Technische Universität Dortmund
Fakultät Statistik

Optimierung der Toneinsatzzeiterkennung

Dissertation
zur Erlangung des akademischen Grades
Doktorin der Naturwissenschaften

von

M.Sc.
NADJA BAUER

Vorgelegt: Dortmund, 18. Januar 2016
Tag der mündlichen Prüfung: 4. Februar 2016

Erstgutachter: Prof. Dr. Claus Weihs
Zweitgutachter: Dr. Uwe Ligges
Kommissionsvorsitz: Prof. Dr. Jörg Rahnenführer

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Dissertation ist bisher keiner anderen Fakultät vorgelegt worden. Ich erkläre, dass ich bisher kein Promotionsverfahren erfolglos beendet habe und dass keine Aberkennung eines bereits erworbenen Doktorgrades vorliegt.

Nadja Bauer

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit ist im Laufe der Forschung im Teilprojekt C2 (Versuchsplanung für dynamische Prozesse) des Sonderforschungsbereichs (SFB) 823 (Statistik nichtlinearer dynamischer Prozesse)[1] entstanden. Der Forschungsstandort – Lehrstuhl Computergestützte Statistik (Fakultät Statistik) an der Technischen Universität Dortmund unter Leitung von Prof. Dr. C. Weihs – zeichnet sich insbesondere durch mehrjährige Erfahrungen und breit gefächerte Expertise im Bereich von Optimierung industrieller und computergestützter Anwendungen sowie durch Entwicklung innovativer Optimierungsmethoden aus.

Das Teilprojekt C2 befasst sich mit Konstruktion neuer optimaler Versuchspläne zur Optimierung komplexer dynamischer Prozesse. Ziel der Optimierung ist es, mit möglichst wenigen Versuchen eine Einstellung der Einflussparameter zu finden, die den besten Wert der Zielgröße liefert. In der klassischen Terminologie wird ein Versuchsplan als optimal bezeichnet, wenn die zugehörige Planmatrix bestimmte Eigenschaften (Optimalitätskriterien) erfüllt. Für die Konstruktion solcher Planmatrizen ist neben einiger weiterer Annahmen häufig auch die Kenntnis über den zugrunde liegenden Zusammenhang zwischen den Einflussparametern und der Zielgröße notwendig. Dabei ist der Großteil der Theorie für lineare Zusammenhänge mit unkorrelierten Beobachtungen entwickelt worden (Pukelsheim, 2006). Dette et al. (2015) erarbeiten aktuell – ebenso in Rahmen des C2 Teilprojekts – neue Theorien für optimale Versuchspläne mit korrelierten Beobachtungen.

Bei der Optimierung industrieller bzw. computergestützter Anwendungen liegen jedoch oft sehr komplizierte Probleme vor, die durch eine Reihe von speziellen Nebenbedingungen und eine große Anzahl an numerischen (meinst nicht normalverteilten) und diskreten Einflussgrößen charakterisiert sind. Weiterhin ist meistens der Zusammenhang zwischen den Einfluss- und Zielgrößen unbekannt. Dieses wird üblicherweise als „Black-Box“ bezeichnet. Bei solchen Problemen stößt der theoretische Ansatz an seine Grenzen. Weit verbreitet in diesem Bereich ist daher die Anwendung effizienter modellbasierter Techniken für globale (sequentielle) Optimierung (Jones et al., 1998). In Kapitel 6 dieser Arbeit wird ausführlich auf die klassische und sequentielle Versuchsplanung bzw. Optimierung eingegangen.

[1] Der SFB 823 wird von der Deutschen Forschungsgemeinschaft (DFG) gefördert.

In Rahmen der Kooperation mit dem Teilprojekt B3 (statistische Modellierung zeitlich und spektral hoch aufgelöster Audiodaten in Hörgeräten, SFB 823) entstand das Thema dieser Arbeit. Denn die Einsatzzeiterkennung der musikalischen Töne ist ein maßgebender Bestandteil bei der Optimierung von Hörgeräten. Das akustische Signal ist dabei ein dynamischer Prozess, deren spontane Änderungen in der Regel den neuen Toneinsätzen entsprechen.

In den letzten dreißig Jahren wurden zahlreiche Algorithmen für die Lösung des Einsatzzeiterkennungsproblems vorgeschlagen. Einige basieren dabei auf Verteilungsannahmen über das akustische Signal und verwenden statistische Testmethoden, um auf Verteilungsänderung zu bestimmten Zeitpunkten zu testen. Andere fassen plötzliche Änderungen in der zeitlichen oder spektralen Struktur des Signals als Indikator für einen neuen Toneinsatz auf. Außerdem wurden zahlreiche Methoden zur Vor- und Nachverarbeitung des Signals vorgeschlagen wie Signalzerlegung in mehrere Frequenzbänder oder Glättung des extrahierten Merkmalsvektors.

Die meisten Algorithmen vereint allerdings die Tatsache, dass sie von vielen einstellbaren Parametern abhängen. Bisher wurde in der „State of the Art“ Literatur ihre Optimierung nur sehr grob behandelt: die meisten Parameter werden auf Erfahrungswerte eingestellt und die wenigen verbliebenen (meistens numerischen) Parameter werden entweder mit Gittersuche oder mit Hilfe einfacher Optimierungsstrategien getuned. Häufig wird durch so eine Optimierung eine Überanpassung des Verfahrens auf den verwendeten Datensatz herbeigeführt. Es gab bisher keine Arbeit, in der alle „Bauteile“ der Einsatzzeiterkennung simultan optimiert werden. Dies kann an der Komplexität des Problems liegen, denn die gleichzeitige Optimierung einer großen Anzahl an stetigen und diskreten Parametern bedarf der Anwendung von angemessenen Optimierungsstrategien.

Im Weiteren wird ein Überblick über die Struktur dieser Arbeit gegeben. Kapitel 2 fasst die grundlegenden Begriffe der Signalanalyse zusammen. Ausgehend von den periodischen und nichtperiodischen Signalen wird die stetige und diskrete Fourier Transformation eingeführt sowie die Konstruktion eines Spektrogramms erläutert. In dem anschließenden Kapitel 3 werden verschiedene Definitionen eines Toneinsatzes besprochen und die gängigen Verfahren der Einsatzzeiterkennung ausführlich vorgestellt. Diese dienen als Grundlage für die Konstruktion eines umfassenden Algorithmus mit zahlreichen zu optimierenden Parametern. Außerdem wird die Bewertung der Einsatzzeiterkennung diskutiert. Der F-Wert ist dabei ein etabliertes Gütemaß, welches jedoch eine Reihe von Nachteilen aufweist. Aus diesem Grund wird ein alternatives Gütemaß vorgeschlagen und diskutiert.

Die in Kapitel 3 erfolgte Einarbeitung in die klassischen Algorithmen zur Einsatzzeiterkennung legt nahe, dass sich auch Klassifikationsverfahren für diese Aufgabe sehr gut eignen können. Angesichts der mangelnden Behandlung dieses Ansatzes in der Literatur wird in Kapitel 4 eine eigene Idee für die multivariate Einsatzzeiterkennung vorgeschlagen. Auch dieses Verfahren hängt von einer Menge von einstellbaren Parametern ab, die im Weiteren optimiert werden.

Da das Ergebnis der Optimierung oft von den zugrunde liegenden Daten abhängt, widmet sich Kapitel 5 der Problematik der Datenbank-Konstruktion. Es werden solche Musikstücke benötigt,

für die die Information über die wahren Einsatzzeiten bekannt ist. Zum einen können die echten Audioaufnahmen von den Experten „per Hand“ annotiert werden. Zum anderen aber können die Musikstücke nach einem festgelegten Schema mittels spezieller Software künstlich erzeugt werden. In der aktuellen Forschungsliteratur werden überwiegend kleine manuell annotierte Datensätze für den Vergleich verschiedener Algorithmen betrachtet, während die zweite Möglichkeit der Datensatz-Erzeugung kaum berücksichtigt wird. Eines der Ziele dieser Arbeit ist es daher, Optimierungsergebnisse in Abhängigkeit von dem Typ der verwendeten Datenbank zu vergleichen.

In Kapitel 6 wird die verwendete modellbasierte Optimierungsmethode ausführlich erklärt. Es handelt sich dabei um ein iteratives Verfahren: neue Parameterkombinationen werden aufgrund der Information aus den Auswertungen von vorherigen Parametereinstellungen mit Hilfe eines Modells vorgeschlagen. Das Modell wird nach jeder zusätzlichen Iteration neu angepasst. Außerdem wird ein neues Verfahren – instanzgebundene modellbasierte Optimierung – vorgeschlagen. Dieses ist durch das Anwendungsproblem motiviert und behandelt die Optimierungsdatenbank nicht als eine Einheit, sondern als ein Set von Probleminstanzen. In jeder Iteration wird eine neue Parametereinstellung aufgrund ihrer Auswertung auf einer kleinen „repräsentativen“ Menge von Probleminstanzen als aussichtsreich bzw. nicht aussichtsreich bewertet. Lediglich die aussichtsreichen Einstellungen werden anschließend auf allen Probleminstanzen evaluiert. Dadurch sollen unnötige Auswertungen vermieden und die Optimierungszeit verkürzt werden. Interessant ist, ob das vorgeschlagene Verfahren trotz weniger Auswertungen mit der gewöhnlichen modellbasierten Optimierung konkurrieren kann (in Bezug auf das Finden des Optimums). Weiterhin stellt dieses Kapitel das Schema für die Validierung der Optimierungsverfahren und die verwendete statistische Testmethodik vor.

Kapitel 7 umfasst die Forschungsfragen und das Vorgehen bei der Planung der Computerexperimente. Hinsichtlich der Auswertung der Experimente wird auf die Problematik des multiplen Testens hingewiesen und eine Lösung für ihre Behandlung vorgeschlagen. Die Ergebnisse der Optimierung werden anschließend in Kapitel 8 ausgewertet sowie die Forschungsfragen beantwortet.

Ein Ausblick über weitere interessante Fragestellungen bzw. viel versprechende Ideen für die Verbesserung der Einsatzzeiterkennung wird in Kapitel 9 gegeben. Abschließend fasst Kapitel 10 die relevanten Erkenntnisse dieser Arbeit zusammen.

Kapitel 2

Grundlagen der Signalanalyse

Digitale Signalverarbeitung (engl. *digital signal processing*) ist ein Oberbegriff für verschiedene Anwendungen wie Bild-, Video-, Musik- oder Sprachverarbeitung, bei denen Signale in digitaler (elektronischer) Form aufbereitet werden. Diese Arbeit befasst sich mit der Musikverarbeitung. Das Wort Musik eindeutig und umfassend zu definieren würde den Rahmen dieser Arbeit sprengen. Allerdings liefert der Duden eine für die weitere Analyse ausreichende Definition: Musik ist die „Kunst, Töne in bestimmter (geschichtlich bedingter) Gesetzmäßigkeit hinsichtlich Rhythmus, Melodie, Harmonie zu einer Gruppe von Klängen und zu einer stilistisch eigenständigen Komposition zu ordnen“[1].

Die Grundsteine der Musik sind also die Musiktöne, welche unter anderem durch folgende Charakteristiken beschrieben werden können: Tonhöhe, Ton erzeugendes Instrument, Länge und Lautstärke. Es existieren verschiedene Instrumentenarten wie Schlaginstrumente (Trommel oder Pauke), Streichinstrumente (Geige oder Bratsche), Saiteninstrumente (Gitarre oder Klavier) und Blasinstrumente (Flöte oder Trompete). Bei monophonen Signalen darf ein neuer Ton erst dann anfangen, wenn der vorherige Ton aufgehört hat. Bei polyphonen Signalen, dagegen, dürfen mehrere Töne auch gleichzeitig gespielt werden, so dass die sogenannten *Akkorde* entstehen.

Dieses Kapitel zielt darauf hin, eine kurze Einführung in die digitale Musikverarbeitung zu geben. Als erstes wird das akustische Signal definiert und klassifiziert. Anschließend wird die mathematische Herleitung eines der wichtigsten Instrumente der Signalverarbeitung, nämlich der diskreten Fourier Transformation, in aufeinander aufbauenden Schritten vorgestellt. Außerdem werden instrumentbedingte Unterschiede bestimmter Toncharakteristiken illustriert, um die Probleme der Toneinsatzzeiterkennung anzudeuten.

[1] `http://www.duden.de/rechtschreibung/Musik`, Stand: 01.05.2015.

2.1 Akustisches Signal

Hammond und White (2008) (S. 5) definieren das *akustische Signal* als Ausgabe eines Druckmessumformers – eines elektronischen Geräts, welches die physikalische Größe *Schalldruck* in ein elektrisches Signal umwandelt. Dabei stellt das Signal eine zeitliche Entwicklung dar, die von der räumlichen Position dieses Geräts abhängt.

Man unterscheidet zwischen analogen und diskreten Signalen. Das *analoge Signal* $x(t)$ ist eine Funktion der stetigen (Zeit-)Variable t. In der Praxis ist es aber nicht möglich, für ein beliebiges Signal die zugehörige Funktion zu schätzen. Das Signal wird mit einer konstanten *Abtastperiode* Δt abgetastet, wobei $t_k = k\Delta t$ $(k = 0, 1, 2, \ldots)$ gilt. Auf diesem Weg erhalten wir ein *zeit-diskretes Signal*: $x[k] = x(t_k)$. Im Weiteren werden in dieser Arbeit – in Anlehnung an Huang et al. (2001) – runde Klammern für analoge und eckige Klammern für diskrete Signale verwendet.

Die *Abtastfrequenz* bzw. *Abtastrate* (engl. *sampling frequency* bzw. *sampling rate*) F_s ist definiert als Inverse der Abtastperiode: $F_s = 1/\Delta t$ und wird in Hz (Hertz) gemessen (Huang et al., 2001, S. 202). Abbildung 2.1 zeigt ein analoges Signal und dessen zeit-diskrete Darstellung mit Abtastperiode 0.25 ms (Millisekunden) bzw. $F_s = 4$ kHz.

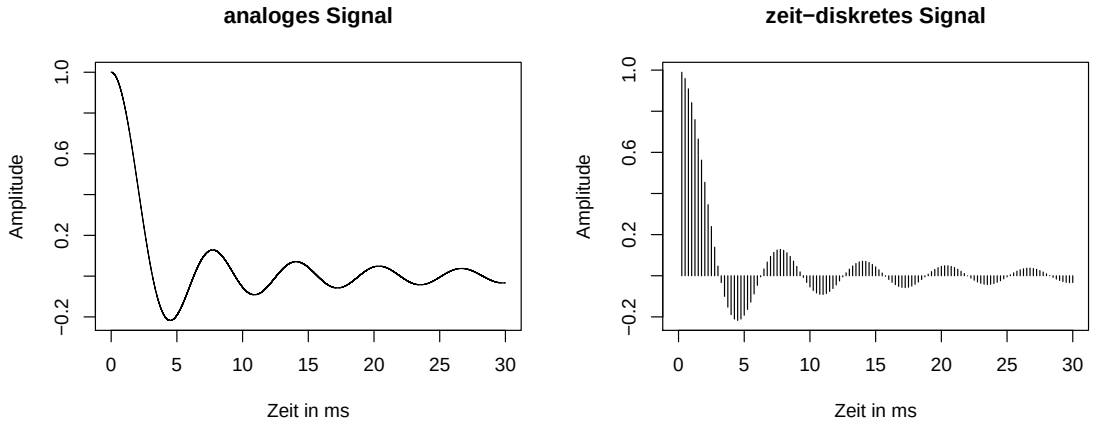

Abbildung 2.1: *Beispiel eines analogen Signals (links) und seiner diskreten Abtastung (rechts) mit Abtastrate 4 kHz.*

Ein sehr weit verbreitetes Format für die digitale Speicherung der Musik ist das *WAV*-Format[2]. Die CD-Qualität einer Audioaufnahme hat folgende Parameter: 16-Bit Auflösung (d.h. die Amplitude des Signals kann zwischen $-32\,767$ und $32\,766$ variieren) mit einer Abtastrate von 44.1 kHz (Ligges, 2006, S. 6). Dieses Format wird in dieser Arbeit vorausgesetzt.

[2] http://en.wikipedia.org/wiki/WAV, Stand: 01.05.2015.

Es wird zwischen mehreren Signalarten unterschieden. Eine umfassende Signalklassifikation wird bei Hammond und White (2008) (S. 4) veranschaulicht und diskutiert (vgl. Abbildung 2.2). In dieser Arbeit wird auf *periodische*, *fast periodische* und *transiente Signale* eingegangen.

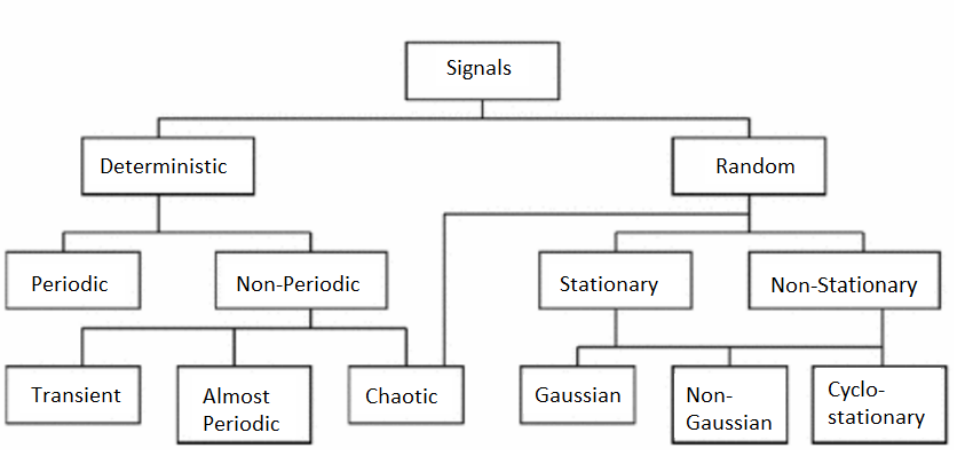

Abbildung 2.2: *Signalklassifikation, entnommen aus Hammond und White (2008) (S. 4).*

2.2 Periodische Signale

Ein Signal heißt *periodisch*, falls eine Periode T existiert mit $x(t) = x(t+T)$. Eine der wichtigsten Klassen von periodischen Signalen sind die sinusförmigen Wellen

$$x(t) = A \cdot \sin(\omega t + \phi) \tag{2.1}$$

mit der Kreisfrequenz $\omega = 2\pi f$. A und f entsprechen der Amplitude und der Frequenz des Signals und ϕ ist die Phasenverschiebung. Im Folgenden wird allerdings $\phi = 0$ verwendet.

Der französische Mathematiker Joseph Fourier stellte Anfang des 19. Jahrhunderts die Behauptung auf, dass sich alle periodischen und stetig differenzierbaren Funktionen als Summe von Sinus- und Kosinus-Termen darstellen lassen. Die Reihenwicklung für $x(t)$ kann wie folgt aufgeschrieben werden (Butz, 2009, S. 5 ff):

$$x(t) = \sum_{k=0}^{\infty} \left(A_k \cos(\omega_k t) + B_k \sin(\omega_k t)\right) \text{ mit } \omega_k = \frac{2\pi k}{T}. \tag{2.2}$$

A_k und B_k sind die *Fourierkoeffizienten* (im Folgenden oft mit FKen abgekürzt). Wichtig zu bemerken ist, dass sich jede zu approximierende Funktion $x(t)$ als Kombination der geraden ($x(-t) = x(t)$) und der ungeraden ($x(-t) = -x(t)$) Abschnitte darstellen lässt. Ist die Funktion gerade, sind alle FKen B_k gleich Null. Bei ungeraden Funktionen sind dagegen alle A_k gleich Null. Eine wichtige Annahme hier ist, dass $B_0 = 0$ ist.

Die Fourierkoeffizienten A_k and B_k sollen so bestimmt werden, dass die Gleichung (2.2) erfüllt ist. Als Vorüberlegung soll daran erinnert werden, dass das Integral von einer Sinus- bzw. Kosinusfunktion über eine volle Periode T gleich 0 ist. Lediglich im Fall von $\int_{-T/2}^{T/2}\cos(0)d(t)$ ist dieses Integral gleich T. Außerdem sind die Integrale von $\sin(\omega_{k1}t)\cdot\cos(\omega_{k2}t)$, $\sin(\omega_{k1}t)\cdot\sin(\omega_{k2}t)$ und $\cos(\omega_{k1}t)\cdot\cos(\omega_{k2}t)$ über eine volle Periode T für $k1 \neq k2$ gleich 0, sonst $T/2$ oder T (je nach Fall). Nun multipliziert man die rechte und die linke Seite von (2.2) mit $\cos(\omega_k t)$ bzw. $\sin(\omega_k t)$ und integriert von $-T/2$ bis $T/2$. Somit erhält man die Formeln für A_k bzw. B_k:

$$A_0 = \frac{1}{T}\int_{-T/2}^{T/2} x(t)d(t), \tag{2.3}$$

$$A_k = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\cos(\omega_k t)d(t), \text{ für } k \neq 0, \tag{2.4}$$

$$B_k = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\sin(\omega_k t)d(t), \text{ für alle } k. \tag{2.5}$$

Butz (2009) (S. 8) interpretiert die obigen Gleichungen wie folgt: „durch die Wichtung der Funktion $x(t)$ mit $\cos(\omega_k t)$ bzw. $\sin(\omega_k t)$ ‚pickt' man sich bei der Integration die spektralen Komponenten auf $x(t)$ heraus, die den geraden bzw. ungeraden Anteilen mit der Frequenz ω_k entsprechen."

Gebräuchlich ist die Formel der Reihenentwicklung in der komplexen Schreibweise. Dafür wird die Eulersche Identität verwendet:

$$e^{i\omega_k t} = \cos(\omega_k t) + i\sin(\omega_k t) \text{ mit } i^2 = -1. \tag{2.6}$$

Die Formel (2.2) lässt sich mit Hilfe von (2.6) wie folgt aufschreiben:

$$x(t) = A_0 + \sum_{k=1}^{\infty}\left(\frac{A_k - iB_k}{2}e^{i\omega_k t} + \frac{A_k + iB_k}{2}e^{-i\omega_k t}\right). \tag{2.7}$$

Weiterhin werden die Abkürzungen

$$C_0 = A_0, C_k = \frac{A_k - iB_k}{2}, \text{ und } C_{-k} = \frac{A_k + iB_k}{2} \tag{2.8}$$

benötigt, um die endgültige Formel in der komplexen Schreibweise aufzuschreiben, wobei k hier nicht mehr von 0 sondern von $-\infty$ läuft:

$$x(t) = \sum_{k=-\infty}^{\infty} C_k e^{i\omega_k t}. \tag{2.9}$$

Die komplexen Fourierkoeffizienten C_k lassen sich auch einfacher aufschreiben, indem man die

linke und die rechte Seite von (2.9) mit $e^{-i\omega_k t}$ multipliziert und über eine volle Periode integriert:

$$C_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t)e^{-i\omega_k t}dt, \text{ für alle } k. \tag{2.10}$$

Musikalische Töne werden mittels periodischer Sinuswellen modelliert (Osgood, 2007, S. 47). Reine Töne entsprechen dabei genau der Gleichung (2.1) und hören sich wie durch die Betätigung der Tasten eines digitalen Telefons erzeugte Signale an. f heißt die *fundamentale Frequenz* oder die *Grundfrequenz* eines Tones (in der Fachliteratur oft als f_0 bzw. F_0 bezeichnet). Musikinstrumente erzeugen neben der Grundfrequenz auch die Harmonischen (Mehrfachen der Grundfrequenz, auch *Obertöne* genannt), so dass nicht mehr von einem Ton, sondern von einem *Klang* die Rede ist. Es sind die verschiedenen Obertonstrukturen, die den Klang verschiedener Instrumente auszeichnen. Zu relativ obertonarmen Instrumenten zählt z.B. die Flöte. Um noch ein Beispiel zu nennen, haben die von der Klarinette erzeugten Klänge die Besonderheit, dass die ungeraden Obertöne besonders stark ausgeprägt sind. Im Weiteren wird in Bezug auf Musikinstrumente lediglich der Begriff *Ton* statt *Klang* verwendet, um die zugrunde liegende Grundfrequenz zu akzentuieren.

2.3 Nicht-periodische Signale

In der Natur kommen *nicht-periodische* Signale sehr häufig vor. Eine Klasse solcher Signale sind die *transienten* Signale: nicht-periodische Signale, die im Wesentlichen lokalisiert sind, d.h. einen Anfang und ein Ende haben (Hammond und White, 2008, S. 7). Bello et al. (2005) definieren das transiente Signal als ein kurzes Intervall, während dessen sich das Signal schnell und in einer nicht-trivialen, relativ nicht vorhersagbaren Art und Weise entwickelt. Transiente Signale spielen bei der Definition des Toneinsatzes eine wichtige Rolle (s. Abschnitt 3.1).

Fast-periodische (engl. *almost periodic*) Signale bestehen aus einer Summe mehrerer an sich periodischer Komponenten, wobei das Verhältnis ihrer Frequenzen keine rationale Zahl ergibt. Ein mathematisches Beispiel dafür ist folgendes Signal:

$$x(t) = \sin(2\pi t) + \sin(2\pi\sqrt{2}t). \tag{2.11}$$

Hammond und White (2008) (S. 7) zeigen, dass einzelne Abschnitte der durch Sustain-Pedal verlängerten Klaviertöne die Eigenschaften fast-periodischer Signale aufweisen.

Es stellt sich die Frage, ob die beiden Arten nicht-periodischer Signale auch eine Fourier-Darstellung besitzen. Die Grundidee dabei ist, das nicht-periodische Signal als ein Signal mit unendlicher Periode aufzufassen, also mit $T \to \infty$.

2.4 Stetige und diskrete Fourier Transformation

Für die *stetige Fourier Transformation* (engl. *Continuous Fourier Transform*, CFT) ist der Übergang von Reihen- zu Integraldarstellung für $T \to \infty$ von Bedeutung. Wir betrachten hier nicht mehr $\omega_k = 2\pi k/T$, sondern die kontinuierliche Kreisfrequenz ω. Weiterhin gilt unter Betrachtung von (2.10) (Butz, 2009, S. 34):

$$\lim_{T\to\infty}(TC_k) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt. \tag{2.12}$$

Es lässt sich beweisen (Butz, 2009, S. 35), dass die folgenden Formel für die Hin- bzw. Rück-Fouriertransformation gelten:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi ift}dt, \tag{2.13}$$

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{2\pi ift}df, \tag{2.14}$$

wobei $X(f)$ die Fouriertransformierte von $x(t)$ ist.

In Bezug auf Index k wird im Weiteren anstatt $k = 0,1,\ldots,N-1$ (wie in Abschnitt 2.1 eingeführt) $k = 1,\ldots,N$ verwendet. Uns interessiert nun die Fouriertransformation der Folge der Abtastwerte $x[k]$. Für $T = N \cdot \Delta t$ gilt:

$$e^{2\pi ift} \to e^{2\pi ik/N}. \tag{2.15}$$

Der nachfolgend aufgeführte Übergang von der stetigen zu der *diskreten Fourier Transformation* (DFT) ist nur für T-periodische Signale $x(t)$ möglich (Butz, 2009, S. 96):

$$X[j] = \frac{1}{N}\sum_{k=1}^{N} x[k]e^{-2\pi ijk/N}, \tag{2.16}$$

$$x[k] = \sum_{j=1}^{N} X[j]e^{2\pi ijk/N}. \tag{2.17}$$

Die maximale Frequenz, zu der die FKen bestimmt werden können, heißt die *Nyquist*-Frequenz und entspricht der Hälfte der Abtastrate:

$$F_{Nyq} = 0.5 \cdot F_s. \tag{2.18}$$

Die letzte Frequenzlinie entspricht somit der Frequenz F_{Nyq}.

