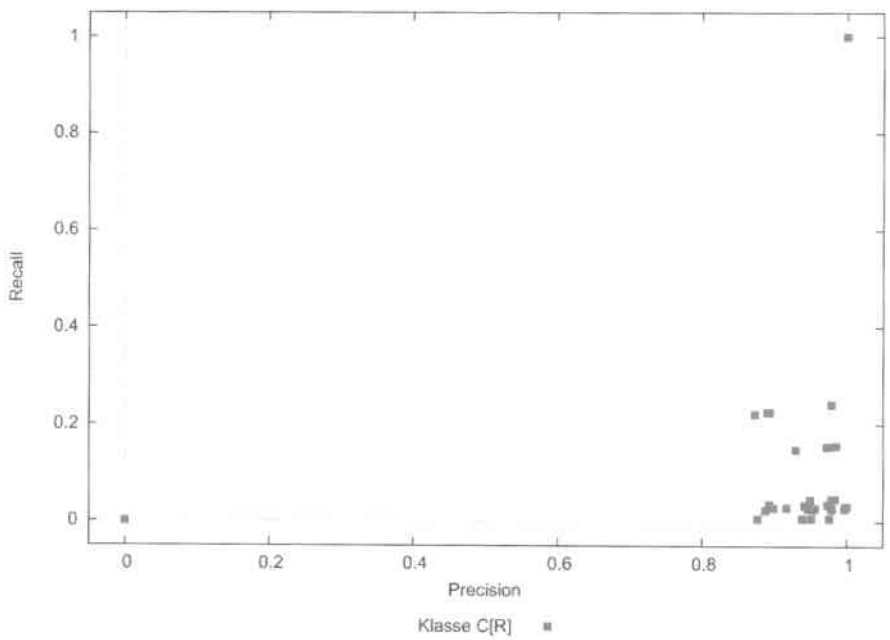


Abbildung 6.32: Multivariate Gaußklassifikation: Precision-Recall Statistik der Klasse C_R

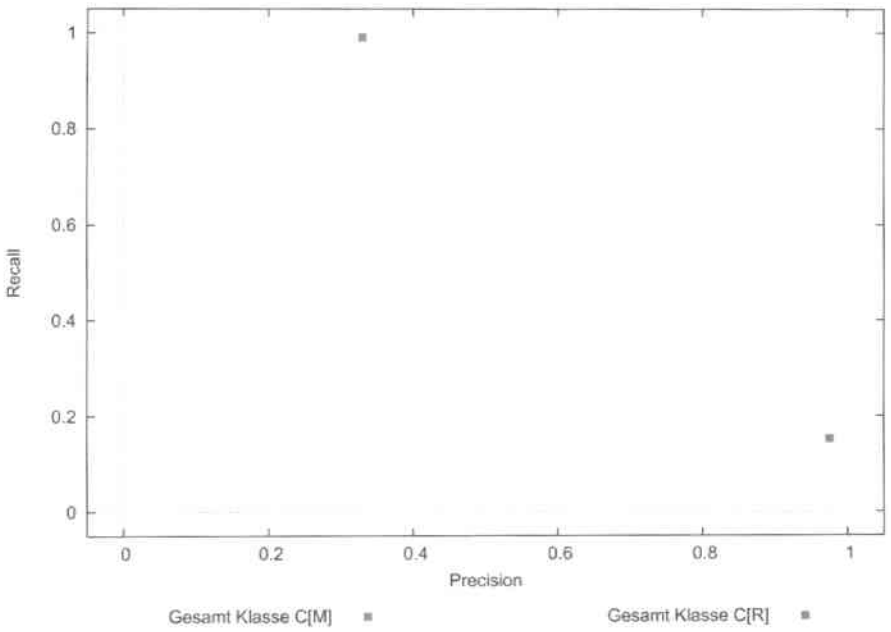


Abbildung 6.33: Multivariate Gaußklassifikation: Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

Eine mögliche Erklärung für diesen Anstieg könnte das Fehlen bestimmter Datenoptimierungsmechanismen sein. In der momentanen Konfiguration werden *alle*, auf dem untersuchten Bauteil bisher ermittelten Messwerte für die Erstellung eines Merkmalsvektors verwendet. Dadurch kann die Dimensionalität des Merkmalsraumes bedeutend ansteigen, und auch die für eine bestimmte Klassifikation unbedeutenden Merkmale auf den Klassifikationsalgorithmus störend wirken.

Wie man dem Diagramm 6.30 entnehmen kann, stammt der Großteil der übersehenen, schlechten Bauteile nur von einer Handvoll Tests. Es überrascht umso mehr, da diese Tests recht häufig auf Verdacht ausgeführt wurden, auch auf Bauteilen, auf welchen keine Notwendigkeit dazu bestand.

Die Bestätigung dafür, dass nur eine geringe Anzahl der Tests für die meiste Anzahl der unentdeckten, schlechten Bauteile verantwortlich ist, lässt nicht nur in der Einzeltestübersicht finden, sondern auch in den Precision-Recall Statistiken. Erkennbar ist das, an den wenigen, unterhalb der 1.0 liegenden Werten des Recalls der Klasse C_M und der Precision der Klasse C_R.

Die große Menge vermeidbarer Testausführungen ist ebenfalls in beiden Diagrammarten (Einzeltestübersicht und Precision-Recall) gut sichtbar. In den Precision-Recall Graphen wird die üppige Anzahl der unnötig ausgeführten Tests, durch die geringen Precisionwerte der Klasse C_M und Recallwerte der Klasse C_R gestützt.

Auswertung

Auch dieses Verfahren eignet sich nicht für einen Einsatz in der Praxis, denn der Anteil eingesparter Testausführungen ist zu klein und der Anteil der Escapes zu groß.

> Zieht man wieder die Kosten der vollen Testsequenzausführung auf allen zu evaluierenden Bauteilen zum Vergleich, ergibt sich, dass die *nicht* optimierten Testkosten bei 183.04 EUR liegen, die durch das Optimierungsverfahren eingesparten Testausführungen einen Gegenwert von 8.04 EUR haben, und die durch die nicht entdeckten Fehler entstehenden Zusatzkosten 19.72 EUR betragen. Damit überwiegen die Mehrkosten auch bei diesem Experiment die Einsparungen.

6.3.2.2 Multivariate Gaußklassifikation - Feature Map

Im vorhergehenden Experiment (Abs. 6.3.2.1) konnte festgestellt werden, dass das auf der multivariaten Gaußklassifikation aufbauende Optimierungsverfahren Schwierigkeiten hatte, die nicht gefilterten und nicht optimierten Messdaten korrekt zu klassifizieren. Aus diesem Grund wird ein weiteres Experiment mit dem gleichen Klassifikationsalgorithmus durchgeführt, die Konfiguration des Optimierungsverfahrens wird ähnlich angepasst, wie dies bereits im Abschnitt 6.3.1.2 geschehen ist.

Dieses Experiment verwendet die selbstorganisierenden Feature Maps, um die Qualität seiner Ergebnisse zu steigern. Für diesen Versuch wurde eine quadratische Topologie der selbstorganisierenden Feature Maps gewählt. Die Feature Map wurde mit insgesamt 100 Knoten versehen, angeordnet in einem 10×10 Gitter[9]. Die von implementierter Feature Map zur Verfügung gestellten Zusatzinformationen (siehe 4.3.2.1.1) werden durch das Klassifikationsverfahren, während der Berechnung der multivariaten Dichtefunktionen explizit ausgewertet und mitberücksichtigt.

Die Auswahl der Datenoptimierungsverfahren in diesem Experiment entpricht exakt der im Abschnitt 6.3.1.2 verwendeten Konfiguration.

In der Grafik 6.34 findet sich ein Überblick über die Ergebnisse der durchgeführten Klassifikation. Auf diesem Diagramm lässt sich schnell erkennen, dass die selbstorganisierenden Feature Maps im Zusammenspiel mit dem aktuell verwendeten Klassifikationsverfahren den Fehleranteil entscheidend reduzieren. Die Quote eingesparter Tests ist bei diesem Verfahren sogar höher, als bei dem k Nächste Nachbarn Klassifikator aus dem Abschnitt 6.3.1.2. Die Anzahl der unentdeckt gebliebenen schlechten Bauteile sinkt in diesem Experiment zwar nicht gänzlich auf Null, bleibt aber dennoch verschwindend klein.

[9] Die Größe des Netzes wurde in mehreren Versuchsreihen variiert, die Kantenlänge 10 erwies sich dabei als optimal.

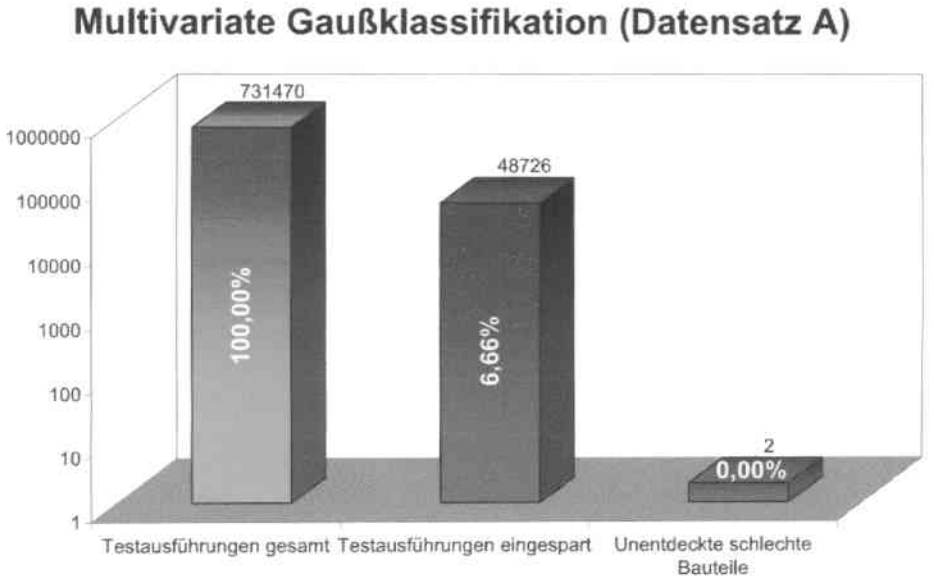


Abbildung 6.34: Multivariate Gaußklassifikation (mit Feature Map): Gesamtüberblick der Klassifikationsergebnisse

Die Einzelübersicht aller Tests, sowie die Precision-Recall Graphen folgen in den Abbildungen 6.35 bis 6.38.

Die Testeinzelübersicht (dargestellt in der Abbildung 6.35) ähnelt stark der Übersicht aus dem vorhergehenden Experiment (Abschnitt 6.3.2.1). Die roten Bereiche unentdeckter, schlechter Bauteile sind gewichen, die Anzahl der unnötigen Testausführungen dagegen leicht gestiegen.

Die Precision-Recall Graphen bringen die hohe Anzahl der unnötigerweise ausgeführten Tests zum Ausdruck. Der Recall der Klasse C_M liegt mit einer Ausnahme bei 1.0. Die Precision der Werte verteilt sich hingegen stark auf den Bereich zwischen 0.25 und 0.40. Sie unterstreicht damit den hohen Anteil der fälschlicherweise als ausführenswert klassifizierter Tests.

Die Precision der Klasse C_R, insofern diese überhaupt definiert ist, besitzt nur einen Datenpunkt dessen Wert ungleich 1.0 ist. Es ist anzunehmen, dass es sich genau bei diesem Test, um den Test mit den übersehenen, schlechten Bauteilen handelt. Umso verwunderlicher ist es, dass genau dieser Test seinen Recallwert nahe der Null hat. Es muss sich also um einen Test handeln, der beinahe durchgehend ausgeführt wird und nur an den Stellen, an welchen er einen Wert außerhalb seiner kritischen Grenzen misst, eingespart wird.

Auswertung

Ungeachtet der beiden Escapes, scheint das in diesem Experiment vorgestellte Optimierungsverfahren ähnlich, wie das Verfahren aus dem Experiment 6.3.1.2

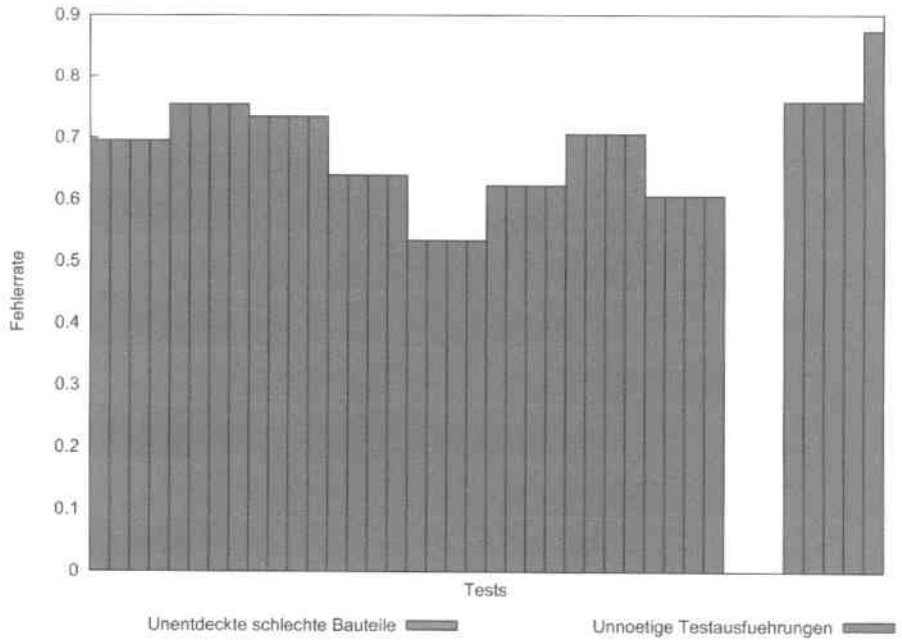


Abbildung 6.35: Multivariate Gaußklassifikation (mit Feature Map): Fehlerraten einzelner Tests vorhergesagter Testgruppen

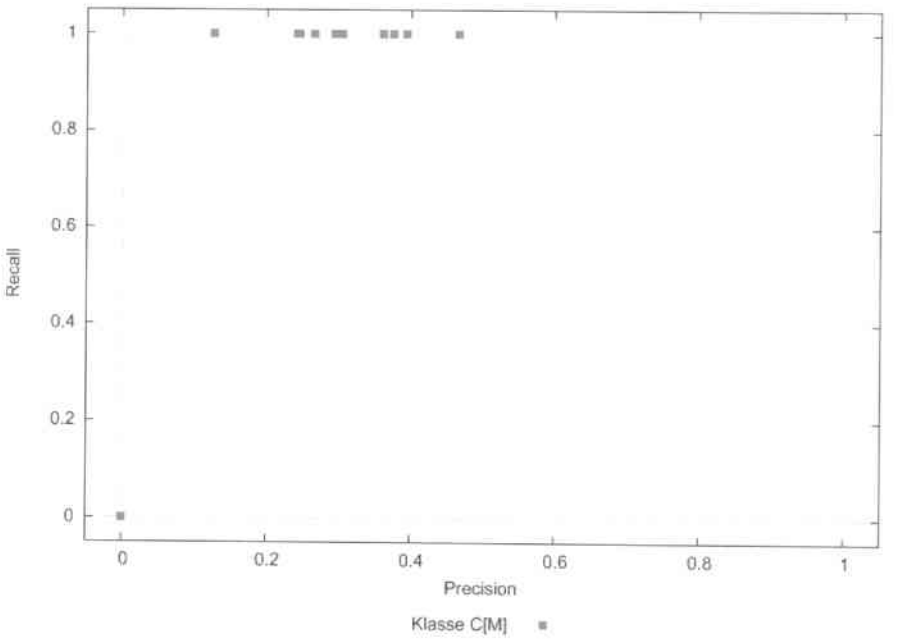


Abbildung 6.36: Multivariate Gaußklassifikation (mit Feature Map): Precision-Recall Statistik der Klasse C_M

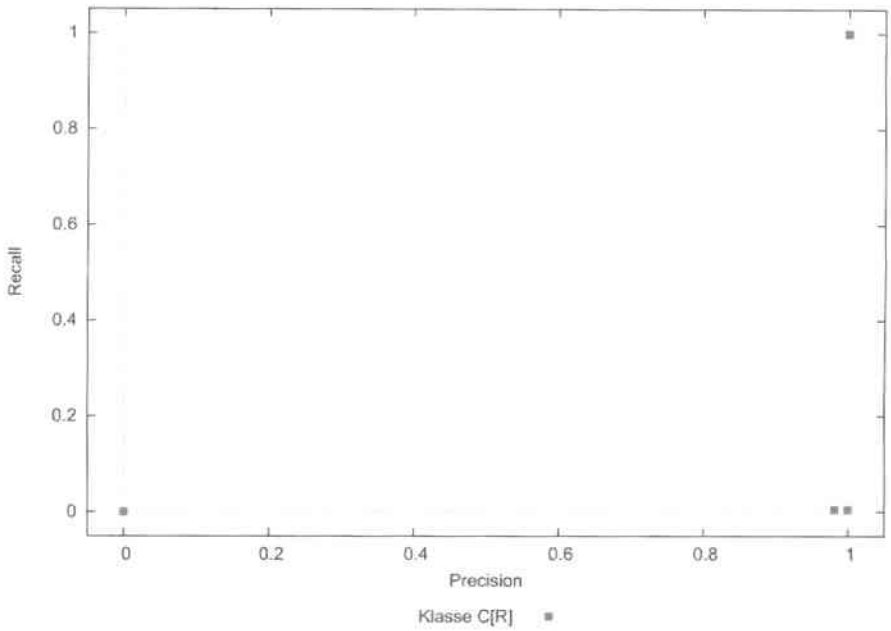


Abbildung 6.37: Multivariate Gaußklassifikation (mit Feature Map): Precision-Recall Statistik der Klasse C_R

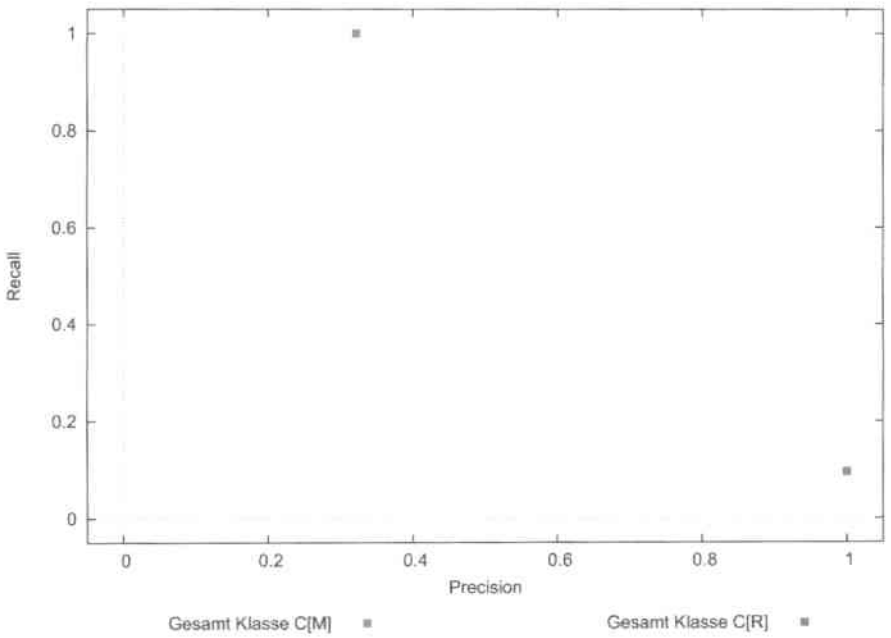


Abbildung 6.38: Multivariate Gaußklassifikation (mit Feature Map): Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

vom Einsatz eines Datenoptimierungsverfahrens deutlich profitieren zu können. Einerseits, könnte der positive Einfluss durch die „interne" Funktionsweise der Feature Map[10] und andererseits, durch die Dimensionsreduktion herbeigeführt sein.