In der Praxis ist allerdings die DFT für nicht-periodische Signale besonders relevant. Solche Signale entstehen z.B. durch Betrachtung einzelner Abschnitte von periodischen Funktionen, die nicht der ganzen Periode entsprechen. Dieses Problem lässt sich mit Hilfe von sogenanntem *Zero Padding* umgehen. Dabei wird das Signal zunächst mit einer bestimmten Anzahl von Nullen

aufgefüllt und anschließend mit seinem gespiegelten Bild fortgesetzt, so dass nun eine periodische Struktur entsteht. Eine genauere Erklärung dieses Vorgehens ist in Butz (2009) (S. 118 ff.) gegeben. Da das durch diese Transformation umgeformte Signal „künstlich“ periodisch ist, sind deren FKen um die Nyquist-Frequenz symmetrisch. Daher wird nur die erste Hälfte dieser Koeffizienten berücksichtigt. Zu N Signalabtastwerten werden also $N/2$ FKen berechnet, die den in gleichmäßigen Abständen liegenden Frequenzen im Bereich zwischen 0 und F_{Nyq} Hz entsprechen.

Die Berechnung der DFT benötigt N^2 Operationen. Cooley und Tukey (1965) haben einen Algorithmus vorgeschlagen, der für $N = 2^q$, $q = 1, 2, \ldots$, lediglich mit $N \log_2(N)$ Operationen auskommt. Dieser Algorithmus hat eine hohe Popularität auf dem Gebiet der Signalanalyse gewonnen und wird als *schnelle Fourier Transformation* (engl. *Fast Fourier Transformation*, FFT) bezeichnet. Die Grundidee besteht darin, die Zahlenfolge der Abtastwerte so lange in die geraden bzw. ungeraden Unterfolgen zu halbieren, bis nur ein einziger Wert pro Unterfolge bleibt. Da die DFT von einem Zahlenwert der Wert selbst ist, erhält man durch iterative Anwendung einfacher Regeln (Addition und Subtraktion) die gewünschten FKen (Butz, 2009, S. 125 ff).

2.5 Fensterung des Signals

Für viele Anwendungen ist die Veränderung bestimmter Signaleigenschaften über die Zeit von Interesse. Dabei wird das eingehende Signal bestehend aus L Abtastwerten in M kleine Abschnitte der Länge N aufgeteilt. In jedem Abschnitt können anschließend verschiedene Merkmale berechnet und miteinander verglichen werden.

Ein Schema der Signalfensterung ist in Abbildung 2.3 veranschaulicht: Im ersten Fenster werden die Abtastwerte 1 bis N des Signals berücksichtigt, wobei N ein wichtiger Parameter – die *Fensterlänge* (engl. *window size*) – ist. Ein weiterer wichtiger Parameter h – *Sprungweite* (engl. *hope size*) – bestimmt den Abstand, in welchem ein neues Fenster anfängt. Je kleiner h gewählt wird, desto mehr überlappende Fenster werden erzeugt. Falls $h = N$ ist, handelt es sich um disjunkte Fenster.

Nicht zu verwechseln ist die Sprungweite mit der *Überlappung* (engl. *overlap*), welche angibt, wie viele Abtastwerte von dem Fensterende aus zurückgelegt werden sollen, bis ein neues Fenster anfängt. In Abbildung 2.3 wäre die Überlappung gleich $N - h$. In der gängigen Literatur zur Einsatzzeiterkennung (und auch in dieser Arbeit) wird der Begriff der Sprungweite verwendet.

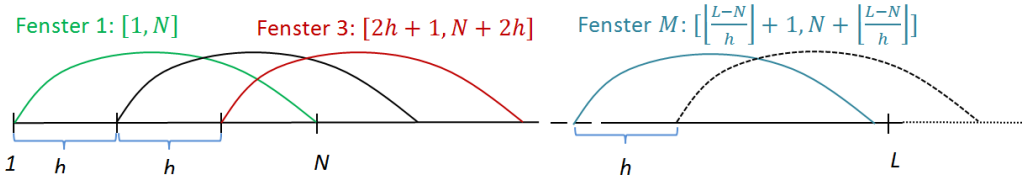

Abbildung 2.3: *Veranschaulichung der Signalfensterung.*

2.6 Spektrogramm als Zeit-Frequenz-Darstellung

Eine der populärsten Zeit-Frequenz-Darstellungen der Audiosignale ist das auf der *Kurzzeit-Fourier-Transformation* (engl. *Short-Time Fourier Transform*, STFT) basierende Spektrogramm. Im Grunde handelt es sich um die DFT, die auf mehrere kurze Signalabschnitte angewendet wird. Dabei wird das eingehende Signal gefenstert und mit einer Fensterfunktion $w(k)$, $k = 1,\ldots,N$, multipliziert, welche die Gewichtung der einzelnen Abtastwerte innerhalb des Signalabschnittes vorgibt. Eine umfassende Übersicht über Fensterfunktionen für DFT ist bei Harris (1978) gegeben. Es gibt verschiedene (zum Teil gegenläufige) Eigenschaften, anhand derer Fensterfunktionen verglichen werden (wie die relative Amplitude des Nebenmaximums, Breite des Hauptmaximums oder maximaler Abtastfehler), wobei es keine universelle Empfehlung für die „beste“ Wahl gibt. Aus diesem Grund werden in dieser Arbeit vier häufig verwendete Fensterfunktionen gegenübergestellt (Parameter *window.fun* für die Optimierung):

- *Rectangular* (Rechteck):
$$w_N^{Rect}(k) = 1, \tag{2.19}$$
- *Hanning*:
$$w_N^{Hann}(k) = 0.5\left(1-\cos\left(\frac{2\pi(k-1)}{N-1}\right)\right), \tag{2.20}$$
- *Blackman*:
$$w_N^{Black}(k) = 0.42-0.5\cos\left(\frac{2\pi(k-1)}{N-1}\right)+0.08\cos\left(\frac{4\pi(k-1)}{N-1}\right), \tag{2.21}$$
- *Gauss*:
$$w_N^{Gauss}(k) = \exp\left(-\frac{1}{2}\left(\frac{2k-(N+1)}{\sigma N}\right)^2\right), 0 \leq \sigma \leq 0.5. \tag{2.22}$$

Die vier oben genannten Fensterfunktionen sind in Abbildung 2.4 für $N = 2048$ dargestellt. Mit Ausnahme des Rechteckfensters sehen die Funktionen auf dem ersten Blick relativ ähnlich aus. Es wird allerdings im Weiteren veranschaulicht, dass die Fensterfunktion einen Einfluss auf die spektrale Darstellung des Signal zu haben scheinen.

Die Kurzzeit-Fourier-Transformation ist wie folgt definiert:

$$X[n,j] = \frac{1}{N}\sum_{k=1}^{N} x[h(n-1)+k]w_N(k)e^{-2\pi ijk/N}. \tag{2.23}$$

$X[n,j]$ stellt die FKen der j-ten Frequenzlinie des n-ten Fensters dar, $n = 1,\ldots,M$. In einem Spektrogramm wird der Absolutwert $|X[n,j]|$ jeder Frequenzlinie über die Zeit dargestellt mit $|z| = \sqrt{Re(z)^2 + Im(z)^2}$. Sehr häufig wird der logarithmierte Absolutwert abgebildet.

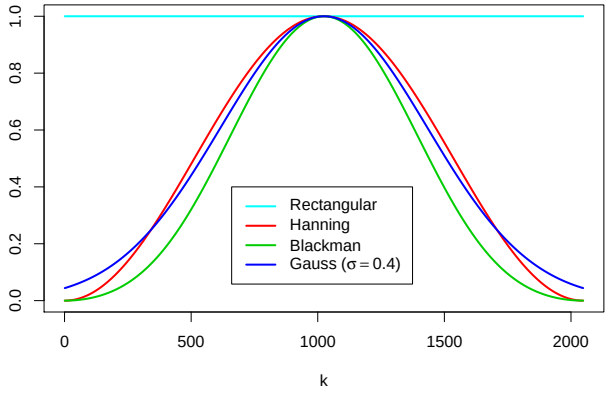

Abbildung 2.4: *Veranschaulichung der Fensterfunktionen für* $N = 2048$.

Beispiel 2.6.1 (Kurzzeit-Fourier-Transformation). *In Abbildung 2.5 ist der Amplitudenverlauf des Tons C4*[3]*, gespielt jeweils von Flöte (links) und Klavier (rechts), veranschaulicht. Der unterschiedliche Verlauf ist durch die Spielart zu erklären: während bei der Flöte für die Tonproduktion die Luft kontinuierlich hinein geblasen wird, wird bei Klavier die Saite einmalig angeschlagen, so dass der Ton langsam abklingt. Abbildungen 2.6 und 2.7 stellen Spektrogramme der beiden C4 Töne dar, die mithilfe der vier oben vorgestellten Fensterfunktionen gewonnen wurden (für* $N = h = 2048$*). Die Absolutwerte der FKen wurden hier zur besseren Darstellung logarithmiert und auf das Intervall* $[0,1]$ *umskaliert. Dies gilt auch für alle nachfolgenden Spektrogramme.*

Es fällt direkt auf, dass durch die Rechteck-Fensterfunktion ein sehr verschwommenes Bild entsteht, während Hanning- und Blackman-Fensterfunktionen auf dem ersten Blick sehr große Ähnlichkeiten haben.

Im Weiteren wird die Polarkoordinaten-Darstellung der komplexen Zahlen benötigt:

$$X[n,j] = |X[n,j]|e^{i\phi[n,j]}. \tag{2.24}$$

$|X[n,j]|$ ist – wie bereits oben erwähnt – der Absolutwert und $\phi[n,j]$ die Phase bzw. die Verschiebung des Signals. $\phi[n,j]$ ist ein auf das Intervall $[-\pi,\pi]$ skalierter Winkel. Der Wert der Phase lässt sich durch die sogenannte *atan2* Funktion[4] ausrechnen mit

$$\phi[n,j] = atan2\left(Re(X[n,j]), Im(X[n,j])\right), \text{ wobei} \tag{2.25}$$

[3] In dieser Arbeit wird die englische Tonnotation verwendet.

[4] `https://nf.nci.org.au/facilities/software/Matlab/techdoc/ref/atan2.html`, Stand: 01.05.2015.

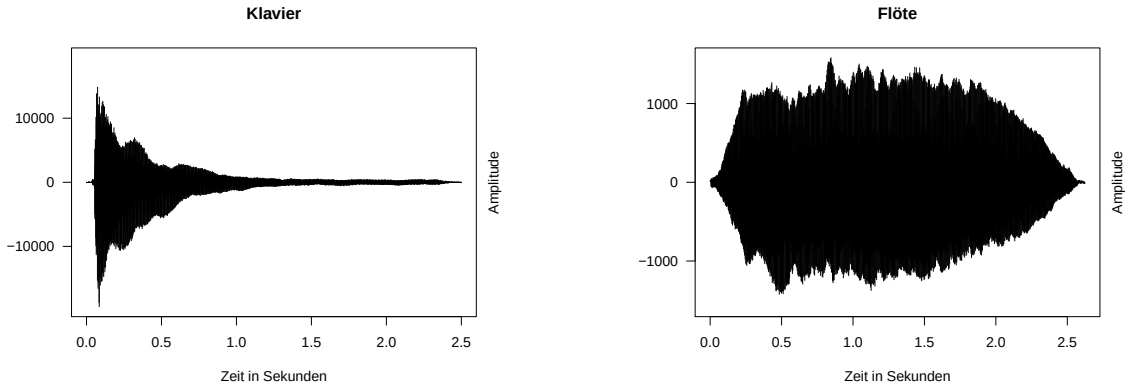

Abbildung 2.5: Amplitudenverlauf des Tons C4. Gespielt von Klavier (links) und Flöte (rechts).

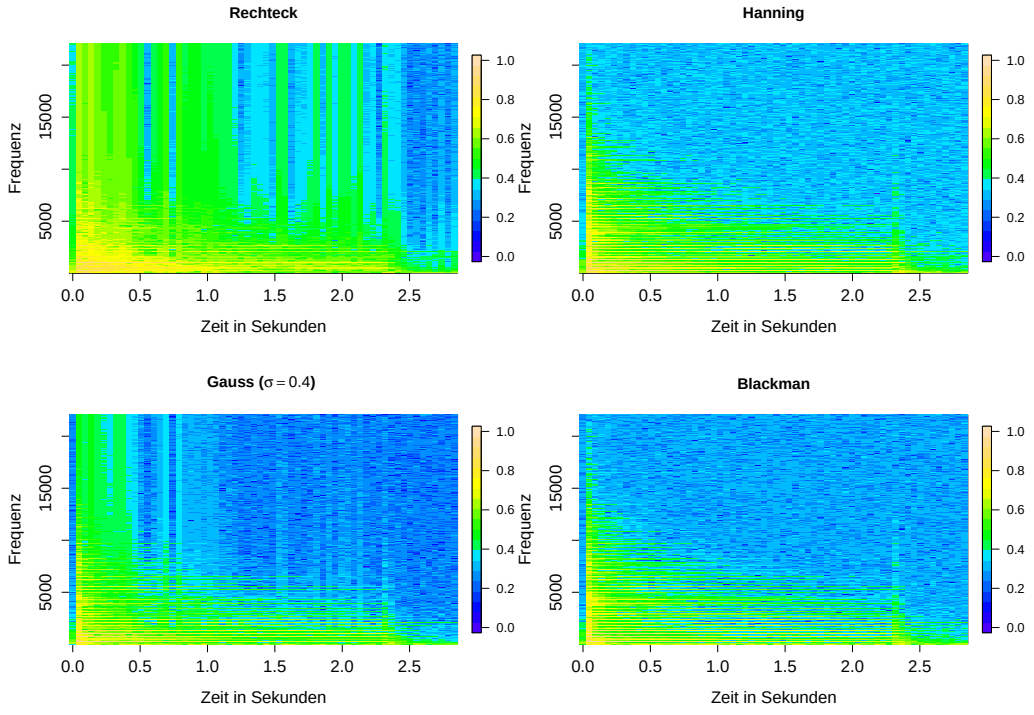

Abbildung 2.6: Spectrogramme vom Klavierton C4 mit verschiedenen Fensterfunktionen.

$$atan2(x,y) = \begin{cases} \arctan(y/x), & \text{falls } x > 0, \\ \arctan(y/x) + \pi, & \text{falls } x < 0, y \geq 0 \\ \arctan(y/x) - \pi, & \text{falls } x < 0, y < 0 \\ \pi/2, & \text{falls } x = 0, y > 0 \\ -\pi/2, & \text{falls } x = 0, y < 0 \\ 0, & \text{falls } x = 0, y = 0. \end{cases} \tag{2.26}$$

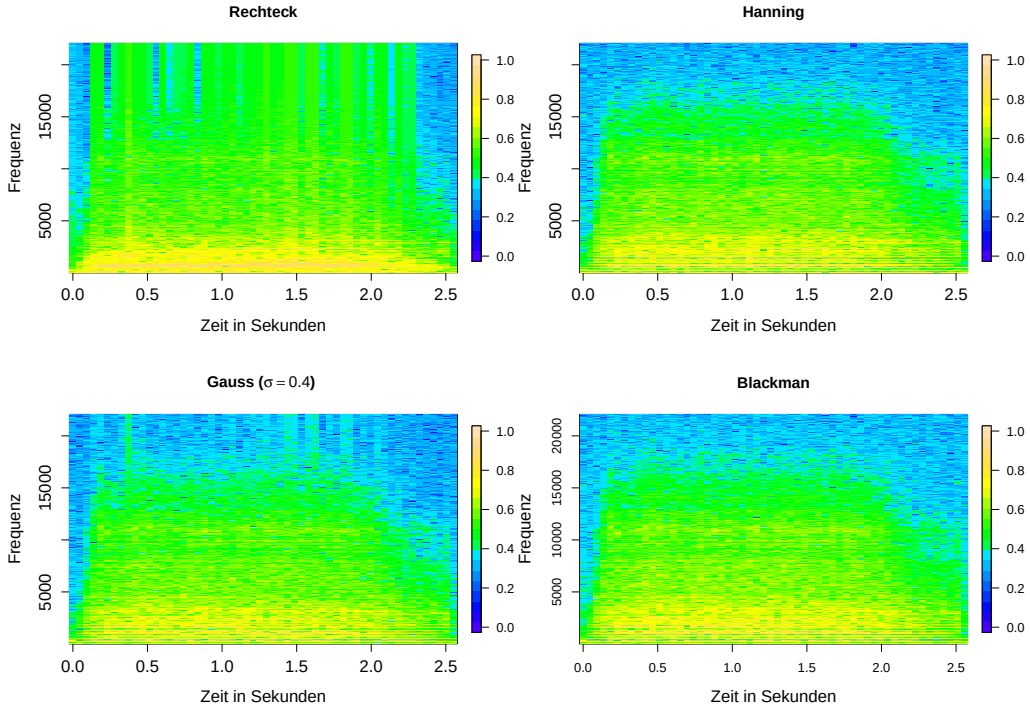

Abbildung 2.7: Spectrogramme vom Flötenton C4 mit verschiedenen Fensterfunktionen.

2.7 Tonhöhe

Klapuri (2006) (S. 8) definiert Tonhöhe als ein wahrnehmbares Attribut eines musikalischen Signals, welches die Anordnung dieses Signals auf einer frequenzbasierten Skala (Tonleiter: von tief zu hoch) erlaubt. Tonhöhe ist somit definiert als Frequenz der Sinuswelle, die dem Zielsignal durch einen Zuhörer zugeordnet werden kann. Wie bereits oben diskutiert, ist die Grundfrequenz nur für periodische oder fast-periodische Signale definiert (als Inverse der Periode).

Eine korrekte Zuordnung der Tonhöhe ist bei komplexen Signalen nicht immer trivial. Hier spielen die Obertöne und andere melodische Effekte eine wichtige Rolle. Eine ausführliche Diskussion dieser Phänomene ist Moore et al. (2010) zu entnehmen. Laut Butler (1992) (S. 39) kann das menschliche Gehör Tonhöhen zw. 20 Hz und 20 kHz wahrnehmen. Der tiefste Musikton ist A0 (27.5 Hz) und der höchste C8 (4 186 Hz)[5].

Die Tonhöhe spielt bei der Einsatzzeiterkennung insofern eine Rolle, als wenn zwei gleiche Töne nacheinander gespielt werden, der zweite Toneinsatz unter Umständen nicht erkannt werden kann – vor allem bei kurzen Anschlagsphasen und demselben Musikinstrument. Das betrifft vor allem solche Fälle, in denen alleine die spektrale Darstellung des Signals als Entscheidungsgrundlage gilt. Denn je ähnlicher die Signale sind, desto schwieriger wird es, eine Veränderung in der spektralen Struktur aufzudecken.

[5] `http://www.sengpielaudio.com/Rechner-notennamen.htm`, Stand: 01.05.2015.

Kapitel 3

Algorithmen zur Toneinsatzzeiterkennung

Die Toneinsatzzeiterkennung (engl. *tone onset detection*) ist der erste Schritt für die Musiktranskription und andere Anwendungen wie die Metrumsanalyse. Aber auch bei der Verbesserung von Hörgeräten spielt die Einsatzzeiterkennung eine wichtige Rolle. Sie stellt die feinste Segmentierung eines musikalischen Signals dar. Im Weiteren wird auf verschiedene Definitionen des Toneinsatzes eingegangen und das klassische Erkennungsschema eingeführt. Neben den etablierten Methoden werden auch eigene Vorschläge vorgestellt.

3.1 Definition

Für die Definition eines Toneinsatzes sind die transienten Signale (Abschnitt 2.3) von entscheidender Bedeutung. Diese sind nicht-periodischer Natur und durch schnelle Änderung der spektralen Struktur erkennbar. Im Falle eines Toneinsatzes entstehen sie durch für die Tonproduktion benötigte Interaktion zwischen Spieler und Instrument und kommen, dementsprechend, in der Anschlagsphase des Tones vor. Zu bemerken ist, dass die transiente Phase kurz nach dem sogenannten *physikalischen Toneinsatz* – dem ersten Amplitudenanstieg vom Nullpunkt – stattfindet. Nach Bello et al. (2005) findet der *Toneinsatz* am *Anfang der transienten Phase* statt.

Polfreman (2013) unterscheidet zwischen einem *wahrnehmbaren Toneinsatz* (engl. *perceptual onset*) als einem Zeitpunkt, an dem eine normal hörende Person den Einsatz frühestens erkennen kann, und *einem wahrnehmbaren Anschlagseinsatz* (engl. *perceptual attack onset*) als einem Zeitpunkt, an dem der Rhythmus (die Tonlänge) des Tones frühestens erkannt bzw. erraten werden kann. Eine von Vos und Rasch (1981) durchgeführte Studie zeigt, dass der wahrnehmbare Toneinsatz zwischen 6 und 15 dB (Dezibel) unter dem maximalen Level eines Tones liegt. Dixon (2006) kritisiert allerdings, dass diese Studie keine komplexen Musiksignale berücksichtigt hat.

Eine passende Definition des Toneinsatzes hängt von dem benutzen Format der Musikdaten ab. Das MIDI-Datenformat enthält notwendige Information über die gespielten Noten, welche dann mittels geeigneter Software in andere Formate (z.B. WAV) umgewandelt werden kann (s. Abschnitt 5.1.1). Die aus MIDI extrahierten Toneinsatzzeiten entsprechen dann den physikalischen Toneinsätzen. Dies ist eine einfache Methode, Musikstücke und die dazugehörigen Einsatzzeiten

zum Training verschiedener Algorithmen zu generieren. Allerdings ist es oft sinnvoll, die Toneinsätze der „realen“ Musikaufnahmen in einer mühsamen Arbeit „per Hand“ zu bestimmen. In diesem Fall scheint die Definition des wahrnehmbaren Toneinsatzes angemessener zu sein. Leider werden in der aktuellen Literatur nur sehr wenige (meist kleine) Datensätze erwähnt, die nach dieser Methode annotiert wurden (Bello et al., 2005; Holzapfel et al., 2010; Böck et al., 2012). Näher wird dieses Thema in Kapitel 5 diskutiert.

Bevor im Weiteren Algorithmen der Einsatzzeiterkennung näher diskutiert werden, ist zu bemerken, dass es zwei Arten der Erkennung gibt: *online* und *offline*. Die *offline* Erkennung ist insofern einfacher, als die Information von dem gesamten Musikstück zu Hilfe genommen werden kann. Dieser Fall ist gut erforscht und liefert für langsame monophone Aufzeichnungen zufriedenstellende Ergebnisse. Einige Anwendungen benötigen dagegen eine *online* bzw. Echtzeit-Erkennung. Dazu zählen beispielsweise Hörgeräte, wobei die Toneinsätze mit minimaler Verzögerung (der sogenannten *Latenz*) erkannt werden sollen. An dieser Stelle ist zu bemerken, dass Menschen – je nach Tempo der Aufnahme – zwei Toneinsätze, die innerhalb von weniger als 20 bis 30 ms liegen, als zeitgleich empfinden (Stowell und Plumbley, 2007).

In dieser Arbeit wird ein neuer Begriff eingeführt: *pseudo-online* Erkennung, welcher durch die Arbeiten von Stowell und Plumbley (2007) und Böck et al. (2012) motiviert ist. In den beiden Veröffentlichungen geht es um die Verbesserung der *online* Toneinsatzerkennung und es wird darauf geachtet, dass keine zukünftige Signalinformation für die Entscheidung zu einem aktuellen Zeitpunkt verwendet wird. Allerdings werden Audioaufnahmen verwendet, deren Amplitude vorab auf das Intervall $[-1,1]$ umskaliert worden ist. Da solche Umskalierung die Information über das Amplitudenmaximum des ganzen Signals benötigt, kann so ein Vorgehen nicht als *online* Erkennung gelten. In dieser Arbeit wird experimentell untersucht, ob und inwieweit sich die Güte der *pseudo-online* Erkennung von der echten *online* Erkennung unterscheidet.

3.2 Allgemeines Schema

Zur Lösung des Einsatzzeiterkennungsproblems wurden in den vergangenen Jahrzehnten zahlreiche Methoden vorgeschlagen. Das detaillierte und gut strukturierte Tutorium von Bello et al. (2005) zu diesem Thema wird hier zusammenfassend vorgestellt sowie um einige später erschienene Ansätze erweitert. Das klassische Vorgehen bei der Einsatzzeiterkennung ist in Algorithmus 3.1 vorgestellt. Nachfolgend findet eine nähere Ausführung der einzelner Schritte statt.

Beispiel 3.2.1 (Aufnahmen des Halleluja Liedes). *Zur Veranschaulichung verschiedener Erkennungsalgorithmen werden in diesem Kapitel zwei monophone Musikstücke analysiert. Es handelt sich dabei um Aufnahmen der ersten Strophe des Halleluja Liedes*[1] *durch zwei verschiedene Musikinstrumente: Klavier und Flöte (Abbildung 3.1). Die vertikalen grauen Linien markieren die*

[1] Auch bekannt als „Ihr seid das Volk, das der Herr sich ausersehn“, `http://www.gesangbuchlieder.de/gesangbuchlieder`, Stand: 01.06.2015.

„per Hand“ ermittelten Toneinsatzzeiten. Das Notenblatt der ersten Strophe ist in Abbildung 3.2 dargestellt.

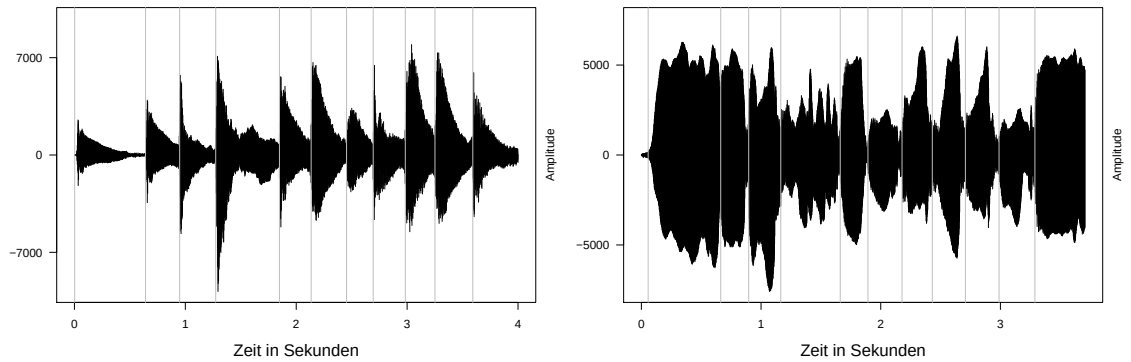

Abbildung 3.1: Die erste Strophe des Halleluja Liedes gespielt von Klavier (links) und Flöte (rechts). Dargestellt ist der Amplitudenverlauf. Die vertikalen Linien markieren die Toneinsatzzeiten.

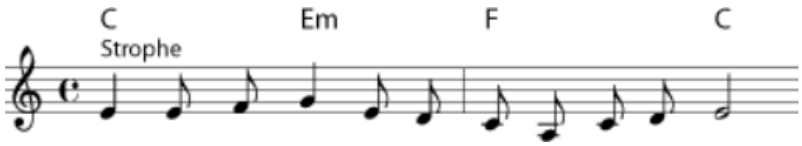

Abbildung 3.2: Das Notenblatt der erste Strophe des Halleluja Liedes.

Algorithmus 3.1: Klassisches Schema der Einsatzzeiterkennung (EZE).

1 Fensterung des Signals in kleine (überlappende) Abschnitte
2 Vor-Verarbeitung des Signals
3 Berechnung eines EZE-Merkmals in jedem Fenster
4 Normalisierung der Merkmalswerte
5 Anwendung einer Schwellenwertfunktion auf die normalisierten Merkmalswerte
6 Lokalisierung der Einsatzzeiten

3.3 Schritt 1: Fensterung des Signals

Wie bereits in Abschnitt 2.5 vorgestellt wurde, wird das eingehende Signal in M (möglicherweise überlappende) Fenster der Länge N Abtastwerte geteilt. Da in jedem solchen Fenster in der Regel die diskrete Fourier Transformation durchgeführt wird, ist es üblich, Potenzen von 2 als Fensterlänge zu wählen (vgl. Abschnitt 2.4). In der gängigen Literatur zur Einsatzzeiterkennung wird oft $N = 2048$ verwendet, was ca. 46 ms bei der Abtastperiode von 44.1 kHz entspricht (Dixon (2006); Rosão et al. (2012); Böck et al. (2012)). Andere Einstellungen kommen jedoch auch vor:

Holzapfel et al. (2010) benutzen beispielsweise $N = 4096$ (ca. 93 ms). Zu bemerken ist, dass kleinere Fenstergrößen eine gute zeitliche Darstellung des Signals ermöglichen und schnell mögliche Änderungen andeuten können. Größere Fensterlängen, dagegen, bieten eine höhere Frequenzauflösung, da bis zu $N/2$ verschiedenen Frequenzlinien die FKen bestimmt werden. Die optimale Wahl der Fensterlänge ist somit ein Trade-off zwischen einer guten spektralen bzw. zeitlichen Auflösung. Für diese Arbeit werden folgende Einstellungen für N betrachtet: 512, 1024, 2048 und 4096.