Die eingesetzte Datenoptimierung scheint letztendlich ausschlaggebend in Bezug auf die Qualität der Vorhersagen eines Klassifikators zu sein, und so lässt sich anhand einer Rechnung feststellen, dass sich auch die aktuell verwendete Konfiguration des Experiments für den Einsatz in der Praxis gut eignet.

> Die Kosten für das Testen der 19268 evaluierten Bauteile des Datensatzes A, ohne die Anwendung einer Optimierung würden sich auf 183.05 EUR belaufen. Unter Einsatz des in diesem Experiment vorgestellten Optimierungsverfahrens könnten 4.87 EUR eingespart werden. Die Mehrkosten durch die entstandenen Escapes würden bei 0.02 EUR liegen. Durch das Optimierungsverfahren liessen sich also etwa 2.65% der Gesamttestkosten einsparen.

Die Einsparungen des Verfahrens würden also durchaus das Risiko unentdeckter, schlechter Bauteile überwiegen und somit den Einsatz dieses Verfahren in der Halbleiterindustrie ermöglichen.

6.3.3 Multilayer Perzeptron

Das dritte, im Rahmen statischer Klassifikatoren, vorgestellte Verfahren basiert auf einem Multilayer Perzeptron. Die Implementierungsdetails des Perzeptrons finden sich im Abschnitt 4.4.2.3. Im Folgenden wird ein Experiment mit diesem Klassifikator auf den Evaluationsdaten des Datensatzes C vorgestellt.

Aufgrund der bisher guten Erfahrungen mit Datenoptimierungsverfahren, wird auch in diesem Experiment ein solches Verfahren eingesetzt. Die an den Klassifikator übergebene Daten werden zuerst, durch die Filter vorverarbeitet werden, um anschließend durch das *k-means* Clusteringverfahren (*k-means*) zu einer kleineren Anzahl möglicher Merkmalsausprägungen zusammengefasst zu werden.

Der grundsätzliche Aufbau des Perzeptrons entspricht der Darstellung im Abschnitt 4.4.2.3. Zusätzlich wird die in dieser Arbeit entwickelte Heuristik zur Steigerung des Lernfortschritts eingesetzt. Die weiteren Einstellungen[11] des Multilayer Perzeptrons, sowie des *k-means* Clusteringverfahrens lassen sich der folgenden Übersicht entnehmen:

Lernepochen: Der Klassifikator wurde über mindestens 10000 Epochen trainiert.

[10] Die besondere Funktionsweise der selbstorganisierenden Feature Maps, versucht die Distanzen und die Nachbarschaftsbeziehungen der ursprünglichen Merkmalsvektoren zu erhalten.

[11] Die Einstellungen basieren auf einer Reihe kleiner, vorhergehender Versuche, in welchen die verwendeten Werte als Bestwerte ermittelt werden konnten.

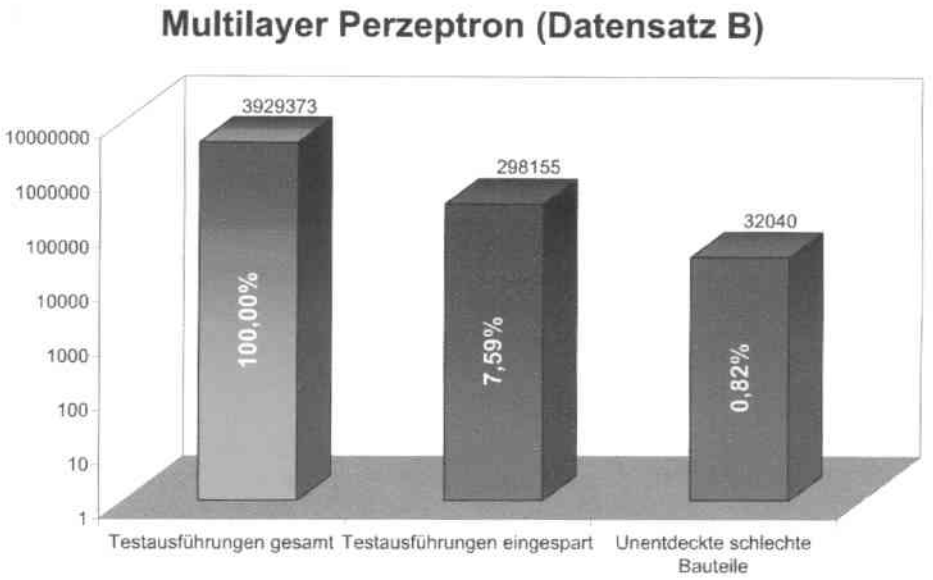


Abbildung 6.39: Multilayer Perzeptron (mit Clustering): Gesamtüberblick der Klassifikationsergebnisse

Lernrate: Die Lernrate wurde auf den Wert 0.25 festgelegt.

Lernmodus: Die Verbindungsgewichtungen zwischen den einzelnen Neuronen wurden nach der Präsentation jedes einzelnen Trainingsvektors gelernt[12].

***k-means* Cluster:** Die Anzahl der zu bildenden Cluster wurde auf 64 festgesetzt.

***k-means* Iterationen:** Die maximale Anzahl der Clusteringiterationen war auf 100 begrenzt.

Die Einstellungen der Filter wurden vom Experiment 6.3.1.1 exakt übernommen.

Die Ergebnisse dieser Klassifikation lassen sich in der Abbildung 6.39 begutachten. Auf dem dargestellten Diagramm lässt sich erkennen, dass der Anteil der zu Einsparung freigegebener Tests grob dem Anteil des Experiments 6.3.2.2 entspricht. Die Quote der übersehenen, schlechten Bauteile ist jedoch mit 0.82 Prozent unverhältnismäßig hoch. In den folgenden Grafiken 6.40 bis 6.43 lassen sich die Einzeltestübersicht und die Precision-Recall Statistiken finden.

Der Einzeltestübersicht lassen sich auch in diesem Experiment viele unnötige Testausführungen entnehmen. Ebenso ist der Anteil der Escapes nicht zu übersehen. Zwei als Dreieck geformte Punktwolken innerhalb der Precision-Recall Statistiken der Klassen C_M und C_R stützen die aus der Einzeltestübersicht gezogenen Schlüsse.

[12] In den durchgeführten Experimenten hat sich der Batchmodus nicht bewährt.

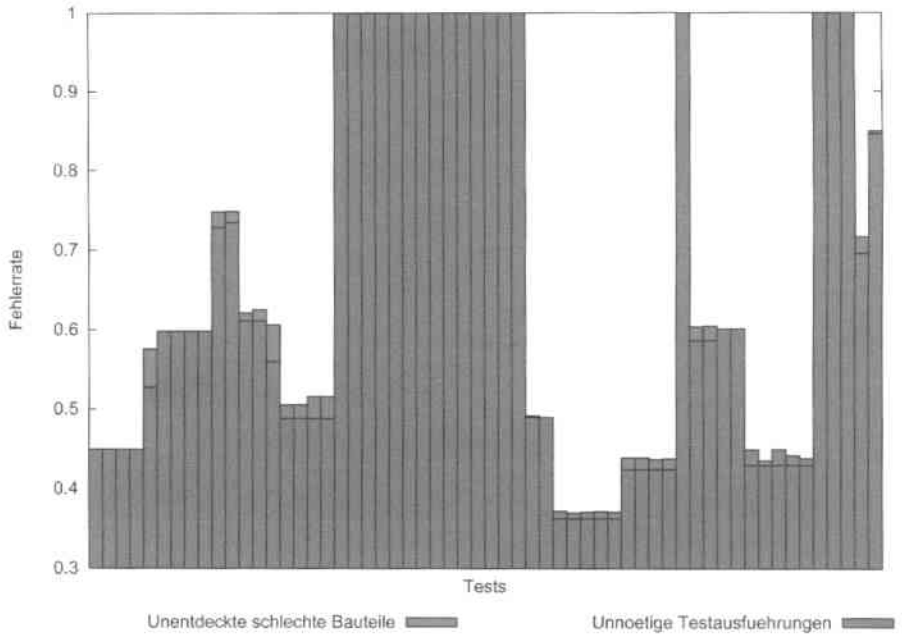


Abbildung 6.40: Multilayer Perzeptron (mit Clustering): Fehlerraten einzelner Tests vorhergesagter Testgruppen

Die Punktwolke der Klasse C_M in der linken oberen Ecke, bedeutet einerseits eine geringe Precision $(0.0 - 0.65)$ und somit häufige Klassifikation von einsparbaren Tests als ausführenswert. Andererseits, weist die große Streuung der Recallwerte $(0.4 - 1.0)$ auf viele Escapes hin.

Die Punktwolke der Klasse C_R bildet hingegen einen Dreieck in der rechten unteren Ecke. Dies ist wiederum ein Zeichen dafür, dass aufgrund des kleinen Recallwertes $(0.0 - 0.5)$ viele Tests unnötigerweise ausgeführt wurden. Der zwischen 0.65 und 1.0 schwankende Precisionwert, deutet seinerseits die Auslassung der Tests an, die auf dem vorhergesagten Bauteil Messwerte außerhalb ihrer kritischen Grenzen festgestellt hätten und deshalb unbedingt ausgeführt werden mussten.

Ein derartiges Klassifikationsverhalten gehörte beim Multilayer Perzeptron leider zum Regelfall. Die Problematik des Perzeptrons wird am schnellsten bewusst, wenn die selbstentwickelte Heuristik zur Beschleunigung des Lernverhaltens abgeschaltet wird (siehe auch Abbildung 4.6).

Viele Variationen der Anzahl und der Größe der versteckten Schichten wurden ausprobiert, die Quelle und die Art (mit und ohne Einsatz des Datenoptimierungsverfahrens) der Daten wurden modifiziert, die Anzahl der Lernepochen wurde probeweise auf 100000 erhöht, dennoch konnte nicht erreicht werden, dass der Klassifikator einen stabilen Zustand einnahm. Stattdessen sprang die Güte der auf den Validierungsdaten gemachten Voraussagen - in Abhängigkeit von der verwendeten Konfiguration - zwischen 45% und 95%.

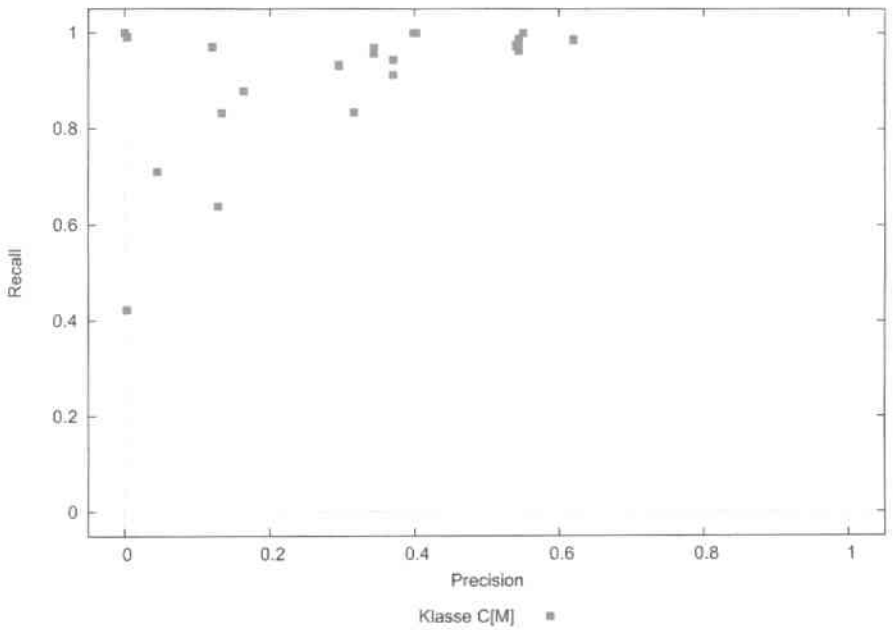


Abbildung 6.41: Multilayer Perzeptron (mit Clustering): Precision-Recall Statistik der Klasse C_M

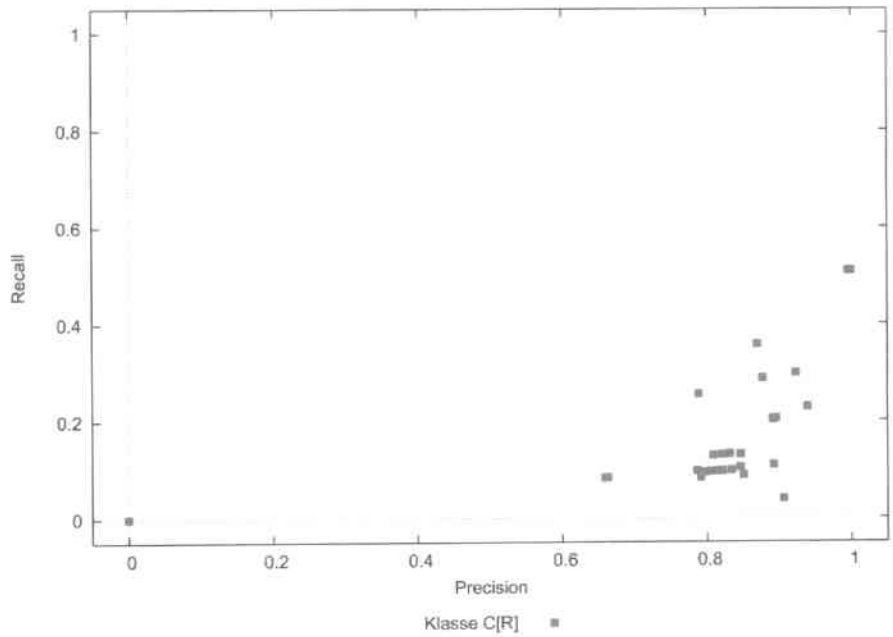


Abbildung 6.42: Multilayer Perzeptron (mit Clustering): Precision-Recall Statistik der Klasse C_R

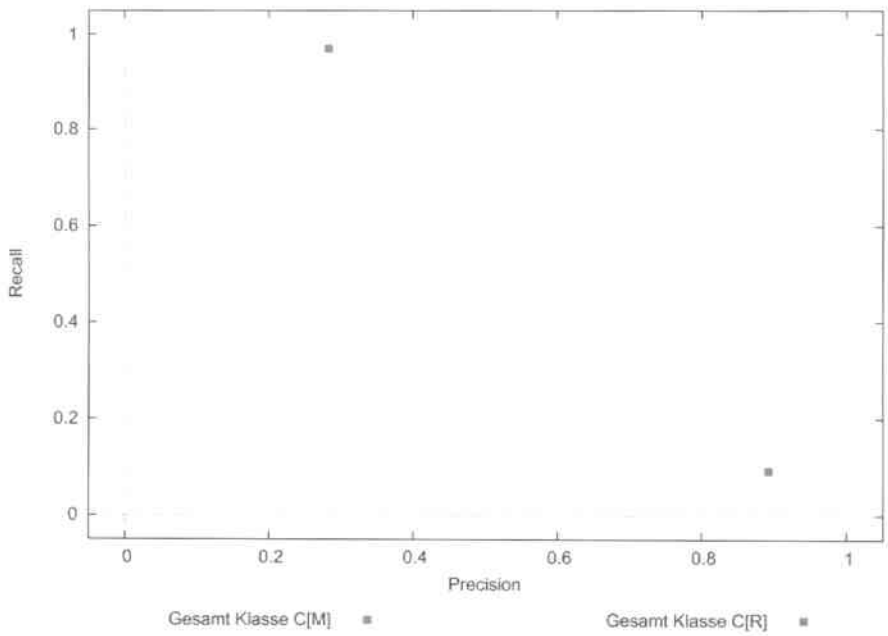


Abbildung 6.43: Multilayer Perzeptron (mit Clustering): Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

Der Einsatz der vorgestellten Heuristik fixierte zwar den Perzeptron auf die bestmöglichen Trainingsergebnisse, die Qualität der Vorhersagen auf großen Datenmengen, blieb jedoch auch in dieser Konstellation mit dem vorgestellten Experiment vergleichbar. Die Vergrößerung der Beispieldatenmenge[13] durch Replikation war ebenfalls nicht vom Erfolg gekrönt.

Mögliche Ursachen für das Scheitern dieses Klassifikators können in der hohen Dimensionalität der vorhergesagten Daten vermutet werden. Es würden wahrscheinlich deutlich mehr echte (nicht replizierte) Beispieldaten benötigt werden, um die Vorhersagequalität dieses Klassifikators über die der anderen Verfahren zu heben.

Auswertung

Die dargestellte Problematik, sowie die präsentierten Resultate des durchgeführten Experiments, lassen dieses Klassifikationsverfahren nicht für einen Einsatz in der Praxis zu, da die mit der Verwendung des Klassifikationsalgorithmus verbundenen Risiken sich im Augenblick nicht abschätzen lassen.

[13] Vor allem der Merkmalsvektoren auf den Bauteilen, auf welchen der vorherzusagende Test Messwerte außerhalb seiner kritischen Grenzen ermittelte

6.4 Dynamische Klassifikationsverfahren

Dieser Abschnitt befasst sich mit dem einzigen, im Sinne der vorliegenden Arbeit *dynamischen Klassifikationsverfahren*. Ein dieser Kategorie angehörendes Verfahren muss in der Lage sein, zusätzlich zu dem, aus der Lernphase gewonnenen Wissen und den, aktuell auf dem Chip gemessenen Daten, auch noch die zeitliche Entwicklung der Messungen vorhergehender Bauteile, während der Klassifikation zu berücksichtigen. Diese Anforderungen erfüllt in dieser Arbeit nur ein, auf dem Hidden-Markov-Modell (*HMM*) aufsetzender Klassifikationsalgorithmus. Ein Experiment mit diesem Klassifikationsverfahren wird im folgenden Abschnitt präsentiert.

6.4.1 Hidden-Markov-Modell basierte Klassifikation

Der Gegenstand dieses Abschnittes besteht in der Präsentation eines Experiments, in welchem die auf einem Hidden-Markov-Modell basierende Klassifikation eingesetzt wird. Die für das Experiment verwendeten Daten, werden dem Datensatz B entnommen.

Eine ausführliche Darstellung der generellen Modellierung, der in dieser Arbeit eingesetzten HMM Klassifikation, lässt sich bereits dem Abschnitt 4.4.3.1 entnehmen und wird an dieser Stelle nicht wiederholt. Die konkrete Konfiguration des eingesetzten Modells, sowie der angewandten Datenoptimierung findet sich hingegen in der folgenden Auflistung:

Topologie: Die Topologie des Hidden-Markov-Modells lässt sich als ein Ring mit dem Bakis-Übergangsschema beschreiben.

Zustände: Alle zehn existierenden Zustände kommen als Startzustände alle gleichermaßen in Frage.

Emissionen: Jeder Zustand emittiert den, auf einem Bauteil beobachtbaren Merkmalsvektor mit einer, vorher auf den Trainingsdaten gelernten Wahrscheinlichkeit.

Emittierte Merkmalsvektoren: Jeder emittierte Merkmalsvektor befindet sich in Kenntnis des für ihn optimalen Klassifikationsalgorithmus.

Klassifikatoren: Wie den vorhergehenden Experimenten zu entnehmen ist, kommen nur zwei, der zur Verfügung stehenden Klassifikationsalgorithmen in die engere Wahl: k Nächste Nachbarn Klassifikation und Multivariate Gaußklassifikation. Beide Klassifikatoren sind auf die gleiche Art konfiguriert, wie in den Experimenten 6.3.1.2 und 6.3.2.2 dargestellt.

Datenoptimierung: Beide unter letztem Punkt genannten Klassifikatoren benutzen zur Datenoptimierung die Learning Vector Quantization. Die Einstellungen des LVQ Verfahrens stimmen mit der im Abschnitt 6.3.1.2 verwendeten Konfiguration überein.