Bezüglich der Wahl der Sprungweite h (s. Abschnitt 2.5) finden sich in der Literatur sehr unterschiedliche Werte. Nur um einige Beispiele zu nennen, verwenden Holzapfel et al. (2010) 5.6 ms, Dixon (2006) 10 ms und Rosão et al. (2012) 23 ms. Je kleiner h gewählt wird, desto größer ist die Fensterüberlappung. Das bedeutet, dass der wahre Einsatz in mehreren aufeinander folgenden Fenstern zu verzeichnen wäre. Hier wird eine maximale Überlappung von 90% angenommen, d.h. h darf im Bereich von $N/10$ bis N variieren.

3.4 Schritt 2: Vorverarbeitung des Signals

Musiksignale sind komplexe Zeitreihen, die verschiedene wichtige Informationen beinhalten. Eine Vorverarbeitung dieser Zeitreihen kann sich aus mehreren Gründen als vorteilhaft erweisen. Zum einen kann das Signal in mehrere Frequenzbänder aufgeteilt werden, wobei jeder Frequenzbereich gesondert analysiert wird. Zum anderen aber können starke dynamische Schwankungen oder Störgeräusche bereits im Vorfeld eliminiert werden. Die Anwendung der Fensterfunktion auf die Signalamplitude (Abschnitt 2.6) (bei der Bestimmung der FKen) zählt ebenso zu der Vorverarbeitung des Signals.

Frequenzbandaufteilung. Die Frequenzbandaufteilung ist durch das menschliche Gehör motiviert. Dabei reagiert jede der ca. 3000 Nervenzellen des Innenohrs auf eine bestimmte Frequenz und sendet die zugehörigen Aktivierungssignale zur Weiterverarbeitung ins Gehirn. Das eingehende Originalsignal kann entweder mit Hilfe spezieller Filter, die einzelne Frequenzbereiche verstärken, oder basierend auf einem Ohrmodell in Frequenzbänder aufgeteilt werden. Die Anzahl der Frequenzbänder variiert in der Literatur, wobei fünf bis sechs Bänder gebräuchlich sind. Bei einem Ohrmodell sind dagegen mehrere dutzende sogenannte Kanäle üblich.

Nach einer solchen Aufteilung kann jedes Band bezüglich der Toneinsätze gesondert analysiert werden. Allerdings sollte dann die Information aus allen Bändern zu einem endgültigen Vektor der Einsatzzeiten aggregiert werden. Einige Lösungen dieses Problems wurden in Bauer et al. (2014) evaluiert (basierend auf dem Ohrmodell von Meddis (2006)).

In einem von Duxbury et al. (2002) vorgeschlagenen Hybridansatz wird das eingehende Signal ebenso in mehrere Frequenzbänder aufgeteilt, mit der Besonderheit, dass in den oberen Linien amplitudenbasierte Verfahren für die Erkennung ausgeprägter transienter Abschnitte verwendet werden, während in den unteren Linien spektralbasierte Merkmale für die Erfassung der sogenannten weichen Toneinsätze (engl. *soft onsets*) zum Einsatz kommen.

Adaptives Weißen. Eine weitere Vorverarbeitungsmethode – adaptives Weißen (engl. *adaptive whitening*) – ist von Stowell und Plumbley (2007) vorgeschlagen worden. Die wichtigste Idee dabei ist eine signalbasierte Umgewichtung der spektralen Amplitude mit dem Ziel, unterschiedliche Aktivitätsschwankungen jeder Frequenzlinie auf das gleiche Intervall abzubilden:

$$\begin{aligned} \tilde{X}[n,j] &= \begin{cases} \max(|X[n,j]|, r, m \cdot \tilde{X}[n-1,j]), & \text{falls } n > 0, \\ \max(|X[n,j]|, r), & \text{sonst;} \end{cases} \\ X[n,j] &\leftarrow \frac{X[n,j]}{\tilde{X}[n,j]}. \end{aligned} \tag{3.1}$$

m und r sind Gedächtnis- bzw. Rundungsparameter und $X[n,j]$ sind – wie in Abschnitt 2.4 eingeführt – die Fourierkoeffizienten der j-ten Frequenzlinie des n-ten Fensters. Im Grunde werden alle Absolutwerte der FKen, die kleiner als r sind, durch r ersetzt. Deshalb soll r nicht zu groß gewählt werden. Die Autoren schlagen ein zulässiges Intervall zwischen 10^{-6} und 0.2 vor und verwenden $r = 0.1$ für ihre Anwendung. Allerdings benutzen sie Audioaufnahmen, deren Amplitude auf $[-1,1]$ standardisiert wurde (vgl. Abschnitt 3.1). Zu bemerken ist, dass die absoluten Werte von FKen umso größer sind, je stärker die Amplitude des Signals schwankt. Da hier keine solche Annahme über den Amplitudenbereich gemacht wird, sollte das zulässige Intervall für r entsprechend angepasst werden. Dabei sollte die obere Grenze dieses Intervalls im Idealfall größer als der maximale Absolutwert aller FKen sein, damit es möglich wäre, durch Einstellungen von r in der Nähe dieser Werte das adaptive Weißen „auszuschalten“.

Der Gedächtnisparameter m hängt von der sogenannten „*Erholungszeit*“ t_{erh} ab: Zeit für das „Vergessen“ des vergangenen Signals. Die Formel für den Zusammenhang zwischen m und t_{erh} kann dem veröffentlichen Code zu dem Verfahren[2] entnommen werden:

$$m = \exp\left(\frac{-2.3 \cdot h}{t_{erh} \cdot 44.1}\right). \tag{3.2}$$

Stowell und Plumbley (2007) schlagen als zulässiges Intervall für t_{erh} 22 bis 446 Sekunden vor, wobei sie sich – nach einer Optimierung – für ein eher kürzeres Gedächtnis entscheiden ($t_{erh} =$ 25.6 s, d.h. $m = 0.59$). Die Parameterkombinationen in der Nähe von $r = 0$ und $m = 0$ bedeuten, dass die FKen durch deren Absolutwerte geteilt werden und somit alle einen Absolutwert von 1 aufweisen. Man kann dies auch als eine „totale Weißung“ bezeichnen.

[2] `http://onsetsds.sourceforge.net`, Sprache: C/C++, Stand: 01.05.2015.

Der Einfluss des adaptiven Weißen als eine Vorverarbeitungsmethode wird bei Böck et al. (2012) untersucht, wobei sich eine geringe Verbesserung gegenüber des nicht vorverarbeiteten Signals erweist. Die Amplitude der Musikstücke von Böck et al. (2012) wurde allerdings auch auf $[-1,1]$ umskaliert (*pseudo-online* EZE) und die Autoren verwenden $r = 0.005$ für ihre Experimente. Im Rahmen einiger Vor-Experimente, welche der Optimierung in dieser Arbeit zuvorkamen, wurde festgestellt, dass im Fall von *online* Erkennung (ohne Umskalierung der Amplitude) das adaptive Weißen keinen Einfluss auf die Güter der EZE aufweist. Aus diesem Grund wird das adaptive Weißen für die weitere Optimierung nicht berücksichtigt.

Filterung des Spektrums. Die im Weiteren vorgestellte Methode der spektralen Filterung wurde von Böck et al. (2012) für ein spezielles EZE-Merkmal (spektraler Fluss, s. Abschnitt 3.5.3) entwickelt und hat eine relevante Verbesserung der Einsatzzeiterkennungsgüte herbeigebracht. Da sich diese Filterungsmethode jedoch auch für andere spaktralbasierte Merkmale eignet, wird sie in dieser Arbeit als eine mögliche Option für alle Merkmale betrachtet. Der Parameter *spec.filter* (spektraler Filter) wird somit in der nachfolgenden Optimierung zwei Ausprägungen haben: eingeschaltet („Ja“) oder ausgeschaltet („Nein“).

Die Frequenzlinien, zu denen die FKen bestimmt werden, werden dabei in mehrere Bänder aufgeteilt. Böck et al. (2012) verwenden dazu die pseudo Constant-Q Filterbank, bei der die Frequenzen gemäß der Halbtöne der europäischen Musikskala (von 27.5 Hz bis 16 kHz) gebündelt werden. Die resultierende Filterbank $F[n,b]$ enthält dann 82 Frequenzlinien. Werden einer Frequenzlinie mehrere FKen zugeordnet, werden deren spektrale Amplituden mittels eines Dreieck-Fensters gewichtet und aufsummiert. Die gefilterten spektralen Amplituden sind dann gegeben durch (Böck et al., 2012):

$$|X_{filt}[n,b]| = |X[n,j]| \cdot F[n,b]. \tag{3.3}$$

Logarithmierung des Spektrums. Die Logarithmierung des Spektrums wird unter anderem bei Böck et al. (2012) und Eyben et al. (2010) als ein Vorverarbeitungsschritt erfolgreich angewendet. Dabei werden die absoluten FKen mit dem sogenannten Komprimierungsparameter ℓ multipliziert und anschließend logarithmiert. Damit sich keine negativen Logarithmus-Werte ergeben, wird den komprimierten FKen eine Eins dazu addiert.

$$|X^{log}[n,j]| = \log_{10}(\ell|X[n,j]| + 1). \tag{3.4}$$

In Anlehnung an Böck et al. (2012) wird ℓ auf dem Intervall $[0.01, 20]$ optimiert. Der Parameter *spec.log* (mit Ausprägungen „Ja“ und „Nein“) gibt dabei an, ob die Logarithmierung angewendet werden soll oder nicht. Zu bemerken ist, dass die Logarithmierung gegebenenfalls sowohl für die originalen als auch für die gefilterten spektralen Amplituden angewendet wird.

3.5 Funktionen zur Einsatzzeiterkennung

Die Berechnung einer Funktion zur Einsatzzeiterkennung (EZE) auf den Fenstern des (vorverarbeiteten) Signals wird auch als Reduktion (engl. *reduction*) bezeichnet, da nach diesem Schritt nicht mehr das Signal, sondern lediglich ein Merkmalsvektor für die Weiterverarbeitung verwendet wird. Verschiedene EZE-Funktionen wurden in den letzten Dekaden in der Literatur vorgeschlagen: einige nutzen Änderungen in der spektralen Struktur des Signals als Indikator für Toneinsätze, andere betrachten Phasenverschiebungen oder lediglich den Verlauf der Amplitude. In dieser Arbeit werden nur die sogenannten merkmalsbasierten EZE-Funktionen berücksichtigt, die nachfolgend in vier Gruppen (A-D) unterteilt werden – je nach Art der verwendeten Signalinformation. Es existieren auch weitere EZE-Funktionen, die unter anderem bei Bello et al. (2005), Eyben et al. (2010) bzw. Duxbury et al. (2002) erklärt werden. So wird z.B. bei den modellbasierten Funktionen oft eine Verteilungsannahme über den „typischen“ Verlauf des Signals (also ohne einen Toneinsatz) getroffen und für jedes neue Fenster überprüft, ob die Verteilung sich geändert hat. Somit wird angestrebt, ungewöhnliches Verhalten mittels statistischen Testens zu entdecken.

Obwohl die meisten Veröffentlichungen zur Einsatzzeiterkennung (und somit die etablierten Begriffe) auf Englisch sind, wurde in dieser Arbeit bisher stets versucht, passende Begriffe in der deutschen Sprache zu finden. Dies gilt zwar auch für die nachfolgenden Abschnitte, allerdings werden in den Formeln für die Merkmale entsprechende Abkürzungen in englischer Sprache benutzt, um eine schnelle Vergleichbarkeit mit den internationalen Publikationen zu ermöglichen.

Ein Teil der Merkmale berücksichtigt nur die Information aus dem aktuellen Fenster, ein anderer Teil benötigt dagegen auch Informationen aus den vorherigen Fenstern. Damit die Länge des Merkmalsvektors dennoch gleich M bleibt, wird die benötigte Anzahl an zusätzlichen Fenstern am Anfang des Signals hinzugefügt mit Amplitude gleich 0. Solche Merkmale werden (nur bei den Definitionen) je nach Anzahl der zusätzlich benötigten Fenster mit einem bzw. doppeltem Stern gekennzeichnet.

3.5.1 Gruppe A: auf der Amplitudenenergie basierende Merkmale

Nullpunkt-Kreuzungsrate. Eine der einfachsten amplitundenbasierten Merkmale ist die Nullpunkt-Kreuzungsrate (engl. *zero-crossing rate*): Anzahl der Vorzeichenwechsel der Amplitude in einem Fenster. Formal lässt sie sich wie folgt definieren:

$$Zero.Cross.Rate(n) = \frac{1}{N-1} \sum_{k=1}^{N-1} \mathbb{I}\{x[h(n-1)+k] \cdot x[h(n-1)+k+1] < 0\}. \tag{3.5}$$

$\mathbb{I}$ ist die Indikator-Funktion, d.h. sie nimmt den Wert 1 ein, falls die Bedingung zutrifft und sonst den Wert 0. Hier und nachfolgend gilt: $n = 1, \ldots, M$. Die Werte der Nullpunkt-Kreuzungsrate hängen von der Tonhöhe und dem Musikinstrument ab. Innerhalb eines Tones sollte die Rate aber konstant bleiben. Gouyon et al. (2013) verwenden dieses Merkmal für die Klassifizierung von

Schlagzeugklängen. Hier – und für viele weitere Merkmale – ist die absolute Differenz zwischen benachbarten Fenstern besonders relevant. Denn nicht die Merkmalswerte an sich sind von Interesse, sondern eine plötzliche Veränderungen in diesen Werten, wobei die Richtung im Allgemeinen keine Rolle spielt (deswegen wird der Absolutbetrag verwendet). Das durch die Zeitreihe der ersten Differenzen berechnete Merkmal erhält die Endung *.Diff*. Handelt es sich um die absoluten Differenzen, so kommt die Endung *.Abs.Diff* hinzu. Das gewünschte Merkmal lautet:

$$*Zero.Cross.Rate.Abs.Diff(n) = |Zero.Cross.Rate(n) - Zero.Cross.Rate(n-1)|. \tag{3.6}$$

Amplitudenmaximum. Ein weiteres einfaches Merkmal, welches für Saiteninstrumente gute Ergebnisse bei Bauer et al. (2012) erzielt hat, ist die Differenz zwischen den absoluten Maxima der benachbarten Fenster:

$$Ampl.Max(n) = \max\left(|x[h(n-1)+1]|, \ldots, |x[h(n-1)+N]|\right), \tag{3.7}$$

$$*Ampl.Max.Diff(n) = Ampl.Max(n) - Ampl.Max(n-1). \tag{3.8}$$

Hier ist zu bemerken, dass das Vorzeichen der Differenz eine wichtige Rolle spielt: nur positive Differenzen (Amplitudenanstieg) sollten Indikatoren möglicher Toneinsätze sein. Allerdings könnte eine stark ausgeprägte negative Differenz für das rasche Abklingen des vorherigen Tones sprechen, welches einem neuen Toneinsatz unmittelbar vorhergeht.

Die Berücksichtigung der Tonende-Information wurde bereits bei Benetos und Dixon (2011) für die Verbesserung der Musiktranskription implementiert und zwar eine auf dem Hidden Markov Modell basierende Tonende-Erkennung (engl. *offset detection*). Es soll in Rahmen dieser Arbeit experimentell untersucht werden, ob die Tonende-Erkennung – für dieses Merkmal gegeben durch die absolute Differenz der maximalen Amplituden – die Einsatzerkennung verbessern kann. Um solche Merkmale besser kenntlich zu machen, werden sie mit der hochgestellten Markierung *offset* versehen:

$$*Ampl.Max.Abs.Diff^{offset}(n) = |Ampl.Max(n) - Ampl.Max(n-1)|. \tag{3.9}$$

Man sollte aber bei solchen Merkmalen damit rechnen, dass anstatt des Toneinsatzes das Ende des vorherigen Tones (womöglich zu früh) erkannt wird.

Amplitudenenergie. Eine andere Möglichkeit, Amplitudenveränderung zu messen, ist die Aufsummierung aller quadrierten Abtastwerte bzw. der *Energie* innerhalb eines Fensters (Schloss, 1985). Auch hier ist nur der Anstieg der Energie von Interesse:

$$Ampl.Energy(n) = \sum_{k=1}^{N} (x[h(n-1)+k])^2, \tag{3.10}$$

$$*Ampl.Energy.Diff(n) = Ampl.Energy(n) - Ampl.Energy(n-1). \tag{3.11}$$

Bei der Berücksichtigung des Tonendes ist die Richtung der Differenzen zu vernachlässigen:

$$*Ampl.Energy.Abs.Diff^{offset}(n) = |Ampl.Energy(n) - Ampl.Energy(n-1)|. \tag{3.12}$$

3.5.2 Gruppe B: auf der Spektralenergie basierende Merkmale

Spektrale Energie. Ähnlich wie bei der Amplitudenenergie werden hier die quadrierten Absolutwerte der FKen einzelner Frequenzlinien in jedem Fenster aufsummiert. In der Literatur wird dazu der Begriff *spektrale Energie* verwendet.

$$Spec.Energy(n) = \frac{2}{N}\sum_{j=1}^{N/2}(|X[n,j]|)^2. \tag{3.13}$$

Auch hier sind die Zeitreihen der ersten (ggf. absoluten) Differenzen von Interesse:

$$*Spec.Energy.Diff(n) = Spec.Energy(n) - Spec.Energy(n-1), \tag{3.14}$$

$$*Spec.Energy.Abs.Diff^{offset}(n) = |Spec.Energy(n) - Spec.Energy(n-1)|. \tag{3.15}$$

Gewichtete spektrale Energie. Oft kann man einen Toneinsatz bei einzelnen Frequenzlinien deutlich erkennen, während andere Frequenzlinien ein sehr verschwommenes Bild liefern. Das legt nahe, eine gewichtete Summe der spektralen Absolutwerte zu betrachten. Wenn für die Produktion eines Musiktones ein Anschlag nötig ist (wie bei Saiten- oder Schlagzeuginstrumenten), spiegelt sich dieser Anschlag in den höheren Frequenzlinien wieder. Aus diesem Grund hat Masri (1996) eine lineare Gewichtung der Absolutwerte vorgeschlagen. In der Literatur findet man sie unter dem englischen Begriff *high frequency content*:

$$High.Freq.Cont(n) = \frac{2}{N}\sum_{j=1}^{N/2}(j \cdot |X[n,j]|)^2. \tag{3.16}$$

Laut Bello et al. (2005) eignet sich diese Methode allerdings nicht besonders gut für andere Instrumentenklassen (wie z.B. Blasinstrumente) bzw. für die Erkennung der „weichen“ Einsätze. Die beiden Merkmale lauten:

$$*High.Freq.Cont.Diff(n) = High.Freq.Cont(n) - High.Freq.Cont(n-1), \tag{3.17}$$

$$*High.Freq.Cont.Abs.Diff^{offset}(n) = |High.Freq.Cont(n) - High.Freq.Cont(n-1)|. \tag{3.18}$$

Es ist naheliegend, auch andere Gewichtungen in Betracht zu ziehen. Zum Vergleich kann eine der in Abschnitt 2.6 vorgestellten Fensterfunktionen benutzt werden. So würde die Gauss-Fensterfunktion die mittleren Frequenzlinien stärker gewichten. Von Interesse ist, ob auf diese

Weise die oben erwähnten Nachteile behoben werden. Es werden folgende neue Merkmale definiert:

$$Gauss.Freq.Cont.(n) = \frac{2}{N}\sum_{j=1}^{N/2}(w_{N/2}^{Gauss}(j)\cdot|X[n,j]|)^2. \tag{3.19}$$

$$*Gauss.Freq.Cont.Diff(n) = Gauss.Freq.Cont(n) - Gauss.Freq.Cont(n-1), \tag{3.20}$$

$$*Gauss.Freq.Cont.Abs.Diff^{offset}(n) = |Gauss.Freq.Cont(n) - Gauss.Freq.Cont(n-1)|. \tag{3.21}$$

3.5.3 Gruppe C: auf dem Absolutwert der FK basierende Merkmale

Spektraler Schwerpunkt. Bei dem spektralen Schwerpunkt (engl. *spectral centroid*) wird diejenige Frequenzlinie ermittelt, in deren Nähe sich der größte Anteil der spektralen Energie (gemessen an Absolutwerten von FKen) befindet (Peeters und Rodet, 2004). Es handelt sich dabei um einen gewichteten Mittelwert der Frequenzlinien:

$$Spec.Centroid(n) = \frac{\sum_{j=1}^{N/2} j\cdot|X[n,j]|}{\sum_{j=1}^{N/2}|X[n,j]|}, \tag{3.22}$$

Kleinere Werte des spektralen Schwerpunktes entsprechen den tieferen Tönen. Die Änderung des Merkmals (positive oder negative) sollte ein Anzeichen auf einen möglichen Toneinsatz sein. Zum einen kann der Wechsel der Grundfrequenz diese Änderung hervorrufen und zum anderen aber auch ein transientes Signal, dessen spektrale Darstellung sehr unsystematisch aussieht, so dass der Schwerpunkt um die mittlere Frequenzlinie erwartet wird.

$$*Spec.Centroid.Abs.Diff(n) = |Spec.Centroid(n) - Spec.Centroid(n-1)|. \tag{3.23}$$

Spektrale Streuung. Hier wird die spektrale Streuung (engl. *spectral spread*) der einzelnen Signalabschnitte gemessen (Peeters und Rodet, 2004), welche einen Aufschluss über die Klangfarbe bieten soll. Kleine Werte deuten dabei auf obertonärmere Musikinstrumente hin. Dieses Merkmal könnte dann besonders wichtig für die Einsatzzeiterkennung sein, wenn mehrere Musikinstrumente abwechselnd spielen. Auch hier interessieren uns die absoluten Differenzen der Merkmalswerte, welche hier allerdings nicht die Berücksichtigung des Tonendes bedeuten:

$$Spec.Spread(n) = \frac{\sqrt{\sum_{j=1}^{N/2}(j - Spec.Centroid(n))^2|X[n,j]|}}{\sqrt{\sum_{j=1}^{N/2}|X[n,j]|}}. \tag{3.24}$$

$$*Spec.Spread.Abs.Diff(n) = |Spec.Spread(n) - Spec.Spread(n-1)|. \tag{3.25}$$

Spektrale Schiefe. Die spektrale Schiefe (engl. *spectral skewness*) ist ein Maß für die Schiefe der Spektralverteilung über die Frequenzlinien (Peeters und Rodet, 2004). Diese Verteilung ist

für die meisten Musiktöne stark rechtsschief. Je tiefer der Grundton ist und je weniger Obertöne vorhanden sind, desto schiefer ist die Spektralverteilung. Weißes Rauschen oder andere unsystematische Signalkomponenten weisen dagegen eher symmetrische spektrale Verteilungen auf (d.h. der Schiefekoeffizient liegt in der Nähe von 0). Das Merkmal berechnet sich wie folgt:

$$Spec.Skewness(n) = \frac{\sum_{j=1}^{N/2}(j - Spec.Centroid(n))^3 |X[n,j]|}{(Spec.Spread(n))^3 \sum_{j=1}^{N/2} |X[n,j]|}. \tag{3.26}$$

$$*Spec.Skewness.Abs.Diff(n) = |Spec.Skewness(n) - Spec.Skewness(n-1)|. \tag{3.27}$$

Spektraler Fluss. Der spektrale Fluss (engl. *spectral flux*) ist wegen seiner besonders guten Erkennungsgüte (Bello et al. (2005); Dixon (2006); Rosão et al. (2012); Bauer et al. (2015); Böck et al. (2012)) eines der populärsten Merkmale zur Einsatzzeiterkennung. Die grundlegende Idee besteht darin, Differenzen der Absolutwerte von FKen der benachbarten Fenster für jede Frequenzlinie zu betrachten. Bei der klassischen Variante wird ein Filter $H(x) = \frac{x+|x|}{2}$ auf diese Differenzen angewendet mit dem Ziel, negative Veränderungen gleich Null zu setzen. Dies wird damit begründet, dass man nicht an der Erkennung des Tonendes, bei dem die spektrale Intensität abnimmt, interessiert ist. Die Formel für den spektralen Fluss lautet wie folgt:

$$*Spec.Flux(n) = \sum_{j=1}^{N/2} H(|X[n,j]| - |X[n-1,j]|). \tag{3.28}$$

Als Vergleich zur klassischen Definition wird im Weiteren eine alternative Definition in Betracht gezogen, die auf dem euklidischen Abstand basiert. Dabei werden sowohl positive als auch negative Änderungen der Absolutwerte berücksichtigt. Dieses Merkmal ist sehr intuitiv und wurde bereits bei Bauer et al. (2012) verwendet:

$$*Spec.Euclid^{offset}(n) = \sum_{j=1}^{N/2} (|X[n,j]| - |X[n-1,j]|)^2. \tag{3.29}$$

3.5.4 Gruppe D: auf dem Absolutwert und der Phase von FKen basierende Merkmale

Schwankung der Phase. Während bisher nur die Absolutwerte der FKen berücksichtigt wurden, wird nun die Schwankung der Phase (engl. *phase deviation*) analysiert. Die Phase enthält nämlich Information über die zeitliche Struktur des Signals (Bello et al., 2005). Formel (2.24) zeigt die Darstellung der komplexen Zahlen in Polarkoordinaten. In jedem Fenster lässt sich somit zu jeder Frequenzlinie der Wert der Phase $\phi[n,j]$ ausrechnen. Es wird erwartet, dass innerhalb eines Tones der Zuwachs der Phase zwischen den benachbarten Fenstern etwa konstant bleibt (Eyben et al., 2010). Änderung der Grundfrequenz oder ein transienter Abschnitt würde einen schwächeren bzw. stärkeren Zuwachs als Folge haben. Aus diesem Grund ist für dieses Merkmal

die Zeitreihe der zweiten Differenzen von Interesse:

$$**Phase.Dev(n) = \frac{2}{N}\sum_{j=1}^{N/2} |\phi''[n,j]|, \tag{3.30}$$

$$\phi''[n,j] = \phi[n,j] - 2\phi[n-1,j] + \phi[n-2,j]. \tag{3.31}$$

Dixon (2006) hat vorgeschlagen, die Veränderung der Phase durch den Absolutwert der entsprechenden FKen zu gewichten (gewichtete Schwankung der Phase, engl. *weighted phase deviation*). Somit sollen besonders stark ausgeprägte Frequenzlinien mehr Einfluss auf die Merkmalswerte erlangen. Wenn mit den anteiligen Absolutwerten gewichtet wird, nennt sich dieses Merkmal die *normalisierte* gewichtete Schwankung der Phase. Hier wird die folgende Definition verwendet:

$$**Norm.Weight.Phase.Dev(n) = \frac{\sum_{j=1}^{N/2} |X[n,j]\phi''[n,j]|}{\sum_{j=1}^{N/2} |X[n,j]|}. \tag{3.32}$$

Komplexe Domäne. Es gibt verschiedene Wege, Absolutbetrag und Phase der FKen simultan zu berücksichtigen. Dixon (2006) erweitert und vereinfacht einen bereits existierenden Ansatz – die sogenannte komplexe Domäne (engl. *complex domain*), bei der die FKen des aktuellen Fensters aufgrund der Information aus zwei vorherigen Fenstern unter der Annahme eines stationären Signals geschätzt werden. Anschließend wird die Schätzung mit den tatsächlich beobachteten Koeffizienten verglichen, wobei große Abweichungen auf einen möglichen Toneinsatz hinweisen:

$$**Compl.Domain^{offset}(n) = \frac{2}{N}\sum_{j=1}^{N/2} |X[n,j] - \hat{X}[n,j]|, \tag{3.33}$$

$$\hat{X}[n,j] = |X[n-1,j]|e^{i(2\phi[j,n-1]-\phi[j,n-2])}. \tag{3.34}$$

Dixon (2006) hat weiterhin eine ähnliche Korrektur wie bei spektralem Fluss vorgeschlagen, um den Toneinsatz vom Tonende zu unterschieden. Hier werden die Unterschiede zwischen den tatsächlichen und den geschätzten FKen für eine Frequenzlinie nur dann berücksichtigt, wenn der Absolutwert der FKen für diese Frequenzlinie ansteigt. Die gerichtete komplexe Domäne (engl. *rectified complex domain*) lässt sich somit folgendermaßen ausrechnen:

$$**Rect.Compl.Domain(n) = \sum_{j=1}^{N/2} RCD(n,j), \tag{3.35}$$

$$RCD(n,j) = \begin{cases} |X[n,j] - \hat{X}[n,j]|, & \text{falls } |X[n,j]| > |X[n-1,j]| \\ 0, & \text{sonst.} \end{cases} \tag{3.36}$$

3.5.5 Gegenüberstellung der Merkmale

Das nachfolgende Beispiel 3.5.1 dient der Veranschaulichung der EZE-Merkmalsfunktionen. Eine wichtige Erkenntnis konnte dank dieses Beispiels gewonnen werden: die Merkmale Amplitudenenergie (Gruppe A) und spektrale Energie (Gruppe B) haben exakt die gleichen Ausprägungen. Das gleiche gilt naturgemäß auch für deren (absoluten) Differenzen. Dies scheint zuerst überraschend zu sein, erklärt sich aber leicht durch die Hin- und Rücktransformation der Fourieranalyse (Abschnitt 2.6). Aus diesem Grund werden die beiden Merkmale der Gruppe B *Spec.Energy.Diff* und $Spec.Energy.Abs.Diff^{offset}$ im Weiteren nicht berücksichtigt. Obwohl zwei andere Merkmale: *High.Freq.Cont.Diff* und *Gauss.Freq.Cont.Diff* dem Verlauf des *Spec.Energy.Diff* Merkmals auch sehr ähnlich sind, werden sie dennoch weiter zur Optimierung herangezogen, weil nicht ausgeschlossen werden kann, dass die scheinbaren Ähnlichkeiten nur für das betrachtete Beispiel bestehen.

Beispiel 3.5.1 (Veranschaulichung der Merkmale zur EZE). *Die eingeführten Merkmale zur EZE werden auf die Klavier- und Flötenaufnahme des Halleluja Liedes (vgl. Beispiel 3.2.1) angewendet. Alle Merkmale sind im Anhang A in Abbildungen A.1, A.2, A.3 und A.4 zu finden. Aus den Abbildungen lässt sich ableiten, dass je nach Musikstück große Unterschiede im Verlauf der EZE-Merkmale zu verzeichnen sind. Für die Klavieraufnahme sind bei den meisten Merkmalen eine deutliche Spitze in der Umgebung des wahren Toneinsatzes zu sehen. Am schlechtesten schneiden die auf der Spektralverteilung basierenden Merkmale aus Gruppe C ab: der spektrale Schwerpunkt, die spektrale Streuung und die spektrale Schiefe. Der spektrale euklidische Abstand (welcher das Tonende berücksichtigt) scheint für die Flöte besser zu funktionieren als der spektrale Fluss. Der gleiche Schluss kann auch für die auf der absoluten Differenz basierenden Merkmale aus Gruppe A und B gezogen werden. Merkmale aus Gruppe D, die sowohl den Absolutwert als auch die Phase des Signals einbeziehen, erweisen sich für die Flöte als weniger geeignet.*

3.6 Schritt 4: Normalisierung

Sei $\boldsymbol{odf} = (odf_1, \ldots, odf_M)^T$ der nach dem Anwenden einer EZE-Funktion zustande gekommene Vektor der Merkmalswerte[3]. Je nach Merkmal wird dieser Vektor sehr unterschiedlich skaliert sein. Das Ziel der Normalisierung ist die Glättung und die Umskalierung der Merkmalswerte. Das Glätten (engl. *smoothing*) erfolgt üblicherweise mit Hilfe der exponentiellen Glättung. Der Vektor der geglätteten Merkmalswerte $\boldsymbol{sm.odf} = (sm.odf_1, \ldots, sm.odf_M)^T$ lässt sich folgendermaßen berechnen:

$$\begin{aligned} sm.odf_1 &= odf_1, \\ sm.odf_n &= \alpha \cdot odf_n + (1-\alpha) \cdot sm.odf_{n-1}. \end{aligned} \tag{3.37}$$

[3] In dieser Arbeit werden die Vektoren fett markiert. Die Elemente der Vektor besitzen den gleichen Namen und werden mit einem tiefgestellten Index versehen.

Der Parameter α bestimmt die Stärke der Glättung: Bei $\alpha = 1$ bleibt die Zeitreihe unverändert (d.h. $\boldsymbol{sm.odf} = \boldsymbol{odf}$) während bei $\alpha = 0$ alle $sm.odf_n$ Werte gleich odf_1 sind.

Es gibt zwei klassische Umskalierungs- bzw. Standardisierungsverfahren. Bei dem Ersten wird die Merkmalsreihe auf das Intervall $[0,1]$ umskaliert:

$$st.odf_n = \frac{sm.odf_n - \min(\boldsymbol{sm.odf})}{\max(\boldsymbol{sm.odf}) - \min(\boldsymbol{sm.odf})}, \tag{3.38}$$

wobei $\boldsymbol{st.odf} = (st.odf_1, \ldots, st.odf_M)^T$ der Vektor der umskalierten Werte ist. Bei der aus der Statistik motivierten Standardisierung wird von dem Merkmalsvektor sein Mittelwert abgezogen und das Ergebnis durch seine Standardabweichung geteilt:

$$st.odf_n = \frac{sm.odf_n - mean(\boldsymbol{sm.odf})}{sd(\boldsymbol{sm.odf})}. \tag{3.39}$$

In diesem Fall werden $mean(\boldsymbol{st.odf}) = 0$ und $sd(\boldsymbol{st.odf}) = 1$ garantiert. Allerdings sind $\min(\boldsymbol{st.odf})$ und $\max(\boldsymbol{st.odf})$ unbekannt. Zu bemerken ist, dass die beiden definierten Umskalierungsverfahren im Allgemeinen nur für die *offline* Erkennung einsetzbar sind. Für *online* Verfahren kann die Standardisierung nicht angewendet werden.

Beispiel 3.6.1 (Veranschaulichung der Glättung). *Der Effekt der Glättung wird in Abbildung 3.3 am Beispiel des Merkmals High.Freq.Cont.Abs.Diff^{offset} (Formel (3.18)) sowohl für die Flöte als auch für das Klavier veranschaulicht (für $\alpha = 0.5$ und $\alpha = 0.7$). Weitere Parameter: $N = h = 1024$ (keine Überlappung), window.fun = „Hanning“, spec.filter = spec.log = „Nein„. Für $\alpha = 0.5$ ist die Glättung definitionsgemäß stärker ausgeprägt: zwar werden kleine Schwankungen zum Teil eliminiert, aber die Höhe der Spitzen verkleinert sich dementsprechend auch. Ob eine starke Glättung sich vorteilhaft oder nachteilig auf die Güte der Erkennung auswirkt, kann sich nur mittels der Optimierung zeigen.*

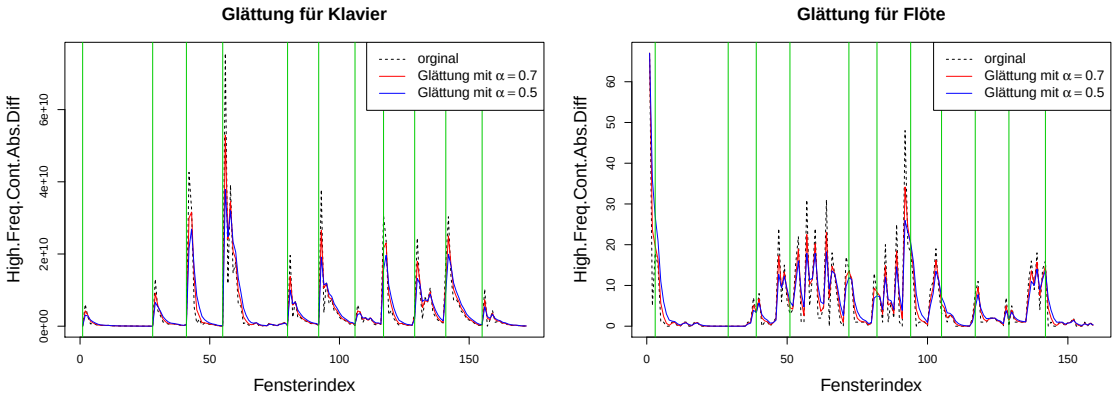

Abbildung 3.3: Effekt der Glättung am Beispiel des *High.Freq.Cont.Abs.Diff^{offset}* Merkmals für Klavier (links) und Flöte (rechts) für die Aufnahmen des Halleluja Liedes (s. Beispiel 3.2.1). Die vertikalen Linien markieren die wahren Toneinsatzzeiten.

3.7 Schritt 5: Schwellenwertfunktion

Der normalisierte Merkmalsvektor ***st.odf*** enthält weniger und stärker ausgeprägte Spitzen, welche auf einen möglichen Toneinsatz hindeuten. Da jedoch nicht jedes lokale Maximum einem Einsatz entspricht, ist die Definition einer Schwelle notwendig, die zwischen „relevanten“ und „nicht-relevanten“ Schwankungen unterscheidet. Ein fester Wert als Schwelle ist unvorteilhaft, weil man damit nicht auf dynamische Änderungen des Signals reagieren könnte. Verbreitet sind daher die gleitende Schwellenwertfunktionen (engl. *thresholding functions*):

$$T_n = \delta + \lambda \cdot mov.fun(|st.odf_{n-l_T}|, \ldots, |st.odf_{n+r_T}|), \tag{3.40}$$

wobei *mov.fun* (von engl. *moving function*) entweder der Median oder das arithmetische Mittel ist. l_T und r_T sind die Anzahl Fenstern, die links bzw. rechts von dem n-ten Fenster liegen und in die Berechnung von *mov.fun* eingehen.

In den meisten Veröffentlichungen ist $l_T = r_T$. Für die Echtzeitanwendungen soll r_T entweder sehr klein oder als Null gewählt werden (je nach Fenstergröße und Sprungweite), um die Latenzzeit so gering wie möglich zu halten. Allerdings sollte die Schwellenwertfunktion für die Echtzeitanwendungen etwas modifiziert werden, da in Formel (3.40) von einem normalisierten Merkmalsvektor ausgegangen wird, so dass sich der Schwankungsbereich für δ gut festlegen lässt auf z.B. $[0,1]$.

Vorschlag für eine Echtzeit-kompatible Einsatzzeiterkennung. Ein erster Vorschlag für die Echtzeiterkennung wäre, nur den multiplikativen Parameter λ zu betrachten. In diesem Fall ist eine Standardisierung nicht notwendig. Böck et al. (2012) betrachten für ihren Echtzeit-Erkennungsansatz dagegen nur den Parameter δ, welchen sie für jede Erkennungsfunktion gesondert optimieren. Dies ist nämlich angesichts der unterschiedlichen Merkmalsskalierung notwendig. Hier wird jedoch kein merkmalsabhängiges Training angestrebt, deswegen wird auf diesen Ansatz verzichtet. Nichtsdestotrotz wird der Parameter δ für die weitere Optimierung mitberücksichtigt. Zum einen um eine besser Vergleichbarkeit mit dem Algorithmus von Böck et al. (2012) zu erreichen und zum anderen, weil die modellbasierte Optimierung auch die Interaktionen zwischen dem Merkmal und δ mitberücksichtigen sollte.

Als eine weitere Lösung wird hier eine auf den Verteilungsquantilen basierende Schwellenwertfunktion vorgestellt. In Bauer et al. (2013b) wurde bereits ein Quantil des ***odf*** Vektors als ein fester Schwellenwert eingeführt[4]. Die Verwendung des Quantils ist intuitiv, da man a-priori weiß, dass der Anteil der Fenster, in denen ein Toneinsatz vorkommt, zwar von der Fensterlänge, der Sprungweite, dem Tempo und anderen Gegebenheiten eines Musikstücks abhängt, in der

[4] In dem Paper wird der optimale Wert des Quantils aus dem Intervall [1,30] unter verschiedenen Umständen ermittelt. Dabei ist dieses Intervall als 1% bis 30% der größten Werte des Merkmalsvektors zu verstehen. In dieser Arbeit wird die klassische Definition des p-Quantils verwendet.

Regel aber viel kleiner als der Anteil der Einsatz-freien Fenster ist. Bei einem festen Schwellenwert haben sich in Bauer et al. (2013b) (je nach oben genannten Einflussgrößen) die 82%- bis 98%-Quantile als optimal erwiesen.

In Anlehnung an vorherige Überlegungen wird ein gleitendes p-Quantil als Schwellenwertfunktion vorgeschlagen:

$$T_n = \delta + mov.quantile_p(|st.odf_{n-l_T}|, \ldots, |st.odf_{n+r_T}|), \tag{3.41}$$

Somit würden folgende drei Echtzeit kompatible Schwellenwertfunktionen in Frage kommen: gleitender Mittelwert bzw. gleitender Median (Formel (3.40)) mit δ aus $[0,10]$ und λ aus $[1.1,2.6]$ und gleitendes Quantil (Formel (3.41)) mit δ aus $[0,10]$ und p aus $[0.8,0.98]$. Das Optimierungsintervall für λ wurde im Vergleich mit Bauer et al. (2015) etwas geschrumpft, weil sich dort eher kleinere λ Werte (um den Wert von 1 bis 1.5) in Kombination mit sehr kleinen δ Werten als erfolgreich erwiesen haben. Der zulässige Bereich für p orientiert sich an in Bauer et al. (2013b) ermittelte Werte und empirische Beobachtungen. Zu bemerken ist, dass im Normalisierungsschritt keine Standardisierung mehr erforderlich ist (***st.odf*** $=$ ***sm.odf***), was zu einer besseren Vergleichbarkeit zwischen *offline* und *online* Verfahren führt.

Damit aber hier kein Problem der bedingten Optimierung entsteht – denn je nach Funktion sollen verschiedene Parameter optimiert werden – wird hier eine Vereinbarung getroffen, die sich aus der Beobachtung erschließt, dass je größer λ und p sind, desto weniger Toneinsätze erkannt werden. Es wird ein Hilfsparameter *mov.par* eingeführt, der auf dem Intervall $[0,1]$ optimiert werden soll. Es gilt:

$$\begin{cases} \lambda = 1.5 \cdot mov.par + 1.1, & \text{falls } mov.fun = mov.median \text{ oder } mov.fun = mov.mean, \\ p = 0.18 \cdot mov.par + 0.8, & \text{falls } mov.fun = mov.quantile. \end{cases} \tag{3.42}$$

Zu bemerken ist, dass die korrektere Lösung dieses Problems wäre eine separate Optimierung der EZE mit dem gleitenden Quantil als Schwellenwertfunktion.

Beispiel 3.7.1 (Schwellenwertfunktionen). *In Abbildung 3.4 sind die drei erwähnten Schwellenwertfunktionen veranschaulicht: gleitender Mittelwert (oben), gleitender Median (mittig) und gleitendes Quantil (unten). Hier handelt es sich – genauso wie in Beispiel 3.6.1 – um das Merkmal High.Freq.Cont.Abs.Diff^{offset} für Klavier (links) und Flöte (rechts). Der Fokus dieses Beispiels liegt auf der Veranschaulichung des Verhaltens der online (die blauen Linien) und der offline (die roten Linien) Ansätze, die sich aus der Einstellung des r_T Parameters ergeben. $r_T = 1$ kann als eine online Erkennung gelten, wenn die erlaubte Latenzzeit (z.B. Definiert durch Verzögerung der Signalwiedergabe in einem Hörgerät) dadurch nicht überschritten wird. Für die beiden Ansätze gilt: $r_l = 7$, $\alpha = 0.7$ und $\delta = 0$. Alle weiteren Parameter wurden in Beispiel 3.6.1 bereits erwähnt.*

Allgemein ist zu sehen, dass die roten Linien (offline Erkennung) etwas vorteilhafter verlaufen: sie erreichen ein höheres Niveau bereits vor dem Anstieg des Merkmalswertes, während die

blauen Linien erst mit dem Eintreten dieses Einstiegs wachsen. Der gleitende Median scheint am schlechtesten zu funktionieren: als ein Ausreißer robustes Mittel „ignoriert“ er einige Zeit den plötzlichen Schwung der Merkmalsfunktion und wächst erst dann, wenn mindestens die Hälfte der Beobachtungen in dem Schwellenwert-Intervall wesentlich größer geworden sind. Da bei dem online Verfahren das Schwellenwert-Intervall kleiner ist als bei dem offline Verfahren (8 Beobachtungen gegenüber von 17), wächst die blaue Linie stärker als die rote, obgleich mit größerer Verzögerung. Die Methoden der gleitenden Mittelwerte und der gleitenden Quantile scheinen sowohl aus theoretischer Sicht als auch aus dem dargestellten Beispiel angemessener für die beiden Arten der Erkennung zu sein.

3.8 Schritt 6: Lokalisierung der Toneinsätze

Der letzte Schritt ist die Lokalisierung der Fenster, in denen ein Toneinsatz geschätzt wird. Bei Rosão et al. (2012) sollen zwei Bedingungen für solche Fenster erfüllt sein: Der Wert der EZE-Funktion soll den gleitenden Schwellenwert übersteigen und ein lokales Maximum sein. In Anlehnung an Böck et al. (2012) wird hier eine zusätzliche Bedingung eingebaut: Es soll ein Mindestabstand *min.dist* (in Anzahl der Fenster) zwischen dem aktuellen Fenster und dem Fenster mit dem letzten Toneinsatz $n_{last.onset}$ eingehalten werden:

$$O_n = \begin{cases} 1, & \text{falls } st.odf_n > T_n \text{ und} \\ & st.odf_n = \max(st.odf_{n-l_O}, \ldots, st.odf_{n+r_O}) \text{ und} \\ & n > n_{last.onset} + min.dist, \\ 0, & \text{sonst.} \end{cases} \tag{3.43}$$

$\boldsymbol{O} = (O_1, \ldots, O_M)^T$ ist der Toneinsatz-Vektor. l_O und r_O sind zusätzliche Parameter: Anzahl der Fenster links bzw. rechts von dem aktuellen Fenster für die Berechnung der lokalen Maxima. Bei zwei dicht aufeinander folgenden Spitzen des EZE-Merkmals, die die gleitende Schwelle übersteigen, wobei die zweite Spitze größer als die erste ist, würden kleinere Werte von l_O, r_O und *min.dist* dazu beitragen, dass die beiden Spitzen als ein Toneinsatz erkannt sein würden. Größere Werte von l_O und r_O und kleine minimale Distanz, dagegen, würden nur zur Erkennung der zweiten Spitze führen. In der umgekehrten Situation würde nur die erste Spitze den Einsatz markieren.

Um von dem Vektor $\boldsymbol{O}$ zu dem Vektor der Einsatzzeiten in Sekunden zu gelangen, werden zu allen Fenstern n mit $O_n = 1$ deren Startzeitpunkte ermittelt. Ein so ermittelter Toneinsatz wird als richtig erkannt bezeichnet, wenn er innerhalb eines Toleranzintervalls um den wahren Einsatz liegt. Zwei Einstellungen für dieses Intervall kommen in der Literatur häufig vor: $\pm$50 ms (Bello et al., 2005; Dixon, 2006) und $\pm$25 ms (Collins, 2005; Böck et al., 2012). Für die Optimierung wird in dieser Arbeit das $\pm$25 ms Toleranzintervall verwendet.

Verschiebung der geschätzten Toneinsatzzeiten. Böck et al. (2012) schlagen vor, die Toneinsätze ein Fenster später als tatsächlich erkannt zu notieren. Dies begründen sie damit, dass

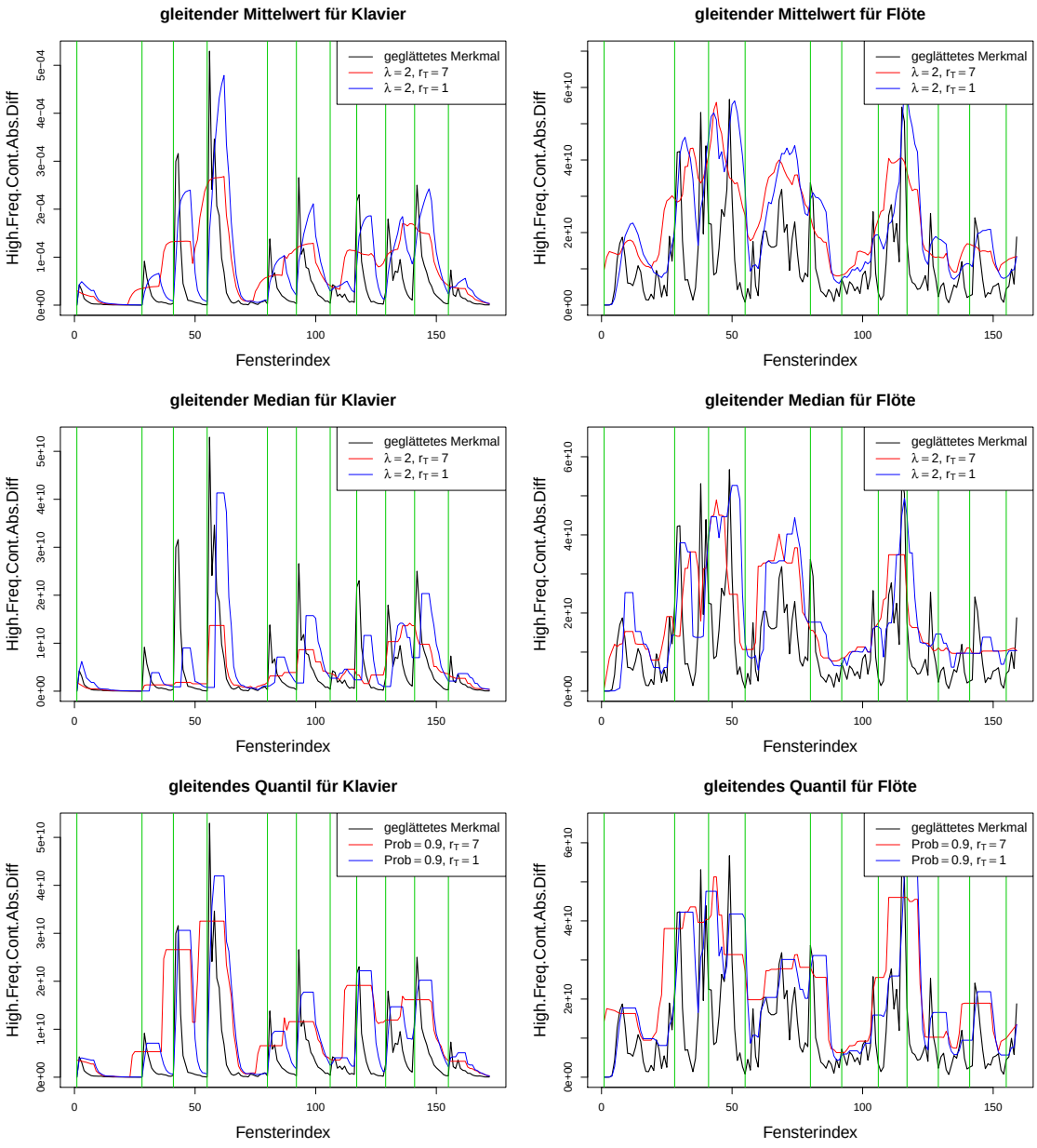

Abbildung 3.4: Schwellenwertfunktionen am Beispiel des *High.Freq.Cont.Abs.Diff*^{offline} Merkmals für Klavier (links) und Flöte (rechts). Die vertikalen Linien markieren die wahren Toneinsatzzeiten. Die blauen Linien entsprechen der *online* Erkennung und die roten der *offline* Erkennung.

die betrachteten Musikstücke per Hand annotiert wurden und die wahren Einsatzzeiten somit den wahrnehmbaren Toneinsatzzeiten entsprechen würden (s. Abschnitt 3.1). Die EZE-Algorithmen würden dagegen die physikalischen Toneinsätze erkennen. Für die in Böck et al. (2012) verwendete Sprungweite ergibt sich eine Verschiebung der geschätzten Toneinsatzzeiten um 10 ms nach rechts.

In dieser Arbeit wird der Parameter *onset.shift* auf dem Intervall $[-0.01, 0.02]$ s optimiert. Der negative Bereich wurde deswegen mitberücksichtigt, weil es nicht ausgeschlossen werden kann, dass die Toneinsätze je nach verwendetem Merkmal mit einem Verzug erkannt werden.

Anzahl der Fenster als Funktion der Zeit. Im Gegensatz zu den meisten gängigen Veröffentlichungen, in denen N und h vorab festgelegt werden, werden diese hier optimiert. So kann z.B. $r_T = 5$ je nach Fensterlänge und Sprungweite einen sehr großen zeitlichen Unterschied bedeuten: Für $N = 1024$ und $h = 512$ sind es 52 ms, die man (zusätzlich zu den 23 ms des Fensters) in die Zukunft „schauen“ soll, für $N = 4096$ und $h = 2048$ sind es aber schon zusätzliche 232 ms. Um damit sinnvoll umzugehen, werden r_T und l_T (in Formeln (3.41) und (3.40)), sowie l_O, r_O und *min.dist* (in Formel (3.43)) als Funktionen der gewünschten zeitlichen Begrenzung, N und h definiert:

$$r_T = n_{Fenster}(t(r_T), N, h). \tag{3.44}$$

Die Ausgabe der $n_{Fenster}$ Funktion ist die Anzahl der Fenster, die die zeitliche Begrenzung $t(r_T)$ nicht überschreiten. Soll z.B. 35 ms in die Zukunft „geschaut“ werden bei $N = 1024$ und $h = 512$ (zusätzlich zu 23 ms des Fensters), so ist $r_T = 3$. Somit werden im Weiteren nicht die Parameter r_T, l_T, r_O, l_O und *min.dist* optimiert, sonder die Zeiten $t(r_T)$, $t(l_T)$, $t(r_O)$, $t(l_O)$ und $t(min.dist)$. Für Echtzeitanwendungen sollen $t(r_O)$ und $t(r_T)$ möglichst klein sein (wegen der Latenzzeit). Hier werden diese Zeiten auf 0 festgesetzt, sodass bei der *online* EZE zwei Parameter weniger optimiert werden müssen.

In Tabelle 3.2 sind alle einstellbaren Parameter des vorgestellten EZE-Schemas sowie deren zulässige Ausprägungen bzw. Intervalle für die nachfolgende Optimierung aufgelistet.

3.9 Güte der Einsatzzeiterkennung

3.9.1 F-Wert

Die Messung der Güte der Einsatzzeiterkennung ist ein nicht triviales Problem. Fest steht, dass pro Fenster eine Entscheidung über einen möglichen Einsatz getroffen wird, wobei die wahren Toneinsatz-Informationen zu den Fenstern auch vorliegen. Es ist daher naheliegend, die bei den Klassifikationsverfahren üblichen Kontingenztafel basierten Maße wie *Recall* und *Precision* zu betrachten (s. Tabelle 3.1, mit Anzahl der Fenster $M = TP + TN + FP + FN$):

$$Recall = \frac{TP}{TP + FN}, \quad Precision = \frac{TP}{TP + FP}. \tag{3.45}$$

Precision ist der Anteil der korrekten Erkennungen unter allen erkannten Einsätzen. *Recall* ist dagegen der Anteil der korrekten Erkennungen unter allen wahren Einsätzen. Die beiden Maße sind gegenläufig: je höher die Precision wird (meistens erreicht durch verminderte Anzahl an Einsatzschätzungen), umso kleiner wird der Recall (Salfner et al., 2010).