Filterung: Für die Filterung werden auf die gleiche Weise, wie im Experiment 6.3.1.1 die Korrelationskoeffizienten (Gl. 4.8) eingesetzt.

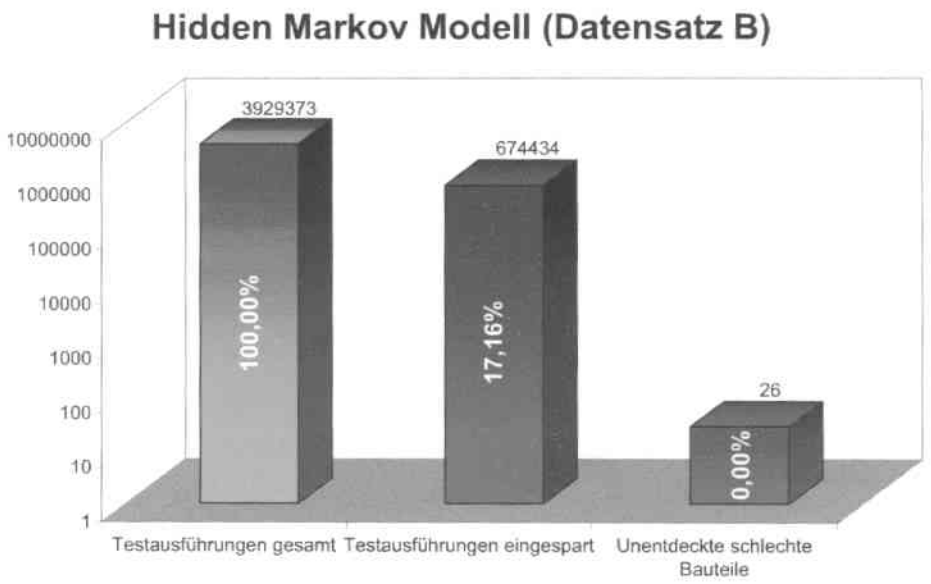


Abbildung 6.44: Hidden-Markov-Modell Klassifikation: Gesamtüberblick der Klassifikationsergebnisse

Datenanreicherung: Die Anreicherung der Daten erfolgt ebenfalls mit Hilfe der Korrelationskoeffizienten.

Normierung: Als Normierungsfunktion dient erneut die Funktion N_1 (Gleichung 4.2).

Rauschunterdrückung: Statisches Rauschen wird mit Hilfe eines Betrachtungspuffers über die letzten 80 Bauteile entfernt.

Die Klassifikationsergebnisse dieses Verfahrens sind in der Abbildung 6.44 dargestellt. Zu erkennen sind auf dem Diagramm einerseits, der vergleichsweise hohe Anteil an eingesparten Tests und andererseits, eine relativ geringe Quote der nicht entdeckten Fehler. Der Anteil ausgelassener Tests ist mit 17.16% in diesem Experiment knapp dreimal so hoch, wie beim Experiment 6.3.2.2 auf dem Datensatz A. Das Risiko von Escapes ist im aktuellen Verfahren ebenfalls nicht Null, dennoch ist es vergleichsweise niedrig. Die weiteren Diagramme 6.45 bis 6.48 erlauben einen tieferen Einblick in den Ablauf dieses Experiments.

Ungeachtet der relativ hohen Ersparnisse, ist in der Einzeltestübersicht die Menge unnötigerweise ausgeführter Tests ganz deutlich zu erkennen. Mehrere Tests wurden sogar beinahe permanent ausgeführt, ohne, dass eine Notwendigkeit dazu bestand. Dieser Eindruck wird, durch die abermals geringen Precisionwerte (0.0 – 0.62) der Klasse C_M unterstrichen. Dem kleinen Anteil der Escapes entsprechend, liegt der Recall dieser Klasse annähernd bei 1.0.

Die Klasse C_R offenbart, durch die beiden sichtbaren Extreme, dass es sowohl einsparbare Testausführungen gab, die jedoch nie als solche erkannt wurden, als auch, dass manche Tests mit hoher Korrektheit eingespart werden konnten. Der

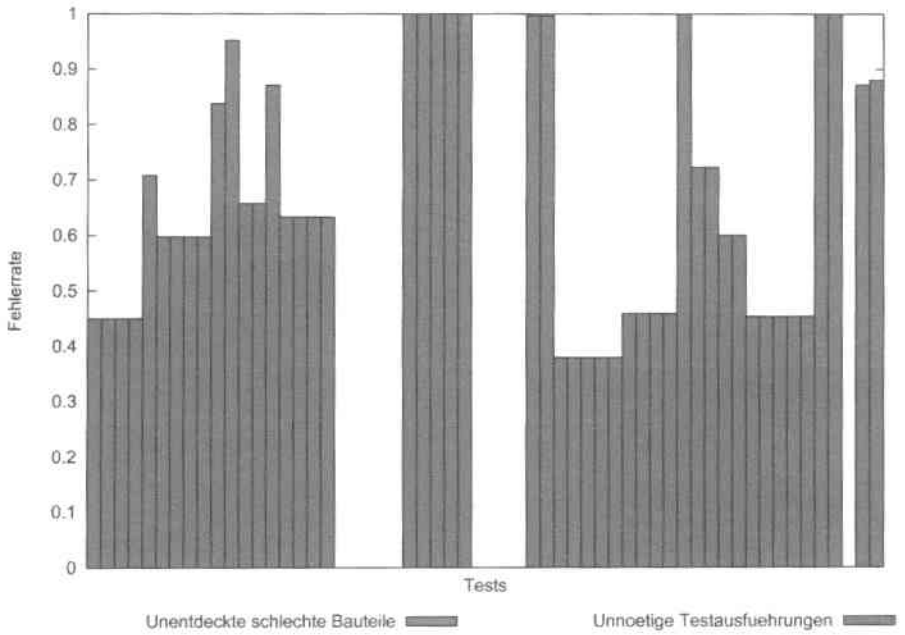


Abbildung 6.45: Hidden-Markov-Modell Klassifikation (mit LVQ): Fehlerraten einzelner Tests vorhergesagter Testgruppen

Gesamtübersicht der Precision-Recall Statistiken lässt sich entnehmen, dass die Anzahl der nicht eingesparten Tests um ein Vielfaches höher war, als die Anzahl der korrekt ausgelassenen Tests.

Auswertung

Das im aktuellen Experiment vorgestellte Optimierungsverfahren, erlaubt zwar hohe Einsparungen bei einem annehmbar kleinen Anteil an unentdeckten, schlechten Bauteilen, doch schöpft es das in den Daten vorhandene Einsparungspotenzial bei weitem nicht aus. Die erzielten Ergebnisse erlauben diesem Verfahren dennoch auch in dieser Konfiguration einen praktischen Einsatz in der Halbleiterindustrie.

> Die vollen Testkosten betragen auf den Evaluationsdaten des Datensatzes B 676.55 EUR. Die durch das Optimierungsverfahren eingesparten Testkosten liegen dagegen bei 67.44 EUR und die Mehrkosten, verursacht durch die übersehenen, schlechten Bauteile bei 0.26 EUR. Somit können auf den 69747 vorhergesagten Bauteilen dieses Datensatzes 67.18 EUR bzw. 9.93 Prozent der Gesamttestkosten eingespart werden.

6.5 Sonstige Verfahren

Im Rahmen dieser Arbeit wurde ein Klassifikationsverfahren nachgebaut, welches sich in keine der vorhergehenden Kategorien einordnen lässt. Es handelt

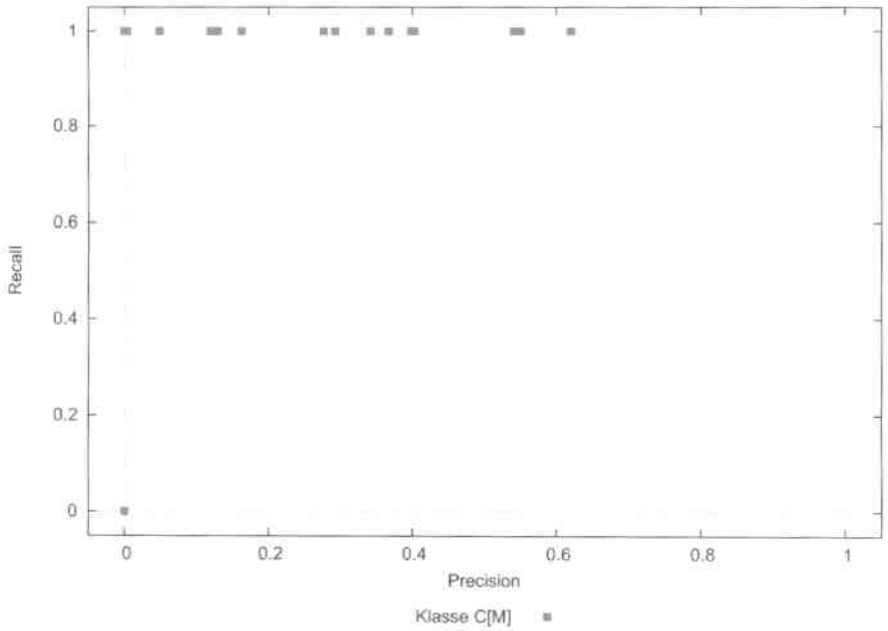


Abbildung 6.46: Hidden-Markov-Modell Klassifikation (mit LVQ): Precision-Recall Statistik der Klasse C_M