Van Rijsbergen (1979) stellte ein kombiniertes Maß – den F-Wert – vor, das als harmonisches Mittel der beiden Werte berechnet wird:

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \quad F \in [0,1]. \tag{3.46}$$

Tabelle 3.1: Kontingenztafel der Klassifikation

	Einsatz (wahr)	*Kein Einsatz (wahr)*
Einsatz (geschätzt)	korrekte Erkennung (*TP*: *true positive*)	falsche Erkennung (*FP*: *false positive*)
Kein Einsatz (geschätzt)	falsche Nicht-Erkennung (*FN*: *false negative*)	richtige Nicht-Erkennung (*TN*: true negative)

Der F-Wert berücksichtigt allerdings nicht die *true negativen* Fälle. Wie Salfner et al. (2010)[5] bemerken, ist eine angemessene *TN*-Berücksichtigung schwierig, da man ein Problem der seltenen Ereignisse hat, so dass das *TN*-Feld der Kontingenztafel alle anderen Felder stark dominiert. Eine einfache Regel: „Erkenne keine Einsätze“ würde bereits zu sehr niedrigen Fehlklassifikationsraten führen. Wenn zwei Musikstücke allerdings den gleichen F-Wert aufweisen, eines davon aber zehnfach so viele Einsätze beinhaltet als das andere, könnte nur die Hinzunahme des *TN*-Wertes der Identifikation der „besseren“ Erkennungsrate beitragen. Nichtsdestotrotz, weder in Salfner et al. (2010) noch in zahlreichen weiteren dort erwähnten Artikeln wird eine *TN*-basierte Alternative für den F-Wert vorgestellt.

Der entscheidende Unterschied zwischen der Anwendung des F-Wertes bei der Einsatzzeiterkennung und bei allgemeinen Klassifikationsproblemen liegt in der Definition einer korrekten Erkennung. Nicht die fensterweise Übereinstimmung der beiden (0,1)-Vektoren ist hier nämlich von Interessen, sondern der Abstand zwischen den geschätzten und wahren Einsatzzeiten. So mag bei sehr großen Fensterlängen zwar ein Einsatz in dem richtigen Fenster geschätzt worden zu sein, zeitlich kann er aber außerhalb des Toleranzintervalls liegen und somit zu den falsch positiven Einsätzen zählen. Die Diskussion über die Anwendung der Klassifikationsverfahren für Einsatzzeiterkennung wird in Kapitel 4 fortgeführt.

[5] Die Autoren betrachten zwar ein anderes Problem: eine *online* Fehlverhalten-Vorhersage der Computeranwendungen, die grundlegenden Ideen stimmen aber mit der Einsatzzeiterkennung überein: seltene Ereignisse in der Zeitskala, Fensterung der Eingansinformation, fensterbasierte Entscheidungen und eine zeitbasierte Toleranz für die Definition einer korrekten Fehlerwarnung.

Tabelle 3.2: Zusammenfassung der Parameter zur Einsatzzeiterkennung

Parameter	*Bezeichnung*	*Formel/Abschn.*	*Ausprägungen bzw. Intervall*
Fensterlänge	N	Abschn. 3.3	$512, 1024, 2048, 4096$
Sprungweite	h	Abschn. 3.3	$[N/10, N]$
Fensterfunktion	*window.fun*	2.19 2.20 2.21 2.22	*Rectangular* *Hanning* *Blackman* *Gauss* ($\sigma = 0.4$)
Glättungsparameter	α	3.37	$[0, 1]$
Filterung des Spektrums	*spec.filter*	3.3	Ja, Nein
Logarithmierung des Spektrums	*spec.log*	3.4	Ja, Nein
Parameter der Logarithmierung	ℓ	3.4	$[0.01, 20]$
Funktion zur Einsatzzeiterkennung	*odf*	3.6 3.8 3.9 3.11 3.12 3.17 3.18 3.20 3.21 3.23 3.25 3.27 3.28 3.29 3.30 3.32 3.33 3.35	*Zero.Cross.Rate.Abs.Diff* *Ampl.Max.Diff* *Ampl.Max.Abs.Diff*^{offset} *Ampl.Energy.Diff* *Ampl.Energy.Abs.Diff*^{offset} *High.Freq.Cont.Diff* *High.Freq.Cont.Abs.Diff*^{offset} *Gauss.Freq.Cont.Diff* *Gauss.Freq.Cont.Abs.Diff*^{offset} *Spec.Centroid.Abs.Diff* *Spec.Spread.Abs.Diff* *Spec.Skewness.Abs.Diff* *Spec.Flux* *Spec.Euclid*^{offset} *Phase.Dev* *Norm.Weight.Phase.Dev* *Compl.Domain*^{offset} *Rect.Compl.Domain*
Schwellenwertfunktion	*mov.fun*	3.40 3.40 3.41	*mov.median* *mov.mean* *mov.quantile*
Additiv. Parameter der Schwellenwertfunktion	δ	3.40	$[0, 10]$
Hilfsparameter der Schwellenwertfunktion	*mov.par*	3.42	$[0, 1]$
Zeit in die Zukunft für Schwellenwertfunktion	$t(r_T)$	3.40, 3.41	$[0, 0.5]$ s (*offline*), 0 s (*online*)
Zeit in die Vergangenheit für Schwellenwertfunktion	$t(l_T)$	3.40, 3.41	$[0, 0.5]$ s
Zeit in die Zukunft für lokale Maxima	$t(r_O)$	3.43	$[0, 0.5]$ s (*offline*), 0 s (*online*)
Zeit in die Vergangenheit für lokale Maxima	$t(l_O)$	3.43	$[0, 0.5]$ s
Mindestabstand zw. zwei Toneinsätzen	$t(min.dist)$	3.43	$[0, 0.05]$ s
Verschiebung der Toneinsätze	*onset.shift*	Abschn. 3.8	$[-0.01, 0.02]$ s

Beispiel 3.9.1 (F-Wert). *Hier wird das Beispiel 3.7.1 fortgesetzt, mit dem Ziel, F-Werte der offline und online Erkennung miteinander zu vergleichen. Als Schwellenwertfunktion wurde die Methode der gleitenden Quantile mit $p = 0.9$ verwendet. Bei der online Erkennung ist $t(r_T) = t(r_O) = 0$ s, während beim offline Ansatz diese Zeiten 0.15 s betragen. In den beiden Fällen sind $t(l_T) = t(l_O) = 0.15$ s. Das Toleranzintervall ist hier ± 50 ms, $t(min.dist) = 0.05$ s und $onset.shift = 0$ s. In Abbildung 3.5 sind die geschätzten Einsatzzeiten mit pinken Linien markiert und die zugehörigen F-Werte angegeben.*

Für die Klavieraufnahme kann mit den gewählten Parametern die perfekte Erkennung erreicht werden und zwar sowohl online also auch offline (fast perfekt). Die F-Werte für die Flötenaufnahme sind dagegen sehr klein: nur zwei der geschätzten Toneinsätze liegen innerhalb des Toleranzintervalls um die zugehörigen wahren Einsatzzeiten. Dabei würde man die Erkennungsgüte auf den ersten Blick eigentlich als nicht ganz schlecht beurteilen. Offensichtlich beeinflusst der Toleranzbereich den F-Wert sehr stark.

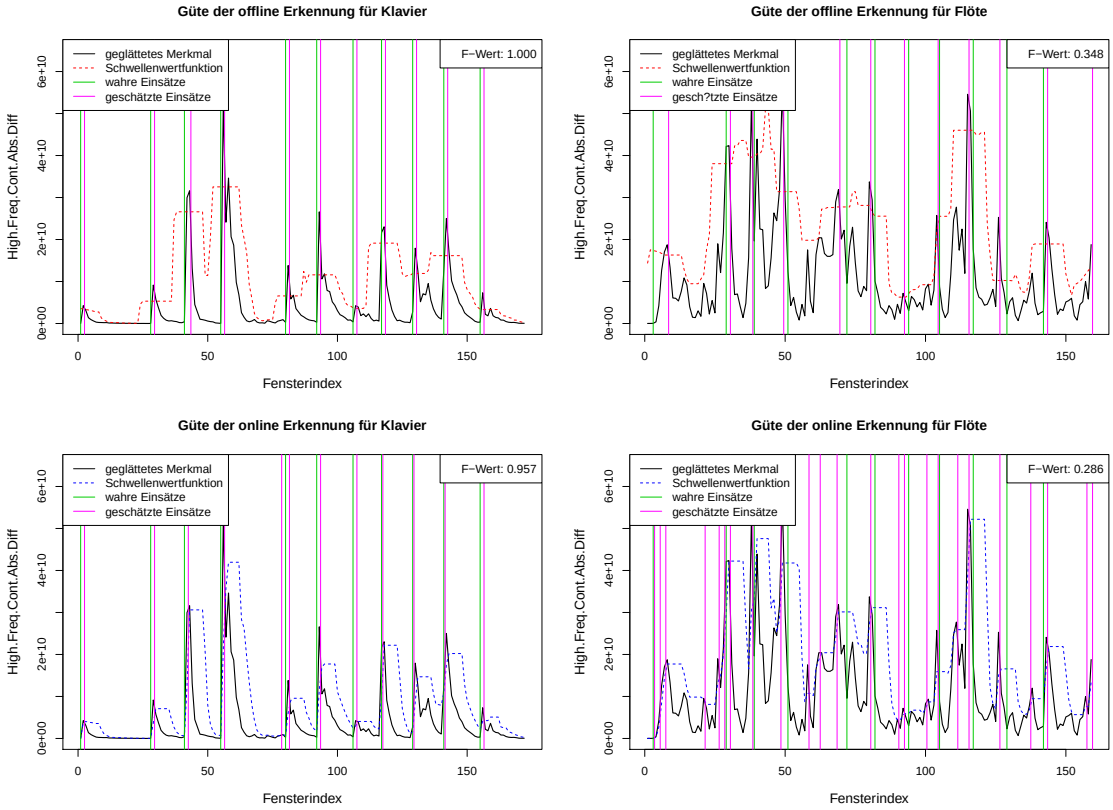

Abbildung 3.5: F-Wert am Beispiel des $High.Freq.Cont.Abs.Diff^{offline}$ Merkmals für Klavier (links) und Flöte (rechts). Die grünen vertikalen Linien markieren die wahren Toneinsatzzeiten, während die pinken vertikalen Linie die geschätzten Einsätze kennzeichnen.

3.9.2 D-Wert

Der F-Wert berücksichtigt den Abstand zwischen wahren und geschätzten Einsatzzeiten nur durch den Toleranzbereich. Ein häufiger Faktor, der zu einem niedrigen F-Wert führt, ist ein geschätzter Toneinsatz, der knapp außerhalb des Toleranzbereiches liegt, denn dadurch wird nicht nur die Anzahl der richtig erkannten Einsätze nicht erhöht, sondern auch ein falsch positiver Einsatz dazu gezählt.

Es bietet sich daher an, ein alternatives Gütekriterium vorzuschlagen und mit dem F-Wert zu vergleichen. Hier wird ein (mittlerer relativer) Abweichungswert (engl. *deviation*, abgekürzt mit D-Wert) vorgestellt. Die Grundidee der D-Wert-Berechnung ist in Algorithmus 3.2 dargestellt.

Algorithmus 3.2: Berechnung des D-Wertes.

input : $est_1, \ldots, est_n$: Zeiten der n geschätzten Toneinsätze in Sekunden
$true_1, \ldots, true_m$: Zeiten der m wahren Toneinsätze in Sekunden
$time.end$: Gesamtzeit der Aufnahme in Sekunden
output: D-Wert

1 Initialisierung: $Dev = 0$; $\mathscr{I} = \emptyset$
2 **for** $i = 1, \ldots, n$ **do**
3 | Finde Index j mit $j = min(|est_i - true_j|)$
4 | Merke Index j: $\mathscr{I} = (\mathscr{I} \cup j)$
5 | Erhöhe Abweichung: $Dev = Dev + |est_i - true_j|$
6 **end**
7 $\mathscr{I}_{false.neg} = \{1, \ldots, j\} \setminus \mathscr{I}$
8 **for** $k \in \mathscr{I}_{false.neg}$ **do**
9 | $time.right = true_k + (true_{k+1} - true_k)/2$ `// Falls` $k = m$, $time.right = time.end$
10 | $time.left = true_k - (true_k - true_{k-1})/2$ `// Falls` $k = 1$, $time.left = 0$
11 | Erhöhe Abweichung: $Dev = Dev + (time.right - time.left)/2$
12 **end**
13 D-Wert$= \dfrac{Dev}{m \cdot time.end}$

Zuerst wird jedem geschätzten Toneinsatz der zeitlich nächste wahre Toneinsatz zugeordnet und die absolute Abweichung zwischen den beiden notiert. Diese Abweichungen werden anschließend aufsummiert (Zeilen 2 bis 6). In Zeile 7 werden die fälschlicherweise nicht erkannten wahren Einsätze als solche Einsätze identifiziert, die keinem geschätzten Einsatz zugeordnet wurden. In Zeilen 8 bis 11 wird für jeden solchen „falsch negativen“ Einsatz folgendes gemacht: Sowohl rechts als auch links von ihm werden Zeitintervalle ermittelt, in denen ein Einsatz hätte geschätzt werden sollen, damit er diesem wahren Einsatz zugeordnet wäre. Dies ist offensichtlich die Hälfte der Zeit zwischen dem falsch negativen Einsatz und seinem rechten bzw. linken Nachbarn. Anschließend wird die halbe Zeitspanne zwischen der rechten und linken „Markierung“ als Abweichung berechnet (Zeile 11). Die so ermittelten Abweichungen werden ähnlich wie in Zeile 5 mit den bisher errechneten Werten aufsummiert. Der D-Wert wird dann in Zeile 13 „relativiert“, indem

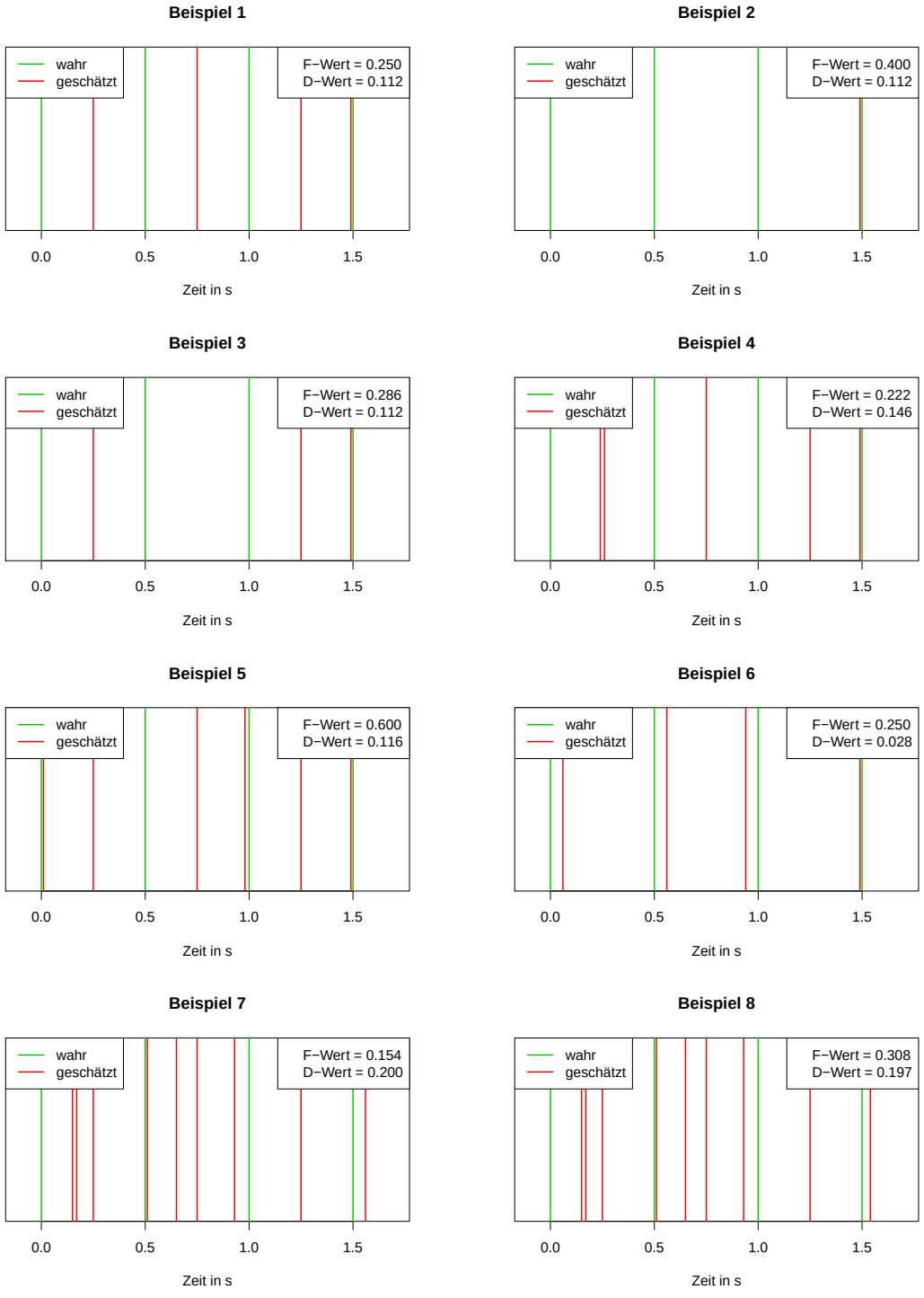

Abbildung 3.6: Gegenüberstellung des F- und D-Wertes anhand acht konstuierter Beispiele.

die kumulierte Abweichung durch die Dauer der Aufnahme (in Sekunden) und durch die Anzahl wahrer Einsätze geteilt wird. Zu bemerken ist, dass im Unterschied zum F-Wert der D-Wert minimiert werden soll. Werden alle Einsätze ohne Zeitverzögerung erkannt (was praktisch unmöglich ist), ist der D-Wert gleich 0. Werte größer als 1 sind zwar möglich, aber sehr unwahrscheinlich.

Die beiden Werte haben Vor- und Nachteile, die anhand acht konstruierter Beispiele aus Abbildung 3.6 exemplarisch veranschaulicht werden. Bei allen Beispielen handelt es sich um ein 1.7 Sekunden langes Stück mit vier wahren Einsatzzeiten: bei 0, 0.5, 1 und 1.5 Sekunden. Der Toleranzbereich für den F-Wert beträgt hier ± 50 ms. Der D-Wert ist so konstruiert, dass ein Einsatz, der genau in der Mitte zwischen zwei wahren Einsätzen geschätzt wird, genauso bestraft wird, wie ein fehlender Einsatz. Aus diesem Grund ist der D-Wert in Beispielen 1-3 gleich, während der F-Wert von den falsch positiven Einsätzen negativ beeinflusst wird (bemerkenswert ist dabei der F-Wert von 0.4 im zweiten Beispiel).

Beispiel 5 verdeutlicht den Nachteil des D-Wertes: zusätzlich zu der Situation in Beispiel 1 sind zwei richtige Schätzungen des ersten und des dritten Tones hinzugekommen. Die zusätzlichen Einsätze verursachen aber mehr Abweichung, so dass der D-Wert sogar leicht ansteigt. Der F-Wert reagiert auf die korrekten Schätzungen positiv und steigt auf seinen besten Wert an. Der beste D-Wert wird dagegen in Beispiel 6 erreicht: hier werden die ersten drei Einsätze mit der Verzögerung von 60 ms erkannt (knapp außerhalb des Toleranzbereichs). Für den F-Wert gibt es hier keinen Unterschied zu der Situation in Beispiel 1, obwohl die Erkennung hier als deutlich besser bezeichnet werden kann.

Die letzten beiden Beispiele 7 und 8 veranschaulichen die „Empfindlichkeit“ des F-Wertes auf eine kleine Veränderung und zwar liegt der letzte geschätzte Einsatz in Beispiel 8 innerhalb des Toleranzbereichs, während er diesen in Beispiel 7 knapp verfehlt. Der F-Wert verdoppelt sich allerdings daraufhin, während der D-Wert sich nur leicht verbessert.

Beispiel 3.9.2 (Korrelation zwischen den F- und D-Werten). *Für eine weitere Analyse der Korrelation zwischen den F- und D-Werten wurden erneut die beiden Halleluja-Stücke betrachtet. Sowohl für das Klavier als auch für die Flöte wurden alle 18 EZE-Merkmale aus Tabelle 3.2 berechnet (mit den in Beispiel 3.9.1 erwähnten Einstellungen für die online Erkennung). In Abbildung 3.7 ist das zugehörige Streudiagramm dargestellt. Da man Punktecluster erkennen kann, wurde zusätzlich deren Zugehörigkeit zu den vier Merkmalsgruppen (Abschnitt 3.5) notiert. Allerdings konnte kein Zusammenhang zwischen Clustern und Merkmalsgruppen erkannt werden. Es ist ein negativer Zusammenhang zwischen F- und D-Wert zu erkennen. Für das Klavier beträgt der Korrelationskoeffizient nach Pearson* -0.82 *und für die Flöte* -0.73.

Im Weiteren werden zwei besonders auffallende Einzelfälle für das Klavier analysiert. Im ersten Fall handelt es sich um die Merkmale Ampl.Max.Diff und Norm.Weight.Phase.Dev, die etwa den gleichen F-Wert allerdings sehr unterschiedliche D-Werte aufweisen. Abbildung 3.8 (oben) stellt die beiden EZE-Funktionen dar. Die geschätzten Einsätze in der rechten Abbildung scheinen günstiger zu liegen als in der linken. Somit würde hier die Betrachtung des D-Wertes helfen,

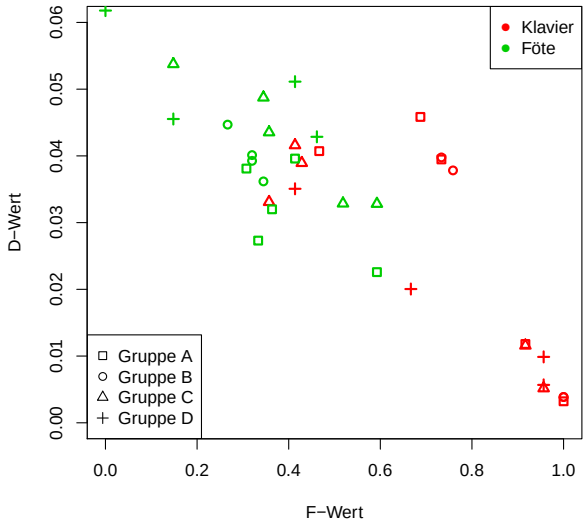

Abbildung 3.7: Zusammenhang zwischen den F- und D-Werten für Klavier- und Flötenaufnamen des Halleluja-Liedes. Pro Aufnahme wurden alle in Tabelle 3.2 erwähnten EZE-Merkmale angewendet. Die Zugehörigkeit der Merkmale zu den in Abschnitt 3.5 definierten Gruppen A-D ist durch verschiedene Symbole markiert.

zwei ähnlich gute Erkennungen (laut dem F-Wert) zu differenzieren. Im zweiten Fall, umgekehrt, liefern die Merkmale Ampl.Energy.Diff und Spec.Spread.Diff etwa den gleichen D-Wert bei unterschiedlichen F-Werten (s. Abbildung 3.8 unten). Hier ist es eindeutig, dass das linke Bild mit dem höheren F-Wert eine bessere Erkennung darstellt als das rechte.

Es stellt sich nun die Frage wie der F-Wert mit dem D-Wert verknüpft werden kann. Es handelt sich dabei um zwei verschiedene Skalen, so dass diese Verknüpfung nicht unproblematisch ist. Außerdem gibt es keine klare obere Grenze für den D-Wert und es ist unklar, wie sehr er von dem Tempo und der Dauer einer Musikaufnahme beeinflusst wird. Der optimale Weg wäre hier die zwei-kriterielle Optimierung. Aus Komplexitätsgründen wird hier allerdings ein anderer Weg gewählt, um eine Empfehlung bezüglich der optimalen Einstellungen des EZE-Algorithmus zu geben. Es wird nämlich nach F-Wert optimiert und anschließend wird aus der Menge der besten Lösungen (die fast identische F-Werte aufweisen können), diejenige ausgewählt, die den kleinsten D-Wert hat.

Die durchgeführte Analyse des in der Literatur etablierten F-Wertes hat gezeigt, dass die Aufstellung eines alternativen Gütekriteriums notwendig ist. Es wurde zwar der abweichungsbasierte D-Wert vorgeschlagen, welcher allerdings in seiner jetzigen Definition noch nicht als ein eigenständiges Gütemaß betrachtet werden kann. Eine weiterführende Erforschung dieses Problems scheint daher sinnvoll und relevant zu sein (s. Kapitel 9).

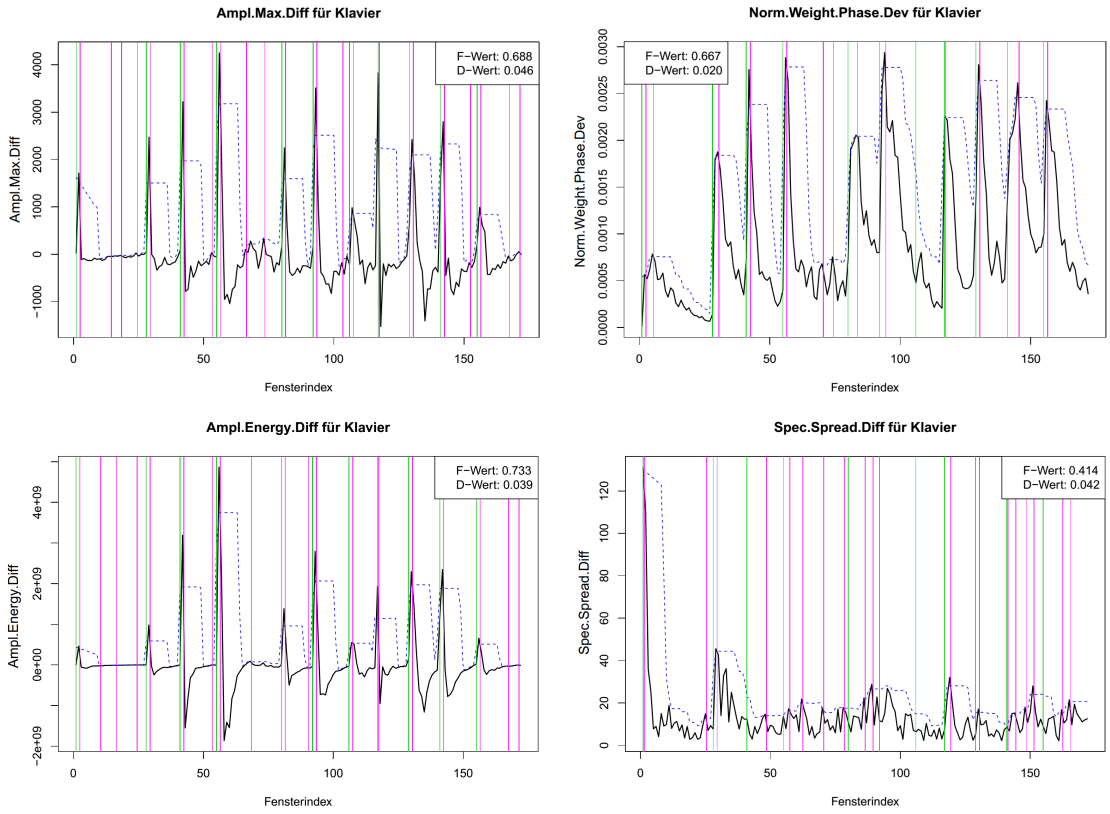

Abbildung 3.8: Veranschaulichung der Extremfälle für Klavieraufnahmen. Oben: ähnliche F-Werte bei stark unterschiedlichen D-Werten. Unten: ähnliche D-Werte bei stark unterschiedlichen F-Werten.

Kapitel 4

Multivariate Einsatzzeiterkennung

Im vorherigen Kapitel wurde das klassische Schema der Einsatzzeiterkennung beleuchtet. Sie kann auch als univariate EZE bezeichnet werden, da immer nur ein EZE-Merkmal berücksichtigt wird. Da bei der EZE zwei binäre Vektoren (der geschätzten bzw. der wahren Ereignisse in den jeweiligen Fenstern) vorliegen, ist es naheliegend, Klassifikationsverfahren für diese Aufgabe zu verwenden. Dabei kann man nicht nur ein, sondern alle EZE-Merkmale einbeziehen.

Allerdings ist die Anwendung der Klassifikationsverfahren nicht unproblematisch. Zum einen kann die Güte der Klassifikation nicht anhand der Konfusionsmatrix beurteilt werden. Denn nicht die direkten Übereinstimmungen sind entscheidend, sondern die Tatsache, ob der geschätzte Toneinsatz im Toleranzbereich des wahren Einsatzes liegt, auch wenn er unter Umständen mehrere Fenster später oder früher erkannt wird (s. Abschnitt 3.9.1). Zum anderen hängt die gelernte Entscheidungsregel von den Parametern der Einsatzzeiterkennung (wie Fensterlänge oder Sprungweite) ab, so dass man nicht auf einer EZE-Einstellung lernen und auf einer anderen vorhersagen kann.