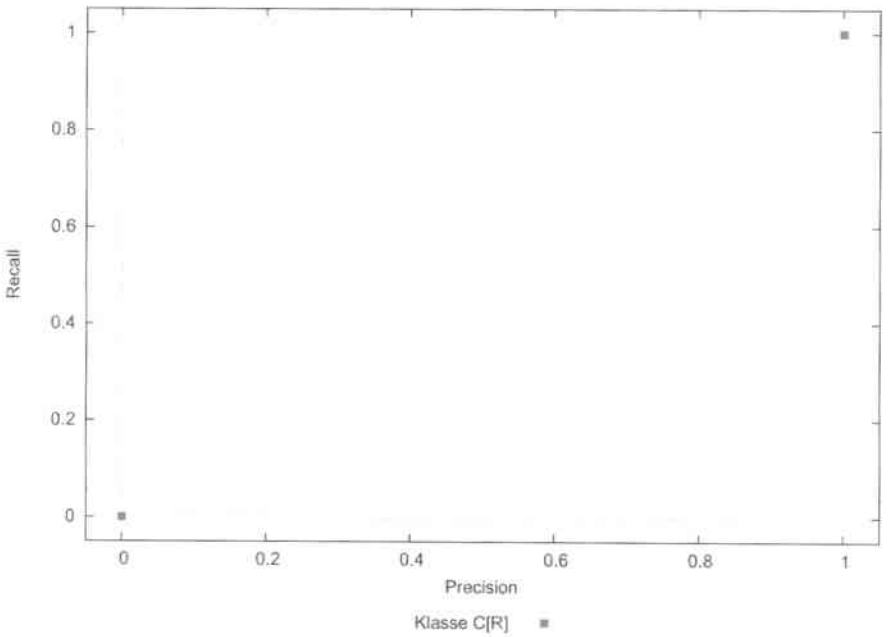


Abbildung 6.47: Hidden-Markov-Modell Klassifikation (mit LVQ): Precision-Recall Statistik der Klasse C_R

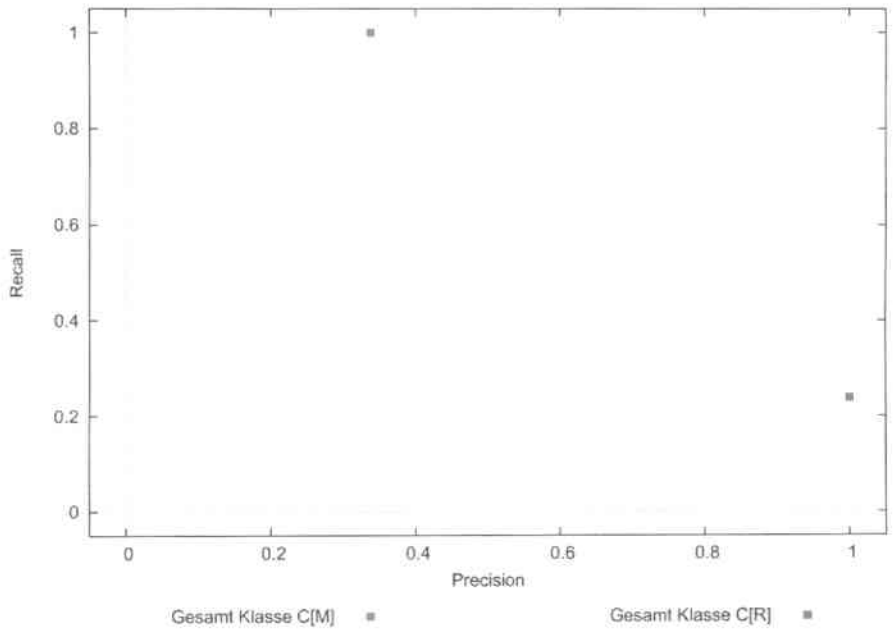


Abbildung 6.48: Hidden-Markov-Modell Klassifikation (mit LVQ): Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

sich hierbei um eine, auf dem Prozessfähigkeitsindex C_{pk} basierende Klassifikationstechnik (siehe auch Abschnitt 4.4.4.1), welche in der Halbleiterindustrie zum Entstehungszeitpunkt der vorliegenden Arbeit tatsächlich erprobt wurde.

Das vorgestellte Verfahren zeichnet sich hauptsächlich dadurch aus, dass es sowohl das dem Lernvorgang entnommene Wissen, als auch die Entwicklung der letzten l Messungen für die Entscheidungsfindung berücksichtigt. Überaschenderweise finden jedoch gerade die auf dem aktuell analysierten Bauteil gesammelten Informationen, während der Klassifikation desselben Bauteils *keine* Beachtung.

6.5.1 Prozessfähigkeitsindex C_{pk} basierte Klassifikation

Dieser Abschnitt widmet sich der Durchführung eines Experiments, in welchem ein, auf dem C_{pk} Index aufbauendes Klassifikationsverfahren (siehe Abschnitt 4.4.4.1) zum Einsatz kommt. Die für die Durchführung dieses Experiments benötigten Daten entstammen dem Datensatz A.

Ähnlich, wie bei den trivialen Klassifikationsverfahren (siehe Abschnitt 6.2), werden auch hier, bedingt durch die Funktionsweise des Klassifikationsverfahrens keine Datenoptimierung benötigt.

Ein Gesamtüberblick über die Klassifikationsergebnisse dieses Verfahrens wird im Diagramm 6.49 gegeben. Diesem Diagramm lässt sich entnehmen, dass

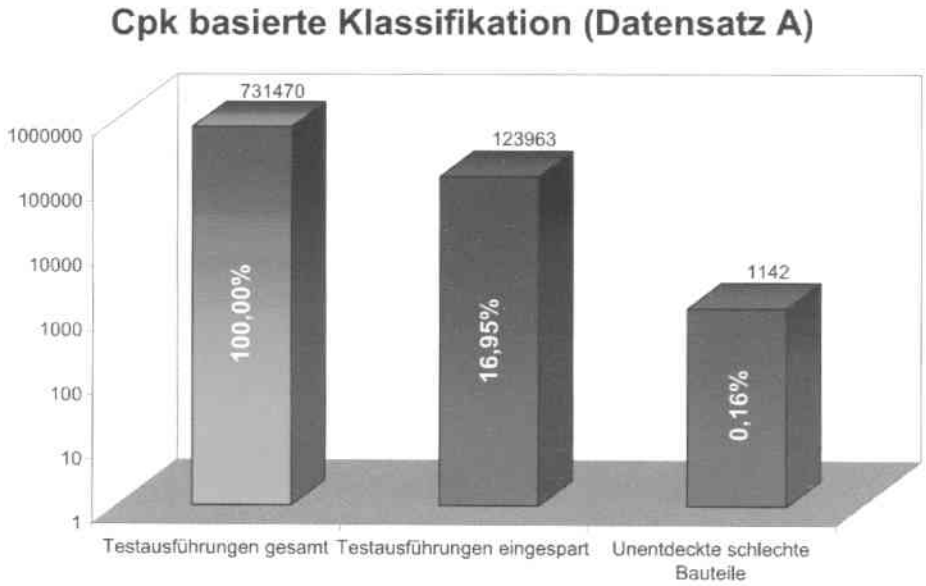


Abbildung 6.49: C_{pk} basierte Klassifikation: Gesamtüberblick der Klassifikationsergebnisse

die Quote unentdeckt gebliebener schlechter Bauteile (0.16%) im Vergleich zu dem Anteil der eingesparten Tests (16.95%) verhältnismäßig gering ausfällt. Weitere Informationen können den Abbildungen 6.50, 6.51, 6.52 und 6.53 entnommen werden.

Der Überblick der Klassifikationsfehler einzelner Tests in der Grafik 6.50 offenbart weiterhin, dass schlechte Bauteile hauptsächlich auf einigen wenigen Tests bzw. Testgruppen unentdeckt blieben. Die Menge der unnötigen Testsausführungen ist im Durchnitt relativ hoch.

Betrachtet man die Precision-Recall Statistik der Klasse C_M, lässt es sich schnell erkennen, dass die Precision dieser Klasse sich ausschließlich in dem unteren Bereich des Intervalls, zwischen 0.0 und 0.15 aufhält. Die Recallwerte liegen jedoch in dem oberen Bereich, zwischen 0.70 und 1.0. Damit lässt sich die hohe Anzahl der unnötig ausgeführten Tests erklären.

Wird nun der Precision-Recall Graph der Klasse C_R untersucht, so können hier die Precisionwerte zwischen 0.95 und 1.0 beobachtet werden. Da gerade die von 1.0 weiter entfernten Datenpunkte für die Anzahl der unentdeckt gebliebenen schlechten Bauteile verantwortlich sind, lässt sich so die relativ kleine Zahl an tatsächlich aufgetretenen Escapes bestätigen. Die streuenden Recallwerte im Bereich von 0.0 bis 0.5 bringen hingegen ein weiteres Mal die häufigen und unnötigen Testausführungen zum Ausdruck.

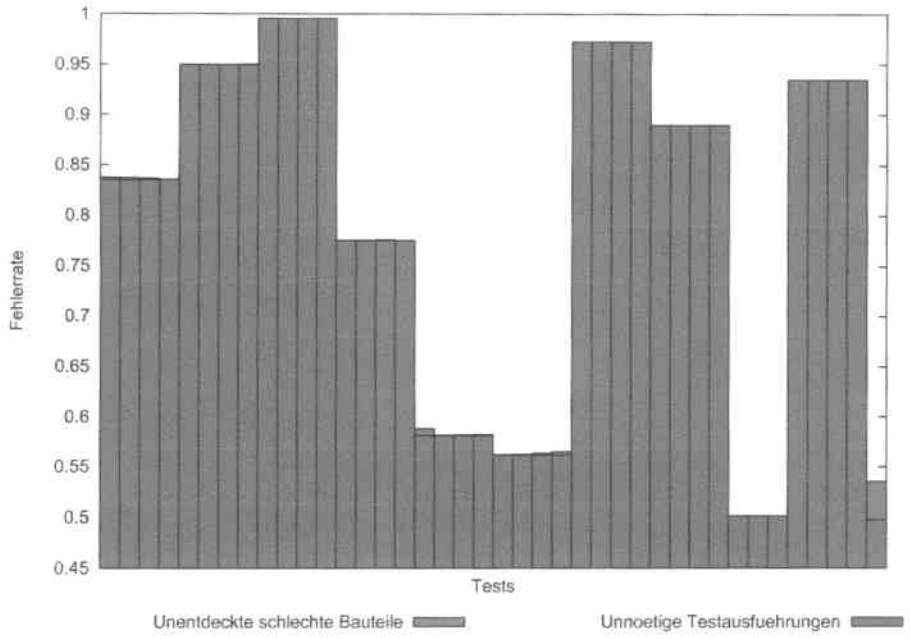


Abbildung 6.50: C_{pk} basierte Klassifikation: Fehlerraten einzelner Tests vorhergesagter Testgruppen