In Algorithmus 4.1 ist ein Vorschlag für die multivariate EZE vorgestellt. In nachfolgenden Abschnitten werden alle Details der einzelnen Algorithmus-Schritte näher diskutiert.

Algorithmus 4.1: Multivariate Einsatzzeiterkennung.

1 Teile die Musikdatenbank in Lern- und Evaluierungsdatenbank auf;
2 Bestimme für alle Stücke der Lerndatenbank die EZE-Matrizen und füge sie zu einer Matrix D_{lern} zeilenweise zusammen;
3 Erweitere D_{lern} mit dem (0,1)-Vektor der wahren Toneinsätze: $\tilde{D}_{lern}$;
4 Selektiere die wichtigsten Merkmale mittels der Vorwärts-Variablenselektion;
5 Passe das Klassifikationsmodell M_{class} auf den selektierten Merkmalen $\tilde{D}^{sel}_{lern}$ an;
6 **for** *Musikstücke in der Evaluierungsdatenbank* **do**
7 Bilde die EZE-Matrix der selektierten Merkmale für das Musikstück;
8 Sage die Wahrscheinlichkeiten der Toneinsätze *prob.onset* mit Hilfe von M_{class} vorher;
9 Bestimme den Vektor der geschätzten Toneinsätze mittels *prob.onset*;
10 Berechne den F-Wert der Erkennung;
11 **end**
12 Mittlere die F-Werte über alle Stücke der Evaluierungsdatenbank.

4.1 Aufteilung der Datenbank

In Zeile 1 von Algorithmus 4.1 soll die Musikdatenbank in zwei Teile aufgeteilt werden: Lern- bzw. Evaluierungsdatenbank zum Lernen bzw. Evaluieren der Klassifikationsregeln. Dabei stellt sich die Frage nach der richtigen Proportion zwischen den beiden Datenbanken. Einerseits bewirkt eine große Lerndatenbank ein allgemeineres Klassifikationsmodell, andererseits kann die Modellanpassungszeit je nach Verfahren exponentiell steigen. Für diese Arbeit wird ein Anteil von 20% für die Lerndatenbank festgelegt.

4.2 Bestimmung der EZE-Matrizen

Pro Signalfenster können – wie bereits in Abschnitt 3.5 besprochen – 18 Merkmale bestimmt werden. In einem einfachen Fall würden die Zeilen der EZE-Matrix eines Musikstückes die Signalfenster und die Spalten die 18 Merkmale darstellen. Da die Anzahl der Signalfenster von der Länge der Musikaufnahme, Fensterlänge N und Sprungweite h abhängt, kann die Dimension der EZE-Matrix stark variieren.

Nicht nur die Information aus dem aktuellen Signalfenster, sondern auch die aus den vorherigen bzw. zukünftigen Fenstern kann für die Klassifikation berücksichtigt werden. Somit können nicht nur die Fenster-Überlappungseffekte, sondern auch die zeitliche Entwicklung eines Toneinsatzes berücksichtigt werden. Es sind dabei zwei Aspekte zu beachten. Zum einen bedeutet die Betrachtung jedes zusätzlichen Fensters die Erweiterung der EZE-Matrix um 18 Spalten. Zum anderen ist eine pauschale Angabe über die Anzahl solcher zusätzlicher Fenster nicht sinnvoll, da je nach N und h sehr unterschiedliche Zeiträume dadurch erfasst werden. Aus diesem Grund wird die Anzahl der zu berücksichtigenden Fenster rechts und links des aktuellen Fensters bestimmt. Diese Anzahlen l_M und r_M hängen – ähnlich wie in Formel 3.44 aus Abschnitt 3.8 – von N, h, $t(l_M)$ und $t(r_M)$ ab. Dabei beschreibt $t(l_M)$ bzw. $t(r_M)$ die maximal erlaubte Zeit in die Vergangenheit bzw. die Zukunft. Allerdings dürfen diese die Anzahl von jeweils drei Fenstern nicht überschreiten, um die Dimension der EZE-Matrix (und der damit verbundenen Lernzeit der Klassifikatoren) nicht erheblich zu erhöhen:

$$\begin{aligned} r_M &= \min(n_{Fenster}(t(r_M), N, h), 3), \\ l_M &= \min(n_{Fenster}(t(l_M), N, h), 3). \end{aligned} \tag{4.1}$$

Außerdem soll hier weiterhin zwischen der *online* und *offline* Erkennung unterschieden werden, so dass in *online* Fall $t(r_M) = 0$ s und $t(l_M) = 0.15$ s betragen, während bei *offline* EZE $t(r_M) = 0.15$ s und $t(l_M) = 0.15$ s gilt. Diese Zeitschranken wurden hier festgelegt, um sie nicht als zusätzliche Parameter in die Optimierung aufzunehmen.

Die Matrix D_{lern} (Zeile 2 von Algorithmus 4.1) ergibt sich aus der zeilenweisen Verknüpfung aller einzelnen EZE-Matrizen der Lerndatenbank. Aus der nachfolgenden Analyse hat sich

ergeben, dass die Anzahl der Zeilen dieser Matrix im fünfstelligen Bereich variiert, während die Anzahl der Spalten meistens zwischen 55 (3 Fenster) und 126 (7 Fenster) liegt.

Die dritte Zeile des Algorithmus ist selbsterklärend: die Matrix D_{lern} wird um eine zusätzliche (0,1)-Spalte erweitert, die die Information darüber beinhaltet, ob in den jeweiligen Fenstern (Zeilen der Matrix) ein bzw. kein Toneinsatz stattfindet. Dabei steht '1' für einen Toneinsatz.

4.3 Variablenselektion

Variablenselektion ist ein wichtiger und gleichzeitig auch ein sehr zeitintensiver Schritt. Durch die Variablenselektion werden die wichtigsten EZE-Merkmale ermittelt, so dass in der Testphase keine unnötigen Merkmale mehr erhoben werden müssen. Dies ist besonders für die *online* Erkennung relevant. Die Variablenselektion wird naturgemäß für dasjenige Lernverfahren angewendet, welches in dem darauffolgenden Schritt für die Anpassung des endgültigen Modells M_{class} verwendet wird. Die Lernverfahren und ihre Parametereinstellungen werden in Abschnitt 4.4 diskutiert.

Hier wird die Vorwärts-Selektion bezüglich des F_{class}-Wert durchgeführt. Dabei bezeichnet F_{class} den F-Wert bei den Klassifikationsproblemen. Im Unterschied zu dem F-Wert der EZE (wie in Abschnitt 3.9.1 definiert) werden hier keine Toleranzbereiche berücksichtigt, sondern die aus der Kontingenztafel der Klassifikation resultierenden Häufigkeiten verwendet (klassische Definition, s. Tabelle 3.1). Eine zusätzliche Variable wird in einem Selektionsschritt erst dann aufgenommen, wenn sie eine Mindestverbesserung von 0.01 zu dem bereits erreichten F_{class}-Wert beitragen kann. Das Ergebnis der Selektion ist die Menge der selektierten Merkmale (meist 4 bis 8 Merkmale) und die dazugehörige reduzierte Lernmatrix $\tilde{D}^{sel}_{lern}$.

Die Vorwärts-Selektion wird mit Hilfe der *selectFeatures* Funktion aus dem R-Paket **mlr** (Bischl et al., 2015c) realisiert. Da die Trainingszeit des Klassifikationsmodell je nach verwendetem Verfahren sehr stark von der Anzahl der Beobachtungen anhängt, wird die Matrix $\tilde{D}_{lern}$, falls notwendig, gekürzt: Übersteigt die Anzahl der Zeilen die Zahl 20 000, so werden zufällig 20 000 Zeilen aus $\tilde{D}_{lern}$ ausgewählt und als Grundlage für die Variablenselektion verwendet.

Des Weiteren wird die Holdout-Methode (s. Abschnitt 6.4.1) verwendet, um die Güte des auf den selektierten Variablen angepassten Modells in jedem Selektionsschritt zu beurteilen. Dabei werden 50% der Daten für das Modelltraining und 50% der Daten für die Güteberechnung verwendet. Das Modell wird also in jedem Selektionsschritt auf maximal 10 000 Beobachtungen angepasst. Nichtsdestotrotz kann die Variablenselektionsphase, je nach Klassifikationsverfahren, mehrere Stunden in Anspruch nehmen (auf dem in Abschnitt 6.4.3 erwähnten Computercluster).

Anzumerken ist, dass hier ein unbalanciertes Klassenproblem vorliegt. Es gibt hierbei viel mehr Fenster, in denen kein Toneinsatz vorkommt, als diejenigen, die einen Toneinsatz enthalten. Somit ist die sogenannte Stratifizierung sinnvoll: eine bezüglich der Zielvariable möglichst gleichmäßige Auswahl der Beobachtung für die Modellanpassung (d.h. Ausbalancieren der Daten). In **mlr** R-Paket wird die Stratifizierung durch Einstellung des Parameters `stratify = TRUE` der *makeResampleDesc* Funktion realisiert.

4.4 Anpassung des Klassifikationsmodells

In diesem Schritt wird das endgültige Klassifikationsmodell M_{class} auf $\tilde{D}^{sel}_{lern}$ angepasst. Zu bemerken ist, dass es sich hierbei bezüglich der Anzahl der Zeilen um die komplette Lernmatrix handelt, auch wenn sie zum Zweck der Selektion reduziert werden musste. Dieses Modell wird für den nachfolgenden Evaluierungsschritt verwendet. In der Validierungsphase (s. Abschnitt 6.4.2) soll kein neues Modell mehr angepasst werden, sondern es wird ein zuvor gespeichertes Modell aus der Optimierungsphase verwendet[1].

In dieser Arbeit werden drei Klassifikationsmodelle berücksichtigt: logistische Regression (aus dem R-Paket **stats**, R Core Team (2014), abgekürzt mit *logReg*), Zufallswald (engl. *random forest*, aus dem R-Paket **randomForest**, Liaw und Wiener (2002), abgekürzt mit *randForest*) und Stützvektormaschine (engl. *support vector machine*, aus dem R-Paket **e1071**, Meyer et al. (2015), abgekürzt mit *svm*). Die Erklärung der genannten statistischen Lernverfahren findet sich z.B. in Hastie et al. (2001). Das Klassifikationsmodell ist allerdings kein zusätzlicher Parameter für die anschließende Optimierung, was theoretisch auch denkbar und sinnvoll wäre. Vielmehr wird je Klassifikationsmodell eine eigene Optimierung durchgeführt.

Da sowohl das *randForest* als auch das *svm* Verfahren sehr stark von den internen Einstellungen – den sogenannten Hyperparametern – abhängen, ist ein vorheriges Tuning (bzw. Optimieren) dieser Einstellungen notwendig. Die modellbasierte Optimierung der Hyperparameter wurde mit Hilfe des **mlrMBO**-Pakets (Bischl et al., 2015a) durchgeführt, wobei sowohl die Optimierungsmethode als auch die ermittelten optimalen Einstellungen der Hyperparameter für die beiden Klassifikatoren nachfolgend zusammengefasst werden.

Die Optimierung der Hyperparameter basiert auf einem Datensatz von 30 Musikstücken[2]: 15 aus der MIDI- und 15 aus der WAV-Datenbank (s. Kapitel 5). Für jedes Stück wird die EZE-Matrix (Abschnitt 4.2) mit den folgenden Einstellungen bestimmt: $N = 2048$, $h = 1500$, $window.fun =$ „Hanning“, $spec.filter =$ „Nein“ und $spec.log =$ „Nein“. In jedem Signalfenster werden die Merkmale des aktuellen, der drei vorherigen und eines zukünftigen Fensters bestimmt. Die EZE-Matrizen der 30 Stücke werden anschließend zeilenweise zu einer *TRAIN*-Matrix zusammengefügt. Die *TRAIN*-Matrix besteht aus 36 226 Zeilen und 91 Spalten (5 Fenster je 18 EZE-Merkmale, s. Abschnitt 3.5 und die letzte Spalte ist der (0,1)-Vektor der wahren Toneinsätze), wobei ein Toneinsatz in ca. 20% der Zeilen vorkommt.

[1] Um den Speicherplatzbedarf zu reduzieren, wird pro Optimierungslauf nur das Klassifikationsmodell der aktuell besten Parameterkombination gespeichert.

[2] Die Datenbank befindet sich auf einem internen Lehrstuhl-Server.

Da auf der *TRAIN*-Matrix während der Optimierung nicht nur einmal, sondern 90 mal (entsprechend der Größe des Startdesigns und der Anzahl der sequentiellen MBO-Schritte, s. unten) das jeweilige Klassikationsverfahren gelernt werden sollte und bei manchen Hyperparameter-Kombinationen auch nach mehreren Tagen auf den vollen Daten keine Modellanpassung erfolgte, musste die *TRAIN*-Matrix gekürzt werden. Dabei hat sich die zufällige Auswahl von 7000 Zeilen als ein Kompromiss zwischen der Modellgüte und einer angemessenen Modellierungszeit erwiesen. Zu bemerken ist, dass es sich bei der *TRAIN*-Matrix nicht um die D_{lern} Matrix aus Algorithmus 4.1 handelt. Die *TRAIN*-Matrix wird nur einmal für die oben erwähnten Parametereinstellungen gebildet und dient lediglich dem Hyperparametertuning. Da dieses Tuning nicht im Vordergrund des multivariaten Verfahrens steht, werden hier für die Modellanpassung weniger Beobachtungen (7000) im Vergleich zu dem Modellselektionsschritt aus vorigem Abschnitt (10000) herangezogen.

Die wichtigsten Einstellungen der MBO-Optimierung (werden in Abschnitt 6.2 eingeführt) für das Hyperparametertraining lauten:

- Größe des Startdesigns: 50,
- Anzahl der sequenziellen Schritte: 40,
- Surrogatmodell: Kriging mit Matérn-Kernel ($\nu = 3/2$),
- Zielkriterium: erwartete Verbesserung (EI),
- Optimierung des Zielkriteriums: fokussierte Suche.

Das Gütekriterium der MBO-Optimierung ist der F_{class}-Wert (wie in Abschnitt 4.3 definiert). Aus Gründen der Einfachheit wurde hier die Größe des Startdesigns und die Anzahl der sequenziellen Schritte für alle Optimierungsprobleme gleich gesetzt, obwohl sie eine verschiedene Anzahl an zu optimierenden Parameter haben (zwischen 2 und 4, s. unten)

Ein wichtiger Hyperparameter von SVM-Verfahren ist der Kernel. Folgende Kernel-Funktionen werden hier verwendet, wobei u,und v zwei Vektoren sind (Fan et al., 2005):

- *linear*: $K(u,v) = u^T \cdot v$,
- *polynomiell*: $K(u,v) = (\gamma \cdot u^T \cdot v + C)^g$,
- *radial*: $K(u,v) = \exp(-\gamma \cdot ||u-v||^2)$,
- *sigmoid*: $K(u,v) = \tanh(\gamma \cdot u^T \cdot v + C)$.

Da je nach Kernel verschiedene weitere Hyperparameter angepasst werden sollten, wurde je Kernel eine eigene Optimierung durchgeführt. Unabhängig von dem Kernel wird der Parameter ν auf dem zulässigen Intervall $[0,1]$ optimiert. Dieser Parameter bestimmt die obere Schranke des Anteils an fehlklassifizierten Beobachtungen und – gleichzeitig – die untere Schranke des Anteils an Beobachtungen, die als Stützvektoren gewählt werden. Weitere Details der ν-SVM sind in Chang und Lin (2002) ausführlich erläutert.

Bei dem *linearen* Kernel soll somit lediglich der Parameter ν optimiert werden. Aus diesem Grund wurde hier keine MBO-Optimierung, sondern eine Gittersuche mit der Schrittweite 0.1 durchgeführt. Für die restlichen Kernel-Parameter wurden folgende zulässige Intervalle berücksichtigt: Werte 1 bis 5 für g, $[0,500]$ für C und $(0,5]$ für γ. Zu bemerken ist, dass es theoretisch keine oberen Schranken für die Parameter g, C und γ existieren. Sie werden hier somit willkürlich festgesetzt. Insbesondere ist die Wahl einer sinnvollen oberen Schranke für den Parameter γ problematisch. Da allerdings als Standardeinstellung in R-Paket **e1071** ein sehr kleiner Wert verwendet wird (1 geteilt durch Anzahl der Zeilen der Datenmatrix), erscheint der Wert 5 als die obere Schranke ausreichend groß zu sein. Als optimal wird anschließend diejenige Hyperparametereinstellung gesehen, die bei der gemeinsamen Betrachtung der Ergebnisse aller Kernels den besten F_{class}-Wert erzielt hat.

Bei der Optimierung des Zufallswaldes kommen folgende Hyperparameter in Frage: n_{Baum} (Anzahl von Bäumen) auf dem zulässigen Intervall $[100,900]$, n_{Blatt} (minimale Größe der Blattknoten) auf dem zulässigen Intervall $[1,10]$ und $n_{Kandidat}$ (Anzahl der Variablen, die als Kandidaten in jedem Schritt zufällig ausgewählt werden) auf dem zulässigen Intervall $[5,91]$.

Die optimalen Einstellungen für die beiden Klassifikationsverfahren (laut der durchgeführten MBO) sind gegeben durch:

- *svm*: $\nu = 0.56$ und *polynomieller* Kernel mit $d = 1$, $C = 57$ und $\gamma = 5$[3];
- *randForest*: $n_{Baum} = 174$, $n_{Blatt} = 9$ und $n_{Kandidat} = 27$.

Zu bemerken ist, dass es sich hierbei um eine einfache Auslegung der Hyperparameteroptimierung handelt, die auch kritisch gesehen werden kann. Sowohl die Zusammensetzung der Trainingsdatenbank, die Parameter der EZE-Matrix-Berechnung, die Größe der endgültigen *TRAIN*-Matrix als auch die zulässigen Bereiche der einzelnen Hyperparameter könnten anders gewählt werden. Die Hyperparameteroptimierung stellt allerdings nicht den zentralen Schwerpunkt dieser Arbeit dar. Sie sollte hier lediglich dazu dienen, ein sinnvolleres Hyperparameterset zu finden als die Standardeinstellung.

4.5 Ermittlung des mittleren F-Wertes

In Zeilen 6 bis 11 des Algorithmus 4.1 wird für jedes Musikstück die Güte der Einsatzzeiterkennung bestimmt und anschließen in Zeile 12 gemittelt. Der gemittelte F-Wert gilt – genauso wie bei der univariaten EZE – als das zu optimierende Gütemaß, wobei auch der D-Wert für die weitere Analyse zusätzlich notiert wird. Dabei wird zuerst für jedes Stück die EZE-Matrix (lediglich für die selektierten Merkmale) bestimmt (Zeile 7), auf welche dann das in der Lernphase trainierte Modell M_{class} angewendet wird. An dieser Stelle sind nicht die zugewiesenen Klassen (Einsatz /

[3] Da der optimale Wert von dem Parameter γ an der Grenze des zulässigen Bereichs liegt, spricht es dafür, dass die Begrenzung nach oben doch zu klein gewählt wurde. Da die Hyperparameteroptimierung für das *svm* Verfahren sehr zeitintensiv ist, wurde sie nicht erneut mit dem größeren zulässigen Bereich für γ durchgeführt.

kein Einsatz) von Interesse, sondern die von dem Klassifikator ermittelte Wahrscheinlichkeit für das Eintreten des positiven Ereignisses (Einsatz): *prob.onset*. Der Vektor *prob.onset* wird dann als das univariate EZE-Merkmal weiter behandelt. Es wird hierbei die Formel (3.43) (Abschnitt 3.8) angewendet, wobei anstelle von *st.odf* der Vektor der geschätzten Wahrscheinlichkeiten und anstelle von T_n ein fester Schwellenwert für die positive Entscheidung verwendet wird:

$$O_n = \begin{cases} 1, & \text{falls } prob.onset_n > classif.thresh \text{ und} \\ & prob.onset_i = \max(prob.onset_{n-l_O}, \ldots, prob.onset_{n+r_O}) \text{ und} \\ & n > n_{last.onset} + min.dist, \\ 0, & \text{sonst.} \end{cases} \tag{4.2}$$

Die Innovation bzw. die Hauptidee der multivariaten EZE liegt in der Verwendung der obigen Formel. Als erstes soll der *classif.thresh* motiviert werden. Standardmäßig liegt die Entscheidungsgrenze für eine positive Klassenzuweisung bei 0.5. Beim Trainieren von unbalancierten Problemen empfehlen Voigt et al. (2014) allerdings eine andere Grenze *classif.thresh* anzupassen. In dieser Arbeit wird diese Idee für alle Klassifikationsverfahren umgesetzt, wobei *classif.thresh* auf dem Intervall $[0.05, 0.95]$ optimiert wird. Hier wird lediglich ein fester Schwellenwert verwendet, wobei auch die gleitenden Schwellenwerte, die in Abschnitt 3.7 diskutiert wurden, sich dafür eignen würde. Da *prob.onset* allerdings im Intervall $[0,1]$ liegt (also standardisiert ist) und es sich um die Wahrscheinlichkeit handelt, ist die Verwendung des festen Schwellenwertes naheliegend.

Die Parameter r_O, l_O und *min.dist* werden mit den gleichen zulässigen Bereichen für *online* bzw. *offline* Erkennung wie in dem univariaten Fall optimiert. Ausgehend von dem Vektor $\boldsymbol{O} = (O_1, \ldots, O_M)$ werden die Zeiten der geschätzten Toneinsätze wie in Abschnitt 3.8 sowie die zugehörigen F- und D-Werte bestimmt (Zeile 10).

Die multivariate EZE hat ein kleineres Set an einstellbaren Parametern als die univariate. Nicht berücksichtigt werden hier das EZE-Merkmal und die Parameter der Schwellenwertfunktion. Auch die exponentielle Glättung wird nicht angewendet, da die stark verrauschte „Zick-Zack“-Struktur der EZE-Merkmale für die Klassifikatoren kein Problem darstellen sollte und somit keinen Grund mehr besteht, die Originalwerte zu verändern. Das adaptive Weißen wird mit der gleichen Begründung wie im univariaten Fall nicht berücksichtigt (s. Abschnitt 3.4). Auch auf die Aufnahme des Parameters *onset.shift* für die Optimierung wird verzichtet. Zum einen ist dies damit begründet, dass nach der Auswertung der optimalen Einstellungen im univariaten Fall (s. Abschnitt 8.1.6) dieser Parameter eher in der Nähe der Null gewählt werden sollte (d.h. keine Verschiebung). Zum anderen reduziert sich durch die Nichtbetrachtung dieses Parameters die Optimierungszeit.

Die Zusammenfassung der zu optimierenden Parameter der multivariaten EZE mit den entsprechenden zulässigen Intervallen bzw. Ausprägungen ist in Tabelle 4.1 vorgestellt.

Tabelle 4.1: Zusammenfassung der Parameter der multivariaten Einsatzzeiterkennung

Parameter	*Bezeichnung*	*Formel/Abschn.*	*Ausprägungen bzw. Intervall*
Fensterlänge	N	Abschn. 3.3	$512, 1024, 2048, 4096$
Sprungweite	h	Abschn. 3.3	$[N/10, N]$
Fensterfunktion	*window.fun*	2.19 2.20 2.21 2.22	*Rectangular* *Hanning* *Blackman* *Gauss*
Filterung des Spektrums	*spec.filter*	3.3	Ja, Nein
Logarithmierung des Spektrums	*spec.log*	3.4	Ja, Nein
Komprimierungsparameter der Logarithmierung	ℓ	3.4	$[0.01, 20]$
Zeit in die Zukunft für lokale Maxima	$t(r_O)$	4.2	$[0, 0.5]$ s (*offline*), 0 s (*online*)
Zeit in die Vergangenheit für lokale Maxima	$t(l_O)$	4.2	$[0, 0.5]$ s
Mindestabstand zwischen zwei Toneinsätzen	$t(min.dist)$	4.2	$[0, 0.05]$ s
Wahrscheinlichkeits-Schwellenwert	*classif.thresh*	4.2	$[0.05, 0.95]$

Auf der beiliegenden CD ist der R-Code (inklusive Beispieldateien) sowohl für die univaraite (im Ordner `UnivariateOnsetDetection`) als auch für die multivariate (im Ordner `Multivariate-OnsetDetection`) EZE zu Verfügung gestellt.

4.6 Unterschied zu anderen Vorgehen

Auch Lacoste und Eck (2007) wenden die überwachte Klassifikation auf die Einsatzzeiterkennung an. Allerdings lassen sich relevante Unterschiede zu dem hier vorgeschlagenen Verfahren erkennen. Lacoste und Eck (2007) klassifizieren direkt auf den Fourierkoeffizienten, ohne zuvor bestimmte Merkmale aus den FKen zu extrahieren. Pro Fenster ergeben sich über 1 000 FKen, wobei die Autoren dann lediglich ca. 200 FKen zufällig auswählen, um den verwendeten Klassifikator (neuronale Netze) im *online* Fall schnell trainieren und anwenden zu können. Weiterhin wird keine Überlappung erlaubt sowie der F-Wert direkt aus der Kontingenztafel der Klassifikation ausgerechnet, ohne die zeitliche Toleranz zu berücksichtigen (F_{class}, s. Abschnitt 4.3). Dies entspricht einer einfachen Auslegung des Problems, die man als „den ersten Schritt“ bezeichnen könnte. Die Autoren zeigen, dass das vorgeschlagene Verfahren sehr gut im Vergleich zu anderen „State of the Art“ Algorithmen abschneidet, wobei der Vergleich auf der MIREX 2005 Audio Onset Detection Datenbank[4] vollzogen wurde.

[4] `http://www.music-ir.org/mirex/wiki/2005:Audio_Onset_Detect`, Stand: 01.05.2015.

Roebel (2005) verwendet Klassifikation in einem etwas anderen Kontext, wobei der Autor einen *online* fähigen EZE-Algorithmus vorschlägt, bei dem die Spitzen des EZE-Merkmals zu transienten und nicht-transienten Spitzen klassifiziert werden. Dieses Vorgehen (in verschiedenen Modifikationen) hat beachtliche Ergebnisse bei verschiedenen MIREX Wettbewerben erzielt. Die Hauptidee basiert auf der Tatsache, dass der Anschlag jedes Musiktones ein transientes Signal beinhaltet (Abschnitt 2.3) und die Spitzen des EZE-Merkmals, die durch das transiente Signal hervorgerufen sind, sich von denjenigen unterscheiden sollen, die durch zufällige Effekte wie Rauschen entstehen. In jedem Fenster wird zunächst das Merkmal ‘spektraler Schwerpunkt’ (Abschnitt 3.5.3) berechnet. Anschließend wird mittels eines vorher festgelegten Schwellenwertes klassifiziert, ob der Merkmalswert eher für eine transiente Spitze oder für ein zufälliges Rauschen spricht. Lediglich die ersten Fälle werden als Kandidaten für einen Toneinsatz in den nachfolgenden Schritten berücksichtigt.

Obwohl in Roebel (2005) das Wort Klassifikation für die Vorselektion der Merkmalsspitzen verwendet wird, handelt es sich hier offensichtlich nicht um die Anwendung der klassischen Verfahren zur überwachten bzw. unüberwachten Klassifikation. Daher kann diese Veröffentlichung nicht als Vergleich zu der in dieser Arbeit beabsichtigten Anwendung von Klassifikationsverfahren für EZE dienen.

Kapitel 5

Musikdatenbank

In diesem Kapitel wird die Datensatzkonstruktion diskutiert. Da die Optimierungsergebnisse von den zugrunde liegenden Daten stark abhängen, wird eine ausreichend große und heterogene Trainingsdatenbank angestrebt. Dabei sollen z.B. verschiedene Musikinstrumente (sowohl im Einzelspiel als auch in Kombination mit den anderen) bzw. Musikgenres vertreten sein. Leider fehlt es in der Literatur an Überlegungen bzw. Vorschlägen zu einer repräsentativen Musikdatenbank, in der diese und weitere sinnvolle Kriterien für eine geschichtete Auswahl der Musikstücke berücksichtigt sein werden. Eine bedeutende Arbeit in dieser Richtung ist die Zusammensetzung eines repräsentativen Korpus der klassischen Musik von London (2013), wobei dort nur Namen der Komponisten und deren bedeutendsten Werke genannt werden, ohne eine endgültige Liste der Musikstücke bzw. eine Sammlung der Audiodateien anzugeben.

Die größte Problematik für die Erstellung einer solchen Datenbank ist die Notwendigkeit der Information über die wahren Toneinsatzzeiten. Wie in Abschnitt 3.1 bereits erwähnt wurde, gibt es zwei Wege, Audiodateien mit den zugehörigen Einsatzzeiten zu erhalten. In der gängigen Literatur wird die erste Möglichkeit verwendet: (oft recht kleine) Datenbanken der manuell annotierten Musikstücke. Meistens werden die Aufnahmen in solchen Datenbanken nach Instrumenten bzw. Instrumentenarten unterteilt. Selten wird der Musikstil zur Unterteilung herangezogen. Bei Holzapfel et al. (2010) wird z.B. zwischen europäischen und türkischen Instrumenten – und somit dem Stil – unterschieden.

Die zweite Möglichkeit – Verwendung des MIDI-Formats – bietet den Vorteil, dass sehr viele Musikstücke mit der notwendigen Information über die wahren Einsatzzeiten produziert werden können. Im Internet sind große Archive klassischer und moderner Musik im MIDI-Format erhältlich. Man hat hier aber das umgekehrte Problem: die wahren Einsatzzeiten sind vorhanden, aber keine Originalaufnahme eines Gesamtklanges. Diese sollen dann aus den MIDI-Daten generiert werden. Es existieren verschiedene Programme, die aus den in MIDI angelegten Toneigenschaften die Töne synthetisch generieren und aneinander knüpfen. Dies geschieht mit Hilfe von instrumentspezifischen Signalmodellen. Ein synthetisch generiertes Musikstück kann natürlich nicht den Klang einer realen Musikaufnahme vollständig nachbilden.

Von Interesse ist, ob die Performance der Algorithmen zur EZE von der Erzeugungsart der Musikstücke abhängt. Denn es wäre nicht angemessen, nur auf synthetischen Stücken EZE-Algorithmen zu optimieren und dann auf echte anzuwenden (und umgekehrt). Im Weiteren werden Eigenschaften echter und künstlich generierter Stücke exemplarisch dargestellt. Außerdem wird eine Verbesserung der künstlichen Musikstück-Generierung vorgestellt, die auf einer Datenbank der echten Töne basiert. Anschließend wird die Zusammenstellung der Musikdatenbanken für die nachfolgende Optimierung diskutiert.

5.1 Gegenüberstellung echter und künstlich generierter Musikstücke

Es werden hier erneut Aufnahmen des Halleluja Liedes betrachtet (s. Beispiel 3.2.1). Im ersten Schritt wird basierend auf dem Notenblatt (vgl. Abbildung 3.2) die MIDI-Datei zu der ersten Strophe des Liedes erstellt. Dazu wird die Software *Anvil Studio*[1] verwendet. In Abbildung 5.1 ist das Programm-Fenster veranschaulicht. Für Klavier wird das Instrument 1: Acoustic Grand und für Flöte das Instrument 74: Flute eingestellt. Die letzte Note sollte laut dem Notenblatt eine halbe Note sein. Jedoch wird sie bei den beiden Originalaufnahmen als eine Viertelnote gespielt. Deswegen wird sie – abweichend vom Notenblatt – als eine Viertelnote im MIDI-Schema eingegeben.

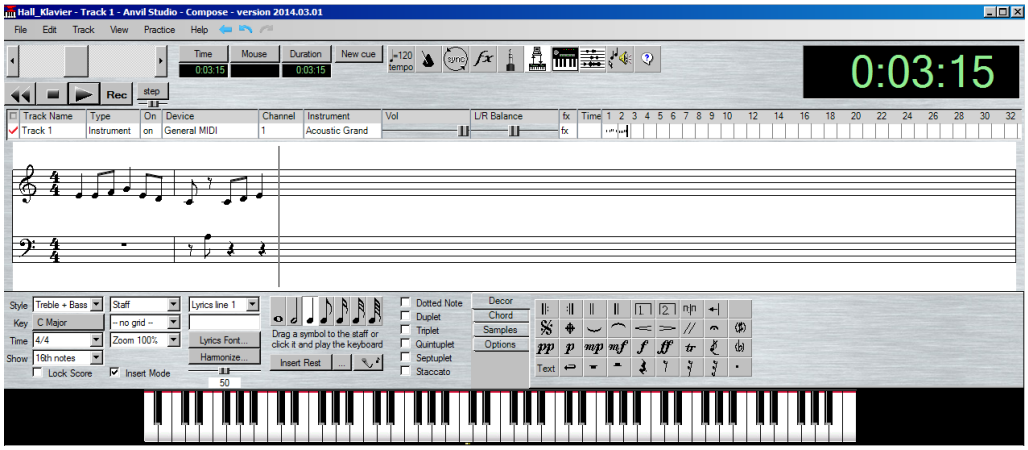

Abbildung 5.1: Benutzeroberfläche des *Anvil Studio* Programms.

Die beiden MIDI-Files werden einmal mit Hilfe synthetischer Töne und einmal mit Hilfe einer Datenbank der echten Töne zu WAV-Files konvertiert. Das Ziel der Analyse in diesem Abschnitt besteht darin, mögliche Unterschiede in den Eigenschaften der originalen und der aus MIDI generierten Stücke festzustellen. Es wird vermutet, dass die „synthetischen“ Tonsequenzen bessere Erkennungsgüte aufweisen als die originalen Aufnahmen (wegen der einfachen und evtl. leicht erkennbaren Struktur der Tonmodellierung). Abbildungen 5.2 bzw. 5.3stellen den Amplitudenverlauf (links) sowie die Spektrogramme (rechts) der drei betrachteten Stücke für jeweils Klavier und

[1] Version 2009.06.006, `www.anvilstudio.com`, Stand: 01.01.2015.

Flöte dar[2]. Im Weiteren werden die beiden Arten der Musikgenerierung näher erläutert und die Abbildungen diskutiert.

5.1.1 Erzeugung mittels synthetischer Töne

Für diese Art der Tonsequenz-Generierung wurde die *MIDI to WAVE Converter*[3] Software eingesetzt. Ein allgemeiner Nachteil vermutlich aller solcher Konverterierungsprogramme ist die Tatsache, dass identische Töne immer durch dasselbe Modell generiert werden und somit immer gleich sind. Bei den realen Aufnahmen ist es dagegen unmöglich, zwei gleiche Töne exakt gleich zu wiederholen. Bei solchen Instrumenten wie Gitarre oder Geige kommt außerdem noch hinzu, dass bestimmte Töne auf unterschiedlichen Saiten gespielt werden können. So kann z.B. der Ton G5 bei einer Geige sowohl auf A als auch auf E Saite erzeugt werden, wobei es keine allgemeine Regel existiert, welche Saite in welchem Fall zu bevorzugen ist.

Blasinstrumente sind in Hinsicht auf die Modellierung des Tonendes etwas problematisch. Denn ein einfaches Abschneiden der Töne an einer beliebigen Stelle erzeugt ein unnatürliches Störgeräusch. Idealerweise sollte ein Ton in der für ihn in MIDI vorgesehenen Zeit vollständig abklingen, was allerdings nicht der Fall ist. Die Töne werden nämlich nach dem Schema Ansatzphase, stabile Phase und Endphase modelliert. Zu dem Zeitpunkt des vorgesehenen Tonendes hört die stabile Phase auf, so dass die Endphase in der Zeit des nächsten Tones stattfindet. Während dieses Schema z.B. bei Klavier den natürlichen Vorgang gut modelliert (beim Loslassen der Taste kann der Ton noch eine Weile klingen), ist das für Blasinstrumente meistens nicht der Fall.

Abbildung 5.3 (unten) veranschaulicht die oben erklärte Problematik: durch eine sehr langsame Ansatzphase (ca. 50 ms) und eine lange Endphase des vorherigen Tones entsteht der Eindruck, dass die wahren Toneinsatzzeiten nach links verschoben sind. Weiterhin ist zu bemerken, dass die Amplitude des erzeugten Signals um ein Vielfaches kleiner ist als die der Originalaufnahme (d.h. das Stück hört sich leiser an), was offenbar an internen Einstellungen des *MIDI to WAVE Converters* liegt. Das Spektrogramm der synthetisch erzeugten Flötentöne sieht etwas obertonärmer aus als die beiden anderen Spektrogramme.

Die Amplitude des mittels synthetischer Töne generierten Klavierstückes weist keine Besonderheiten auf (vgl. Abbildung 5.2). Das Originalsignal scheint gut nachgebildet zu sein, abgesehen von der bereits erwähnten Tatsache, dass der erste, der zweite, der fünfte und der elfte Ton (E4) immer gleich aussehen[4]. Das Spektrogramm der Originalaufnahme des Klaviers ist überraschend obertonarm. Im Vergleich zu der mittels echter Töne erzeugten Sequenz (mittleres Bild) weisen die synthetischen Töne etwas weniger Obertöne auf.

---

[2] Alle sechs Audiodateien befinden sich im Ordner `Musikbeispiele/Halleluja` der beigelegten CD.

[3] `http://www.maniactools.com/soft/midi_converter/index.shtml`, Stand: 01.01.2015.

[4] Die kleinen scheinbaren Abweichungen in den Amplituden der genannten Töne kommen nur dadurch zustande, dass beim Plotten nicht alle Amplituden veranschaulicht werden, sondern nur eine reduzierte Auswahl.

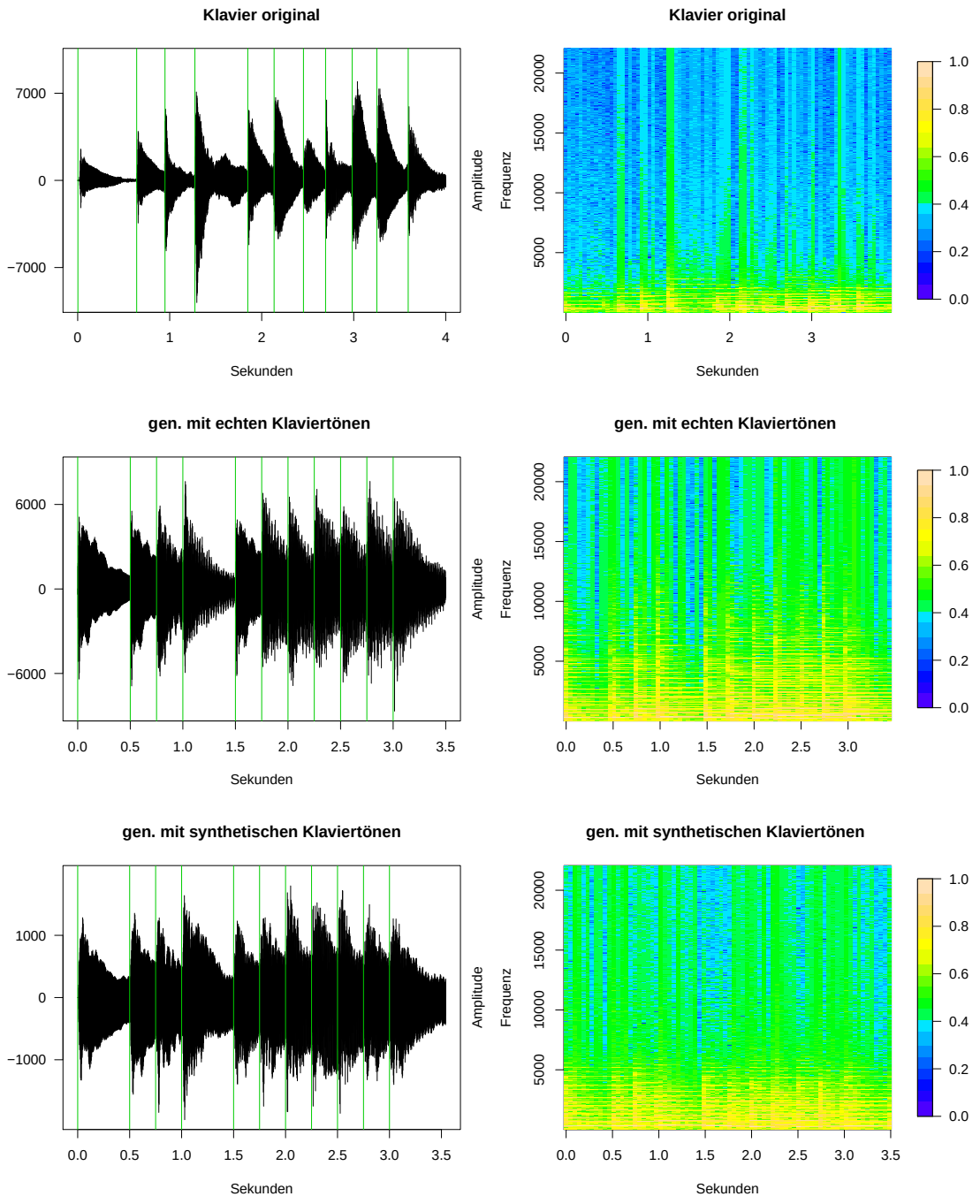

Abbildung 5.2: Amplitudenentwicklungen (links) und Spektrogramme (rechts) einer originalen Klavieraufnahme der ersten Strophe des Halleluja-Liedes (oben), sowie der mittels echter (mittig) und synthetischer (unten) Töne generierten (Abkürzung: gen.) Tonsequenzen.

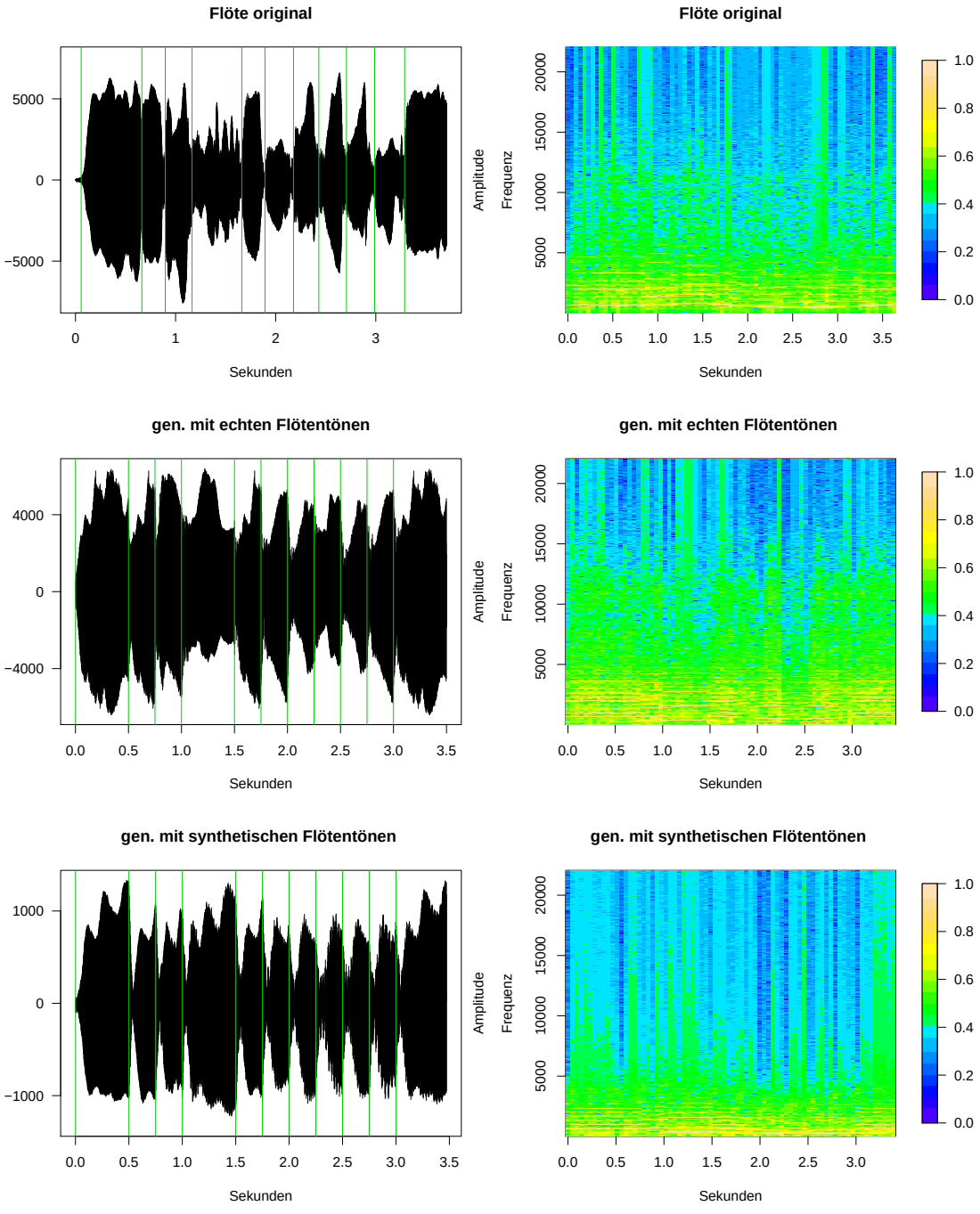

Abbildung 5.3: Amplitudenentwicklungen (links) und Spektrogramme (rechts) einer originalen Flötenaufnahme der ersten Strophe des Halleluja-Liedes (oben), sowie der mittels echter (mittig) und synthetischer (unten) Töne generierten (Abkürzung: gen.) Tonsequenzen.

5.1.2 Erzeugung mittels echter Töne

Für dieses Ziel wird ein speziell entwickeltes Konvertierungsprogramm verwendet, welches bei Bauer et al. (2012) in Detail erklärt wird, wobei die echten Tonaufnahmen aus der kommerziellen Tonbibliothek *RWC music database* (Goto et al., 2003) stammen. Diese Datenbank bietet Töne verschiedener Musikinstrumente in mehreren Ausführungen an: zu jeder Tonstärke (Forte, Mezzoforte und Piano) gibt es jeweils mehrere Spielweisen (z.B. Normal, Staccato und Vibrato). Außerdem sind für jedes Musikinstrument die Töne von mindestens zwei Exemplaren dieses Instruments aufgenommen worden.

Der Programmcode sowie eine kurze Anleitung für das Konvertierungsprogramm befinden sich im Ordner `RealConverter` auf der beiliegenden CD. Der Code ist geschrieben in der Sprache *R* (R Core Team, 2014) und basiert auf dem **tuneR**-Paket (Ligges et al., 2014). Momentan gelingt die Konvertierung nur für Klarinette, Flöte, Gitarre, Klavier, Trompete und Geige[5]. Die für das *RealConverter*-Programm zur Verfügung stehenden Instrumente und deren Tonumfänge (inklusive MIDI-Kodierung) sind in Tabelle 5.1 aufgelistet. Es existieren auch kommerzielle Echtton-Konvertierungsprogramme wie *Vienna Symphonic Library*[6], die hier allerdings aus Kosten- und Komplexitätsgründen nicht herangezogen wurden.

Tabelle 5.1: Die in *RealConverter* implementierten Instrumente und deren Tonumfänge.

Instrument	*Tonumfang*	*MIDI-Codierung*
Flöte	C4 - C7	60 - 96
Geige	G3 - E5	55 - 100
Gitarre	E2 - E5	40 - 76
Klarinette	D3 - F6	50 - 89
Klavier	A0 - C8	21 - 108
Trompete	E3 - D6	52 - 86

Um den „Wiederholungseffekt“ bei gleichen Tönen etwas zu mindern, wird für jeden Ton entsprechend seiner Länge und Lautstärke entschieden, aus welcher Tonbibliothek er stammen soll. Außerdem besteht die Möglichkeit, das Exemplar des zu spielenden Instruments zufällig auszuwählen, was allerdings – falls Instrumentenwechsel deutlich hörbar ist – zu unnatürlich klingenden Stücken führen kann. Der bessere Weg ist hier deswegen eine zufällige Wahl des Instrumentenexemplars vor Beginn der Konvertierung. Besteht das MIDI-Schema aus mehreren Spuren, erfolgt für jede Spur eine zufällige Zuweisung. Bei dieser Technik entsteht der Eindruck, dass mehrere Instrumente gleichzeitig spielen[7]. Außerdem kann angegeben werden, ob bei Gitarre und

[5] Da die *RWC* Tondatenbank urheberrechtlich geschützt ist, können die benötigten Töne nicht veröffentlicht werden. Sie befinden sich daher auf einem internen Server des Lehrstuhls Computergestützte Statistik.

[6] http://vsl.co.at, Stand: 01.01.2015.

[7] Im Ordner `Musikbeispiele/Kapitel_Musikdatenbank` der beiliegenden CD befinden sich zwei Dateien, die

Geige die Saite, auf der ein Ton zu spielen ist, zufällig ausgewählt oder immer die höhere Saite verwendet werden soll[8].

Auch hier ist das vorzeitige Beenden eines Tones problematisch und geschieht auf die ähnliche Art und Weise wie bereits im vorherigen Abschnitt beschrieben. Der Unterschied liegt allerdings darin, dass die echten Tonaufnahmen zuerst getrimmt wurden: am Tonanfang wurde ein Abschnitt mit Amplituden unter einem festgelegten Niveau abgeschnitten[9]. Außerdem wurde ein instrumentabhängiges Vorgehen (Bauer et al., 2012) für das Beenden der Töne verwendet, so dass die Töne der Blasinstrumente zwar immer noch nach der für sie vorgesehenen Zeit abklingen, dafür aber relativ schnell und ohne Störgeräusche.

Eine weitere Besonderheit der Echtton-Generierung soll nicht unerwähnt bleiben, nämlich die Restriktion durch den Tonumfang der zu spielenden Instrumente. Wie aus dem Notenblatt des Halleluja-Liedes hervorgeht, ist der achte Ton der Ton A3. Dieser kann im Normalfall von der Flöte nicht gespielt werden. Zwar können professionelle Flötisten ihn dennoch spielen (in der Originalaufnahme wird A3 tatsächlich getroffen), für eine künstliche Erzeugung des Stückes soll aber ein Ausweg gefunden werden (weil dieser Ton in der *RWC* Datenbank nicht vorhanden ist). Hier wurde einfach der Ton A4 statt A3 verwendet, um den Vergleich zwischen verschiedenen Erzeugungsarten auch für die Flöte stattfinden zu lassen. Bei den konventionellen Konvertierungsprogrammen spielt die Tonhöhe keine Rolle, so dass auch Töne generiert werden können, die außerhalb der Tonumfänge der entsprechenden Instrumente liegen. In Abbildungen 5.2 und 5.3 sind die mittels *RealConverter* erzeugten Sequenzen mittig zu sehen. Bei den beiden Instrumenten fallen die obertonreichen Strukturen im Spektrogramm auf.

5.1.3 Unterschiede zwischen den Generierungsarten

Hier werden exemplarisch zwei Merkmale auf die sechs Halleluja-Versionen angewendet: das auf der Veränderung der maximalen absoluten Amplitude basierende Merkmal $Ampl.Max.Abs.Diff^{offset}$ und das spektralbasierte Merkmal $Spec.Euclid^{offset}$. In den Abbildungen 5.4 und 5.5 sind die Verläufe der Merkmals- und Schwellenwertfunktionen, die geschätzten und die wahren Einsatzzeiten sowie die F- und D- Werte dargestellt. Außerdem sind die Einstellungen der restlichen Parameter der Einsatzzeiterkennung erwähnt. In Tabelle 5.2 sind die F- und D- Werte zusammengefasst.

Mit Ausnahme des amplitudenbasierten Merkmals für Klavier, sind die F-Werte der aus MIDI erzeugten Stücke immer besser als die der Originalaufnahme. Aufgrund dieses Ergebnisses scheint

---

den Unterschied zwischen mehrspurigen Aufzeichnungen veranschaulichen: `GitarreRealConverter.wav` (erzeugt mittels *RealConverter*) und `GitarreSynthConverter.wav` (erzeugt mittels *MIDI to WAV Converter*).

[8] Im Ordner `Musikbeispiele/Kapitel_Musikdatenbank` der beiliegenden CD befinden sich Dateien `GeigeSaiteFest.wav` und `GeigeSaiteZufall.wav`, die den Unterschied zwischen zufällig ausgewählten und fixierten Saiten veranschaulichen.

[9] Mit Hilfe der Matlab Funktion `miraudio` aus **MIRToolbox** (`https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox`, Stand: 01.01.2015), wobei der Parameter `TrimThreshold` auf 0.01 eingestellt wurde.

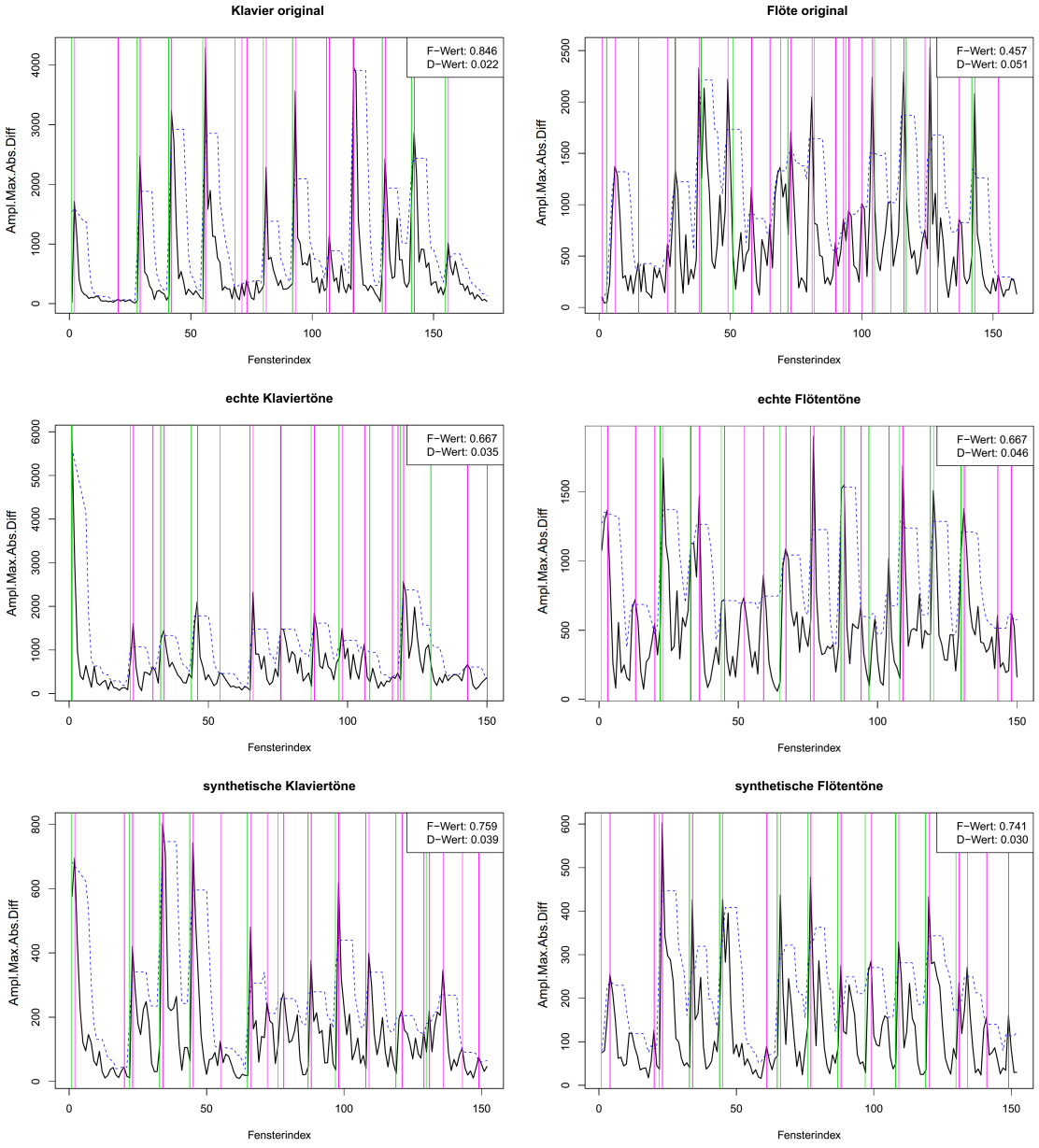

Abbildung 5.4: Anwendung des Merkmals $Ampl.Max.Abs.Diff^{offset}$ auf die originalen sowie die mit echten und synthetischen Tönen erzeugten Musikstücke von Klavier und Flöte. Schwarze Linie: das Merkmal, blaue Linie: die Schwellenwertfunktion, grüne vertikalen Linien: die wahren Toneinsätze, pinke vertikalen Linien: die geschätzten Toneinsätze. Weitere Parameter: $N = h = 1024$, Fensterungsfunktion: *Hanning*, *spec.filter* = *spec.log* = „Nein“, $\alpha = 0.7$, $th.fun = mov.quantile$, $p = 0.9$, $t(r_T) = t(r_O) = 0$ ms, $t(l_T) = 120$ ms, $t(l_O) = 30$ ms, $min.dist = 50$ ms, $onset.shift = 0$ ms.