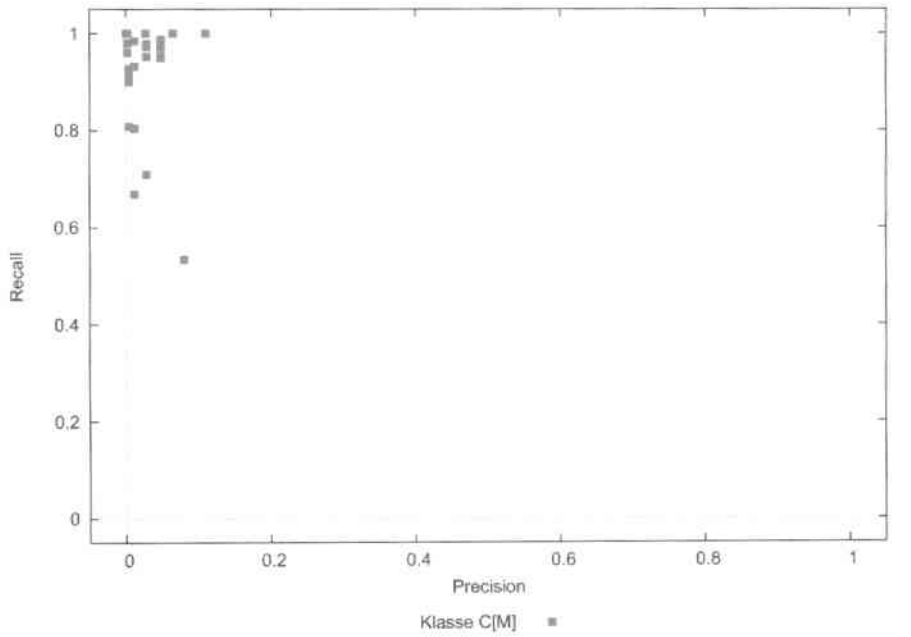


Abbildung 6.51: C_{pk} basierte Klassifikation: Precision-Recall Statistik der Klasse C_M

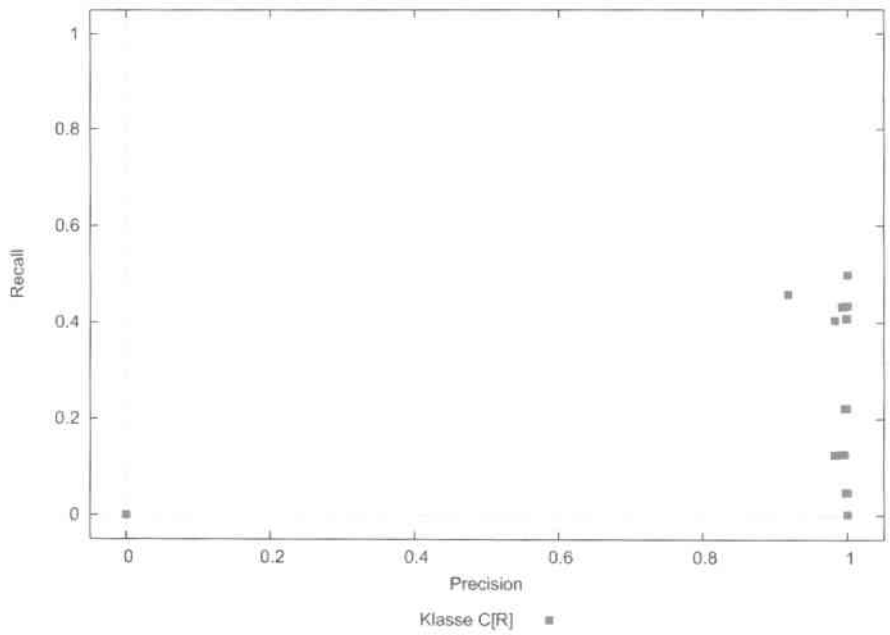


Abbildung 6.52: C_{pk} basierte Klassifikation: Precision-Recall Statistik der Klasse C_R

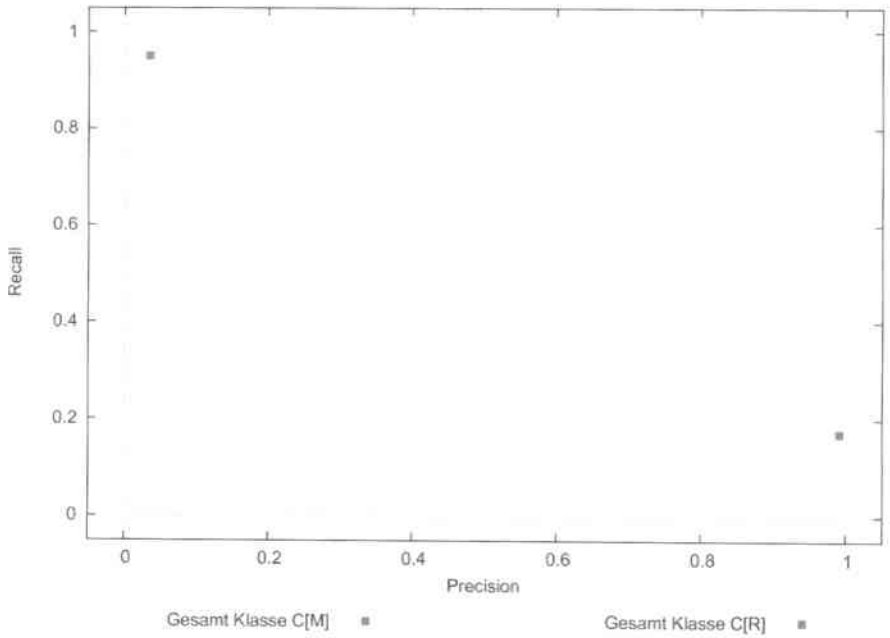


Abbildung 6.53: C_{pk} basierte Klassifikation: Gesamtübersicht von Precision und Recall der beiden Klassen C_M und C_R

Auswertung

Zusammengefasst lässt sich anhand der folgenden einfachen Gegenüberstellung feststellen, dass dieses Verfahren zwar für den Einsatz in der Praxis geeignet sein kann, vom tatsächlichen Einsatz allerdings abzuraten ist, solange die Kosten für einen Escape wesentlich teuerer als eine (unnötige) Testausführung sind. Die Gründe dafür liegen in der relativ hohen Anzahl an übersehenen, schlechten Bauteilen und, wie nachfolgend zu sehen sein wird, in dem geringen wirtschaftlichen Gewinn. Weitere Gründe wurden bereits unter 2.2 aufgezählt.

> Die Gesamttestkosten ohne den Einsatz eines Optimierungsverfahrens betragen für die 19268 evaluierten Bauteile des Datensatzes A 183.05 EUR. Der Einsatz des im aktuellen Experiment vorgestellten Optimierungsverfahrens ermöglichte eine Ersparnis in Höhe von 12.40 EUR. Von diesen Einsparungen müssen allerdings die durch die Escapes verursachten Mehrkosten in Höhe von 11.42 EUR wieder abgezogen werden. Damit würde dieses Optimierungsverfahren zwar weniger Zusatzkosten verursachen als Testkosten einsparen, doch wäre der Gewinn relativ niedrig, und das Risiko eines Escapes verhältnismäßig hoch.

Kapitel 7

Fazit

In der vorliegenden Arbeit konnte auf Basis echter Chipmessdaten dreier weltweit agierenden Chiphersteller gezeigt werden, dass die Gesamtkosten des Testvorganges um etwa 10 Prozent gesenkt werden können. Bei einer Produktion von 100 Mio. Chips pro Monat, kann ein Chiphersteller damit monatlich über 100000 EUR an Testkosten einsparen (siehe auch Diagramm 7.1).

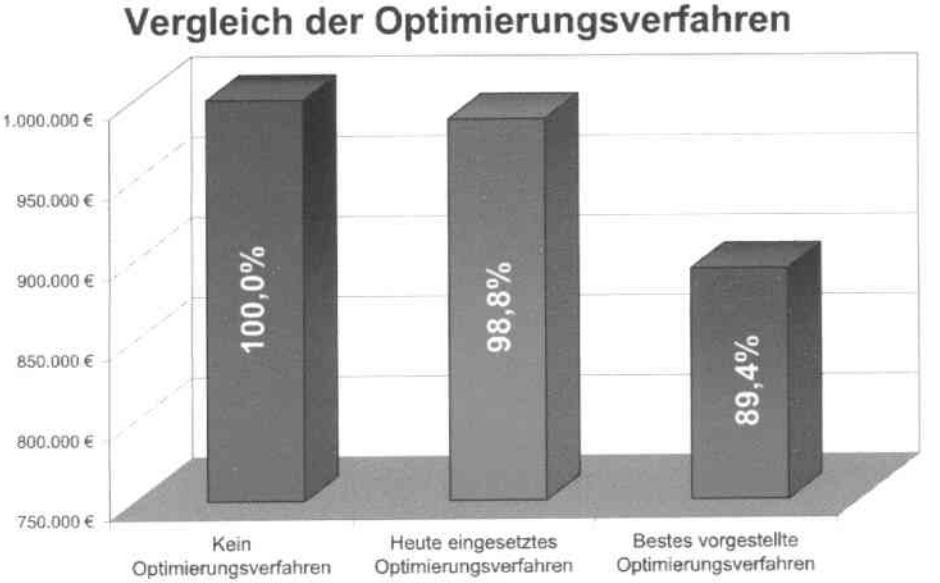


Abbildung 7.1: Vergleich der Optimierungsverfahren: Übersicht der Gesamttestkosten nach Einsatz der Optimierungsverfahren

Die, diesem und dem folgenden Diagramm 7.2 zu Grunde liegende Annahmen, gehen von einer Monatsproduktion mittelgroßer Chiphersteller von 100 Mio. Chips aus. Weiterhin wurde die Anzahl der auf einem Bauteil durchgeführten Tests auf maximal 100 begrenzt. Die Anzahl der Tests, die als mögliche Kandidaten für die Einsparung betrachtet werden, wurde auf

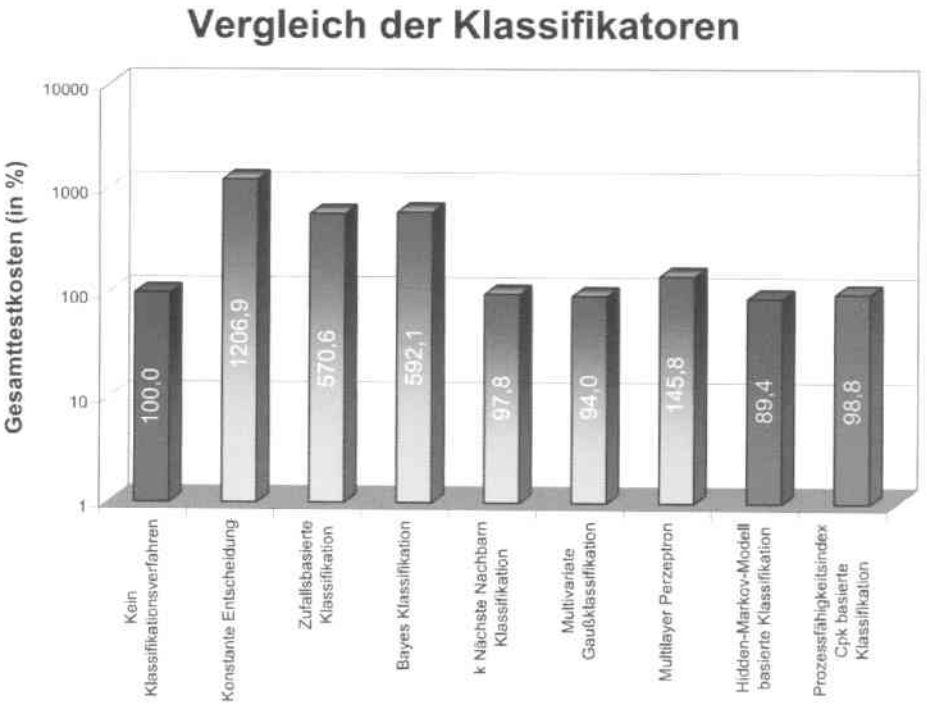


Abbildung 7.2: Vergleich der Klassifikationsverfahren: Überblick der eingesparten Testkosten in Prozent

60 festgesetzt. Die Kosten einer Testausführung wurden gemäß der im Abschnitt 3.1.1 gemachten und auf tatsächliche Vorgaben gestützten Annahmen auf 0.0001 EUR und die Kosten eines übersehenen, schlechten Bauteils auf 0.01 EUR festgelegt.

Diese Arbeit untersuchte gleich mehrere für die Optimierungsverfahren einsetzbaren Klassifikations- sowie Datenoptimierungsverfahren, die im Kapitel 4 präsentiert wurden und aus der Spracherkennung bekannt sind. Es konnte einerseits festgestellt werden, dass diese Verfahren in passenden Kombinationen gute oder sogar sehr gute Ergebnisse erzielen können. Andererseits stellte sich heraus, dass das von der Chipherstellungsindustrie zum Zeitpunkt der Entstehung dieser Arbeit erprobte Verfahren auf Basis des Prozessfähigkeitsindex C_{pk} auf keinem der verwendeten Datensätze überzeugen konnte.

Eine Übersicht der verwendeten Klassifikationsverfahren, sowie der mit diesen Verfahren erzielten Einsparungen sind im Diagramm 7.2 prozentuell angegeben. Zur besseren Übersicht, wurde die Skala des Diagramms logarithmisch gewählt.

Der in dem Diagramm dargestellte Vergleich der Klassifikatoren bringt zwei Sachverhalte deutlich zum Ausdruck:

1. Einige Verfahren verursachen Zusatzkosten[1] statt der erhofften Einsparungen.

2. Das Optimierungsverfahren mit den höchsten Gewinnen basiert auf einem Hidden-Markov-Modell.

Generell stellt diese Arbeit eine ganze Palette von unterschiedlichen Algorithmen und Verfahren vor, die zur Optimierung der Testkosten und der Beseitigung eines wichtigen Flaschenhalses der Chipherstellungsindustrie beitragen können. Als eine Bestätigung der praktischen Relevanz, kann das an den Ergebnissen dieser Arbeit bekundete Interesse seitens der Chiphersteller gedeutet werden. Weitere Untersuchungen sind jedoch notwendig, um das volle, in den Messdaten vorhandene Einsparungspotenzial auszuschöpfen. Der Schlüssel dazu könnte in den bisher ungetesteten Kombinationen der vorgestellten Verfahren liegen.

[1] Die Zusatzkosten entstehen durch eine zu hohe Quote der übersehenen, schlechten Bauteile an den eingesparten Testausführungen

Abbildungsverzeichnis

Literaturverzeichnis

[BD88] J.B. Brockman and S.W. Director. Predictive subset testing for ic performance. In *ICCAD-88: Digest of Technical Papers*, pages 336–339. IEEE Press, November 1988.

[Cha01] M. Jeya Chandra. *Statistical Quality Control*. CRC Press LLC, 2001.

[DHS00a] Richard Duda, Peter Hart, and David Stork. *Pattern Classification (2nd Edition)*, chapter Chapter 10.4.3, K-means clustering. Wiley-Interscience, October 2000.

[DHS00b] Richard Duda, Peter Hart, and David Stork. *Pattern Classification (2nd Edition)*, chapter Chapter 4.10, Fisher Linear Discriminant. Wiley-Interscience, October 2000.

[DHS00c] Richard Duda, Peter Hart, and David Stork. *Pattern Classification (2nd Edition)*, chapter Chapter 10.14.1, Self-organizing feature maps. Wiley-Interscience, October 2000.

[DHS00d] Richard Duda, Peter Hart, and David Stork. *Pattern Classification (2nd Edition)*, chapter Chapter 2, Bayesian decision theory. Wiley-Interscience, October 2000.

[DHS00e] Richard Duda, Peter Hart, and David Stork. *Pattern Classification (2nd Edition)*, chapter Chapter 6, Multilayer Neural Networks. Wiley-Interscience, October 2000.

[DHS00f] Richard Duda, Peter Hart, and David Stork. *Pattern Classification (2nd Edition)*, chapter Chapter 3.10, Hidden Markov Models. Wiley-Interscience, October 2000.

[Fei05] Maxim Feinleb. Einsatz von Mustererkennungsverfahren zur Optimierung von Chip-Testplänen, July 2005.

[FLJ91] P.E. Fieler and N. Loverro Jr. Defects tail off with six-sigma manufacturing. *IEEE Circuits and Devices Magazine*, 7(5):18–20, September 1991.

[Fuk90] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition (2nd Edition)*, chapter Chapter 7, Nonparametric classification and error estimation. Academic Press, Inc., September 1990.

[ITR03] ITRS. The itrs international technology roadmap for semiconductors. online: http://public.itrs.net/, 2003.

[Koh90] T. Kohonen. Improved versions of learning vector quantization. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 545–550, San Diego, CA, USA, June 1990.

[LWMa99] Jungran Lee, D.M.H. Walker, L. Milor, and alteri. IC performance prediction for test cost reduction. In *Semiconductor Manufacturing Conference Proceedings*, pages 111–114. IEEE Press, 1999.

[Mal86] Wojciech Maly. Optimal order of the VLSI IC testing sequence. In *DAC '86: Proceedings of the 23rd ACM/IEEE conference on Design automation*, pages 560–566, Piscataway, NJ, USA, 1986. IEEE Press.

[MB89] E.J. McCluskey and F. Buelow. IC quality and test transparency. *IEEE Transactions on Industrial Electronics*, 36(2):197–202, May 1989.

[Mil98] Linda Milor. A tutorial introduction to research on analog and mixed-signal circuit testing. *IEEE Transactions on Circuits and Systems-II*, 45(10):1389–1407, October 1998.

[MSV90] L. Milor and A. Sangiovanni-Vincentelli. Optimal test set design for analog circuits. In *Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*, pages 294–297, Santa Clara, CA, USA, November 1990.

[MSV94] L. Milor and A.L. Sangiovanni-Vincentelli. Minimizing production test time to detect faults in analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(6):796–813, June 1994.

[PR02] I. Pomeranz and S.M. Reddy. Improving the efficiency of static compaction based on chronological order enumeration of test sequences [logic testing]. *2002. (ATS '02). Proceedings of the 11th Asian Test Symposium*, pages 61–66, November 2002.

[Rij79] C. J. Van Rijsbergen. *Information Retrieval*, chapter Chapter 7, Evaluation. Butterworth-Heinemann, Newton, MA, USA, 1979.

[RP99] E.M. Rudnick and J.H. Patel. Efficient techniques for dynamic test sequence compaction. *IEEE Transactions on Computers*, 48(3):323–330, March 1999.

[RvSK05] Ivica Rogina, Hans-Martin von Staudt, and Gunther Karner. Controlling test plans by information-content-based redundancy analysis. In *11th Annual International Mixed-Signals Testing Workshop, IMSTW'05*, June 2005.

[SKCa99] Sanjay Sengupta, Sandip Kundu, Sreejit Chakravarty, and alteri. Defect-based test: A key enabler for successful migration to structural test, intel technology journal q1/99. Technical report, Intel Corp., 1999.

[SS91a] T.M. Souders and G.N. Stenbakken. Cutting the high cost of testing. *IEEE Spectrum*, 28(3):48–51, March 1991.

[SS91b] G.N. Stenbakken and T.M. Souders. Linear error modeling of analog and mixed-signal devices. In *Test Conference, 1991, Proceedings., International*, October 1991.

[Swi04] Cost saving opportunities in the semiconductor test process. Technical report, Pintail Technologies, Inc., 2004.