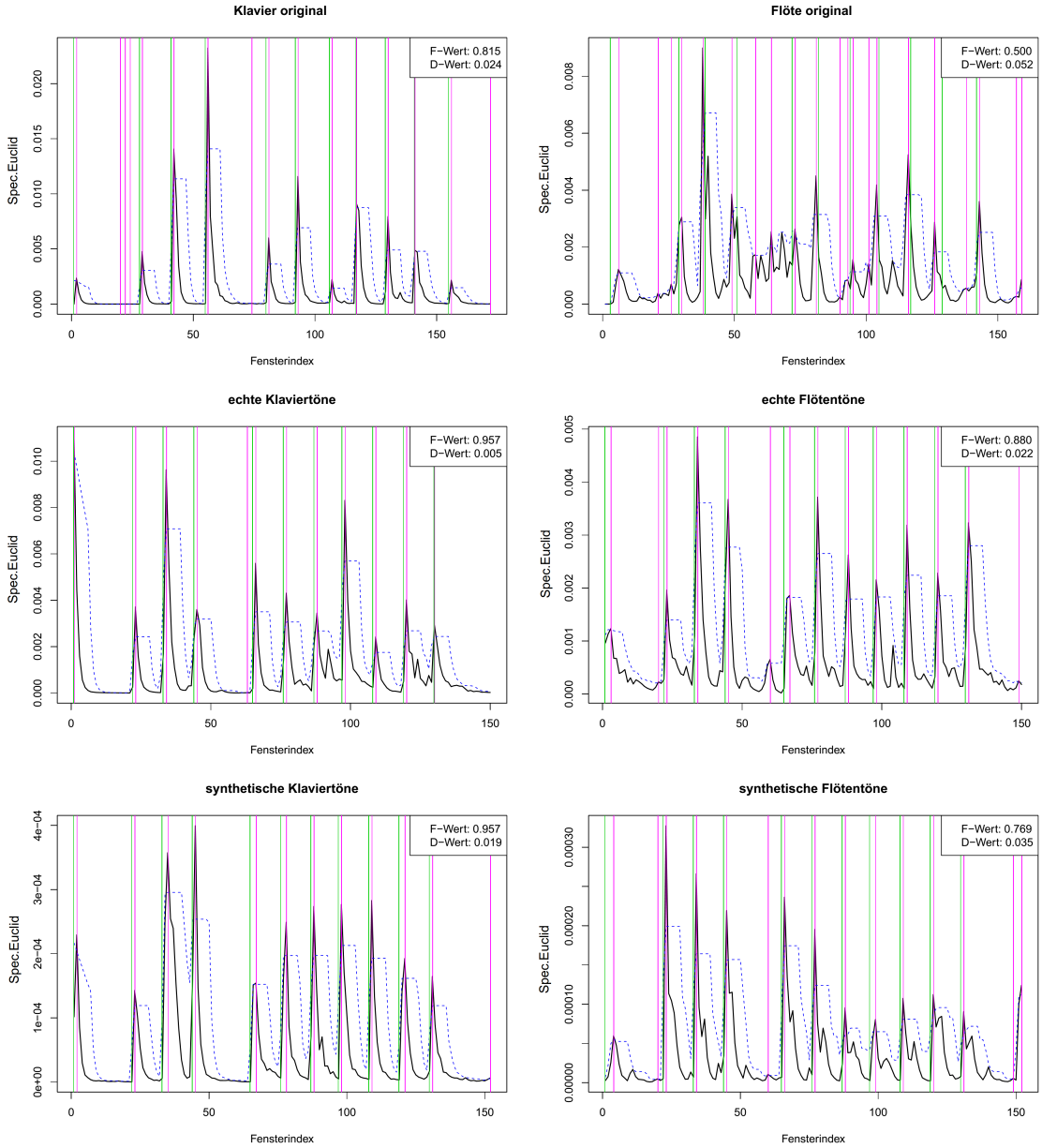

Abbildung 5.5: Anwendung des Merkmals $Spec.Euclid^{offset}$ auf die originalen sowie mit echten und synthetischen Tönen erzeugten Musikstücke von Klavier und Flöte. Schwarze Linie: das Merkmal, blaue Linie: die Schwellenwertfunktion, grüne vertikalen Linien: die wahren Toneinsätze, pinke vertikalen Linien: die geschätzten Toneinsätze. Weitere Parameter wie in Abbildung 5.4.

die im Anfang des Kapitels formulierte Vermutung plausibel zu sein. Natürlich kann hiermit nicht ihre allgemeine Gültigkeit behauptet werden, da zu wenig Daten und nur zwei Algorithmuseinstellungen überprüft wurden. Das Verhalten der mittels echter Töne erzeugten Sequenzen ist sehr unterschiedlich: für das spektralbasierte Merkmal liefern sie die besten F-Werte, während sie für das amplitudenbasierte Merkmal den synthetischen Tönen unterlegen sind. Die gleichen Schlussfolgerungen würde man ziehen, wenn man den D-Wert betrachten würde. Interessant zu bemerken

ist, dass das spektralbasierte Merkmal den gleichen F-Wert für die beiden Erzeugungsarten (für Klavier) aufweist. Allerdings ist der D-Wert der Echtton-Generierung deutlich kleiner. Insgesamt scheint die Erzeugungsart bei dem Blasinstrument Flöte einen größeren Einfluss auf die Erkennungsgüte zu haben als bei Klavier.

Tabelle 5.2: F- und D-Werte für die originalen und die künstlich generierten Halleluja-Aufnahmen.

Merkmal	*Erzeugung*	*F-Wert*		*D-Wert*	
		Klavier	*Flöte*	*Klavier*	*Flöte*
$Ampl.Max.Abs.Diff^{offset}$	original	0.846	0.457	0.002	0.051
	mit echten Tönen	0.667	0.667	0.035	0.046
	mit synth. Tönen	0.759	0.741	0.039	0.030
$Spec.Euclid^{offset}$	original	0.815	0.500	0.024	0.052
	mit echten Tönen	0.957	0.880	0.005	0.022
	mit synth. Tönen	0.957	0.765	0.019	0.035

5.2 Zusammenstellung der Datenbanken

Bei der Erstellung einer Musikdatenbank sollten möglichst viele Aspekte abgedeckt werden: Generierungsart, Art des Musikinstruments, Grad der Polyphonie, Tempo, Musikgenre bzw. Musikstil. Im Idealfall sollten zu jedem Aspekt die gewünschten Kategorien aufgelistet werden mit dem Ziel, ein voll-faktorielles Design für die Erstellung des Datensatzes zu entwerfen. In der Praxis ist das sehr schwierig zu implementieren, da z.B. jedes reale Musikstück im MIDI-Format nachimplementiert werden müsste.

Aus diesem Grund werden verschiedene Datenbanken, die die gewünschten Aspekte nur teilweise berücksichtigen, zur Optimierung herangezogen. Zu bemerken ist allerdings, dass es nicht sinnvoll ist, über die echten und aus MIDI erzeugten Aufnahmen gleichzeitig zu optimieren. Denn bei den beiden Typen der Aufnahmen würden unterschiedliche Definitionen des Toneinsatzes in Frage kommen (wahrnehmbarer Toneinsatz bei echten Aufnahmen und physikalischer Toneinsatz bei aus MIDI erzeugten Aufnahmen, s. Abschnitt 3.1). Somit können die optimalen Parametereinstellungen (insbesondere der *onset.shift* Parameter, s. Abschnitt 3.8) für die beiden Aufnahmetypen stark variieren. Um diese möglichen Unterschiede zu erforschen, werden im Weiteren zwei Datensätze gebildet und auf den beiden die Einsatzzeiterkennung optimiert.

Der erste Datensatz, im Späteren auch als WAV-Datensatz bezeichnet, enthält die „per Hand“ annotierten Stücke (Bello-Datenbank, Holzapfel-Datenbank und Böck-Datenbank, s. unten). Diese Datenbank wird vorrangig analysiert, da somit ein Vergleich mit „State of the Art“ Veröffentlichungen möglich ist. Der zweite Datensatz – MIDI-Datensatz – enthält die aus den MIDI-Files generierten Stücke (Volkslied-Datenbank, Musikepochen-Datenbank, Internet-Datenbank, s. unten). Mit Hilfe einiger Datenbanken des MIDI-Datensatzes wird außerdem der Einfluss des Musikinstrumentes und der Generierungsart auf die Güte der EZE analysiert.

Die insgesamt 6 Datenbanken (bestehend aus 460 Musikdateien mit 58 545 Toneinsätzen) befinden sich auf einem internen Lehrstuhl-Server der Fakultät Statistik.

Bello-Datenbank. Die Bello-Datenbank (Bello et al., 2005) wurde freundlicherweise nach Anfrage zur Verfügung gestellt. Es wurde auch darauf hingewiesen, dass die Anzahl der Toneinsätze im Rahmen einer umfangreichen Revision im Vergleich zu dem Originalpaper nach unten korrigiert wurde (7 Einsätze wurden entfernt). Es handelt sich dabei um eine sehr oft zitierte manuell annotierte Datenbank bestehend aus 23 Aufnahmen und 1 058 Toneinsätzen. Es wird zwischen drei Arten von Musikinstrumenten unterschieden: *Pitched Percussion* (PP) als Schlaginstrumente, die für die Produktion der musikalischen Noten verwendet werden (z.B. Glockenspiel), *Non Pitched Percussive* (NPP) als Schlaginstrumente, die keinen musikalischen Ton produzieren (z.B. Becken) und *Pitched Non Percussive* (PNP) als alle anderen Instrumente, die eine musikalische Note wiedergeben können (z.B. Klavier). Außerdem werden gemischte Aufnahmen dieser drei Musikinstrumentarten betrachtet.

Holzapfel-Datenbank. Auch hier wurde der „per Hand“ annotierte Datensatz aus Holzapfel et al. (2010) freundlicherweise von André Holzapfel zur Verfügung gestellt. Es wurden ebenso die von Sebastian Böck (Böck et al., 2012) korrigierten Einsatzzeiten für diesen Datensatz mitgeliefert. Der Datensatz unterteilt sich in den „Development Datensatz“ (mit 29 Dateien), der für die Trainingszwecke verwendet wurde und den „Testdatensatz“ (mit 63 Dateien). Insgesamt enthalten die beiden Datensätze 3 278 Toneinsätze. Auch hier wird nach Instrumentenart unterschieden: PP, Blasinstrumente, Streichinstrumente und gemischt. Außerdem wird in jeder Instrumentengruppe mindestens ein orientalisches Instrument verwendet und somit laut den Autoren zwischen abendländischem und orientalischem Musikstil unterschieden.

Böck-Datenbank. In Böck et al. (2012) werden neben den Datenbanken von Bello und Holzapfel 206 weitere manuell annotierte Aufnahmen verwendet. Der Hauptautor hat nach Anfrage die Daten zur Verfügung gestellt. Die Böck-Datenbank enthält also 206 Dateien mit insgesamt 23 414 Toneinsätzen. Es handelt sich dabei um sehr heterogene Aufnahmen: von Klassik bis Pop, von nur einem Musikinstrument bis zu großen Ensembles, mit und ohne Gesangsstimme.

Anlehnend an Böck et al. (2012) werden die wahren Einsatzzeiten aggregiert: da es für das menschliche Gehör in der Regel nicht möglich ist, mehrere hintereinander folgende Toneinsätze, die innerhalb von 30 ms stattfinden, auseinander zu halten, werden solche Toneinsätze zu einem Einsatz aggregiert. Die Toneinsatzzeit entspricht dann der mittleren Zeit dieser Einsätze. Das betrifft allerdings nur sehr wenige Musikstücke dieser und der beiden oben erwähnten Datenbanken.

Volkslied-Datenbank. Hier handelt es sich um einen in Bauer et al. (2014) vorgestellten Datensatz, der für eine systematische Untersuchung der Einflüsse von Musikinstrument und Tempo auf die Güte der Einsatzzeiterkennung entworfen wurde. Dabei wurden sechs Musikinstrumente

untersucht, die einen gemeinsamen Tonumfang von 17 Tönen aufweisen: von C4 bis E5. Es konnten MIDI-Files zu zwei deutschen Volksliedern[10] gefunden werden, welche in dem gewünschten Tonumfang liegen. Das Tempo wird üblicherweise in Beats pro Minute (BPM) gemessen. Hier wurden zwei Einstellungen gegenübergestellt: 90 BPM (klassische Bezeichnung: *Andante*) und 200 BPM (klassische Bezeichnung: *Presto*). Insgesamt umfasst der Datensatz 24 Musikstücke mit 3 120 Toneinsätzen: 2 MIDI-Vorlagen, 6 Instrumente und 2 Tempi. Die MIDI-Dateien wurden mit dem *RealConverter*-Programm in WAV umgewandelt.

Musikepochen-Datenbank. Ein Teil dieser Datenbank (12 Stücke) wurde bereits in Bauer et al. (2013b) vorgestellt und zwar handelt es sich um klassische europäische Musik aus 6 Zeitepochen: Mittelalter, Renaissance, Barock, Klassik, Romantik und Neue Epoche. Pro Epoche wurden zwei berühmte Komponisten ausgewählt mit jeweils einem Musikstück (als MIDI-Datei). Bei allen 12 MIDI-Files wurde Klavier als Instrument eingestellt (obwohl die Stücke aus früheren Zeitepochen zu damaliger Zeit eher für Oboe und Flute geschrieben wurden). Hiermit wurde angestrebt, verschiedene Musikstile abzudecken, die sich in mehreren Charakteristiken unterscheiden.

Der zweite Teil der Datenbank basiert auf 10 Musikstücken in MIDI-Format: sechs Solo Stücke (Klarinette, Flöte, Gitarre, Klavier, Trompete und Geige (2 Stücke)) und 3 Duette (Geige mit Flöte, Klavier mit Flöte und Klavier mit Geige).

Eine weitere Besonderheit dieser Datenbank besteht darin, dass jedes der 22 MIDI-Files einmal mit synthetischen Tönen (mittels *MIDI to WAV Converter*) und einmal mit echten Tönen (mittels *RealConverter*) in WAV umgewandelt wurde. Somit ergeben sich 44 WAV-Dateien mit 14 456 Toneinsätzen. In den beiden Fällen handelt es sich um die ersten 60 Sekunden der ursprünglichen MIDI-Files.

Internet-Datenbank. Es handelt sich hier um die in Bauer et al. (2015) verwendete Datenbank. Dabei wurden ursprünglich über 1 000 im Internet frei verfügbare MIDI-Dateien ausgewählt, so dass Repräsentanten verschiedener Musikstile (darunter viel Pop-Musik) und Instrumente vertreten sind. Anschließend wurden 200 MIDI-Dateien mit mindestens 50 ms Abstand zwischen zwei aufeinanderfolgenden Toneinsätzen ausgewählt, welche die Grundlage der Internet-Datenbank bilden. Die ersten 60 Sekunden der MIDI-Files wurden mit dem *MIDI to WAV Converter* (also mit synthetischen Tönen) in WAV umgewandelt. Die komplette Datenbank enthält 45 491 Toneinsätze.

[10] `http://www.ingeb.org/Lieder/haidschi.mid` und `http://www.ingeb.org/Lieder/esgetein.mid`, Stand: 01.01.2015.

Kapitel 6

Optimierung

6.1 Klassische und sequentielle Optimierung

Parameteroptimierung ist ein wichtiges Thema sowohl für industrielle Prozesse als auch für computerbasierte Anwendungen. Das klassische Vorgehen bei der Optimierung industrieller Prozesse basiert auf der Versuchsplanung, welche im Allgemeinen in mehreren Stufen erfolgt (z.B. Screening, Modellierung und Optimierung, Weihs und Jessenberger, 1999). Dabei werden die Versuche nach einem vorher festgelegten Plan durchgeführt, so dass die sogenannte Planmatrix bestimmte statistische Kriterien erfüllt (wie A- oder D-Optimalität). Anschließend wird der Zusammenhang zwischen der zu optimierenden Zielvariable und den Einflussfaktoren mittels eines Regressionsmodells (üblicherweise das lineare Modell mit Wechselwirkungen und quadratischen Termen) modelliert. Die optimale Parametereinstellung wird dann als Punkt mit der besten Modellvorhersage ermitteltt. In den meisten Fällen wird zusätzlich ein Bestätigungsversuch in diesem Punkt durchgeführt und die Güte des Modells beurteilt.

Das klassische Vorgehen ist zwar von einer langen Erfolgsgeschichte gekennzeichnet (Spall, 2003, S. 467), hat allerdings zwei relevante Einschränkungen: reale Experimente sind oft zeitaufwändig bzw. kostspielig und das verwendete lineare Modell ist nicht angemessen, um den bestehenden komplexen Zusammenhang zwischen den Ziel- und Einflussgrößen zu beschreiben.

Das erste Problem kann in einigen Fällen dadurch gelöst werden, dass die realen Prozesse durch aufwändige Simulationen ersetzt werden. Allerdings kann selbst bei einer Simulation ein einziger Lauf mehrere Tage oder Monate in Anspruch nehmen (Herbrandt et al., 2015). Das verdeutlicht die Notwendigkeit effizienter Optimierungsstrategien, die mit wenigen Versuchen auskommen. Was das zweite Problem angeht, wurden in den letzten Dekaden verschiedene Methoden der nicht-linearen Optimierung entwickelt (Stoer und Jarre, 2004; Spall, 2003) wie der Bergsteigeralgorithmus (engl. *hill climbing*), die simulierte Abkühlung (engl. *simulated annealing*), die evolutionären Algorithmen (engl. *evolutionary algorithms*) oder die sequentielle modellbasierte Optimierung (engl. *sequential model based optimization*).

Die sequentielle modellbasierte Optimierung hat nach der wegweisenden Arbeit von Jones et al. (1998) besonders stark an Popularität gewonnen. Anders als bei den anderen oben erwähnten Verfahren werden hier keine „günstigen“ Probleme (im Bezug auf die Evaluierungszeit) vorausgesetzt. Dabei werden, im Gegensatz zur klassischen Optimierung, die neuen Versuchspunkte basierend auf einem Modell, welches oft als *Surrogat-* bzw. *Metamodell* bezeichnet wird (Spall, 2003, S. 468), iterativ vorgeschlagen (Bartz-Beielstein et al., 2005). Allerdings ist auch hier ein Startdesign notwendig, um das Surrogatmodell anzupassen. Im nächsten Abschnitt wird das Optimierungsproblem formell eingeführt und das Schema der modellbasierten Optimierung detailliert erklärt. Anschließend wird auf die instanzgebundene Optimierung eingegangen, welche für die Einsatzzeiterkennung von besonderem Interesse ist.

6.2 Modellbasierte Optimierung

Ziel der modellbasierten Optimierung (MBO) ist die Minimierung[1] einer zeitintensiven Zielfunktion $f : \mathscr{X} \subset \mathbb{R}^d \to \mathbb{R}$, $f(\boldsymbol{x}) = y$, $\boldsymbol{x} = (x_1, \ldots, x_d)^T$. Jeder Funktionsparameter x_i sollte auf dem vorgegebenen Optimierungsintervall $[\ell_i, u_i]$ (auch als zulässiges Intervall bezeichnet) stetig sein[2]. Der *Optimierungsbereich* der Zielfunktion (engl. *region of interest* bzw. *parameter space*) ist dann das kartesische Produkt einzelner Intervalle: $\mathscr{X} = [\ell_1, u_1] \times \ldots \times [\ell_d, u_d]$. Eine mögliche Parametereinstellung $\boldsymbol{x}_i \in \mathscr{X}$ heißt ein *Punkt*, wobei y_i der Zielfunktionswert dieses Punktes ist. Eine *Designmatrix* bzw. *Design* besteht aus einer Menge von n Punkten und wird im Weiteren mit $\mathscr{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ bezeichnet. Weiterhin ist $\boldsymbol{y} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^T$ der Vektor der Zielfunktionswerte auf $\mathscr{D}$.

In einfachen Fällen ist die Zielfunktion deterministisch: $f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2)$ für $\boldsymbol{x}_1 = \boldsymbol{x}_2$. Bei der Optimierung realer industrieller Prozesse sowie zufallsbedingter Simulationen arbeitet man allerdings sehr häufig mit verrauschten Zielgrößen, d.h. man beobachtet nicht den wahren Wert der Zielfunktion, sonder einen durch das Rauschen verzerrten Wert $f(\boldsymbol{x}) + \varepsilon$. Die Verteilung von ε kann von $\boldsymbol{x}$ unabhängig oder aber abhängig sein (d.h. unterschiedliche Intensität des Rauschens in verschiedenen Parameterbereichen). Je stärker das Rauschen ist, desto schwieriger gestaltet sich der Optimierungsprozess. In Rahmen dieser Arbeit werden deterministische Funktionen optimiert, nichtsdestotrotz werden an einigen Stellen weiterführende Veröffentlichungen zu diesem und anderen forschungsrelevanten Themen als Ausblick behandelt.

Das in Algorithmus 6.1 dargestellte Vorgehen von MBO kann wie folgt zusammenfasst werden: Im ersten Schritt wird ein Startdesign von n Punkten ausgewertet und das Surrogatmodell angepasst. Solange das Optimierungsbudget nicht ausgeschöpft ist, wird ein neuer Punkt basierend auf einem Zielkriterium (z.B. Modellvorhersage) im Optimierungsbereich ausgewählt und

[1] Falls die Maximierung gewünscht ist, wird die Ausgabe der Zielfunktion mit einem Minuszeichen versehen, so dass im Weiteren die Minimierung erfolgen kann.

[2] Auf die Behandlung von kategoriellen Variablen wird in nachfolgenden Abschnitten eingegangen.

ausgewertet. Das Surrogatmodell wird dann auf der um einen Punkt erweiterten Designmatrix neu gelernt und für die anschließende Iteration verwendet. Der Punkt mit dem minimalen Zielfunktionswert gilt dann als Ergebnis der Optimierung. Nachfolgend werden die wichtigsten Schritte dieses Algorithmus näher diskutiert.

Algorithmus 6.1: Sequentielle modellbasierte Optimirung (MBO).

1 Generiere ein Startdesign $\mathscr{D} \subset \mathscr{X}$;
2 Evaluiere die Zielfunktion auf dem Startdesign: $\boldsymbol{y} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^T$;
3 **while** *Optimierungsbudget ist nicht überschritten* **do**
4 Passe das Surrogatmodell auf $\mathscr{D}$ und $\boldsymbol{y}$ an;
5 Bestimme den nächsten Punkt $\boldsymbol{x}^*$ mittels der Optimierung des Zielkriteriums;
6 Evaluiere die Zielfunktion auf dem neuen Punkt $y^* = f(\boldsymbol{x}^*)$;
7 Aktualisiere Design und Zielgröße: $\mathscr{D} \leftarrow (\mathscr{D}, \boldsymbol{x}^*)^T$ und $\boldsymbol{y} \leftarrow (\boldsymbol{y}, y^*)^T$;
8 **end**
9 **return** $y_{min} = \min(\boldsymbol{y})$ und der zugehörige $\boldsymbol{x}_{min}$.

6.2.1 Startdesign und Optimierungsbudget

Eine der wichtigsten Überlegungen ist die Höhe des Optimierungsbudgets und die Proportion zwischen dem Startdesign und den sequentiellen Schritten. Das Budget hängt naturgemäß von der Dimension der zu optimierenden Funktion ab (Anzahl der Funktionsparameter d) und ist ein Trade-off zwischen einem guten Optimierungsergebnis und dem damit verbundenen Zeitaufwand. Bei der Wahl der Startdesigngröße sollte berücksichtigt werden, dass zu kleine Designs die Form der Zielfunktion nicht ausreichend „abtasten“ und somit zu einer lokalen Konvergenz des Verfahrens führen können. Je nach Wahl des Zielkriteriums kann dieser Effekt weniger oder stärker ausgeprägt sein. Wird dagegen ein zu großer Anteil an Gesamtauswertungen dem Startdesign zugeordnet, bleiben nur wenige Auswertung für die sequentiellen Schritte übrig, so dass die modellbasierte Suche benachteiligt wird.

Eine weitere wichtige Überlegung ist die Zusammensetzung des Startdesigns. In der klassischen Versuchsplanung sind vollfaktorielle, orthogonale bzw. zentral zusammengesetzte Pläne[3] sehr verbreitet. Diese eignen sich besonders gut dazu, den Effekt der einzelnen Parameter[4] bzw. deren Wechselwirkungen oder der quadratischen Terme in einem linearen Modell zu schätzen. Sehr populär sind D-optimale Versuchspläne, die die Determinante der Designmatrix maximieren (Spall, 2003, S. 472). Der Nachteil solcher Pläne ist unter anderem die fehlende Flexibilität in Bezug auf ihre Größe. So ergeben sich z.B. für einen zentral-zusammengesetzten Plan mit d Parameter exakt $d^2 + 2d + 1$ Punkte. Zwar erleichtert das R-Paket **DoE.base** (Groemping, 2014)

[3] Hier sind die Begriffe Plan und Design als Synonyme zu betrachten.

[4] In der Terminologie der Versuchsplanung werden die Funktionsparameter *Faktoren* genannt.

das Finden orthogonaler Pläne für beliebig viele Parameter mit verschiedener Anzahl an kodierten Niveaus, eine genaue Festlegung der Designgröße ist allerdings oft unmöglich. In den meisten Arbeiten zur computergestützten Optimierung werden daher *Latin Hypercube Sampling* Designs (LHS, McKay et al., 1979) verwendet, die den interessierenden Optimierungsbereich gleichmäßig abdecken, wobei deren Größe frei einstellbar ist. In R sind verschiedene Typen von LHS in dem **lhs** R-Paket (Carnell, 2012) implementiert. In dieser Arbeit werden sämtliche LHS Designs mit Hilfe der *maximinLHS* Funktion generiert, deren Ziel ist es, die minimale Distanz zwischen den Designpunkten zu maximieren.

Wenden wir uns nun dem Verhältnis zwischen der Größe des Startdesigns und der Anzahl der sequentiellen Schritte zu. Picheny et al. (2013) untersuchen je zwei Einstellungen des Optimierungsbudgets ($20d$ und $40d$) und der Größe des Startdesigns ($4d$ und $10d$) für sechs mathematische Funktionen der Dimension zwischen zwei und sechs (wie *Branin*[5] oder *Hartmann*[6] Funktionen). Die Experimente zeigen, dass die größeren Startdesigns die Optimierungsergebnisse bei einigen Funktionen nur sehr leicht verbessern, während die größeren Optimierungsbudgets erwartungsgemäß wesentlich besser abschneiden als die kleineren. Die Autoren sprechen eine Empfehlung aus, mehr Auswertungen in die sequentiellen Schritte einzuplanen und somit eher kleinere Startdesigns zu wählen.

In Bauer et al. (2013a) werden die klassische und die sequentielle Versuchsplanung auf dem Problem der Einsatzzeiterkennung verglichen[7]. Als klassischer Vertreter wird ein vollfaktorieller Versuchsplan für drei Parameter mit jeweils drei Niveaus und zusätzlichen Punkten in dem mittleren Bereich (insgesamt 33 Punkte, also $11d$) ausgewählt. Bei der sequentiellen Optimierung besteht das Startdesign aus 15 Punkten ($5d$), wobei zwischen dem zentral-zusammengesetzten und dem LHS Design unterschieden wird. Es wird zum einen gezeigt, dass die klassische Optimierung der sequentiellen deutlich unterlegen ist, und zum anderen, dass der zentral-zusammengesetzte Plan bessere Ergebnisse liefert als das LHS Design.

Die Fragestellung von Bauer et al. (2013a) wird in Bauer et al. (2013b) in einer erweiterten Form weiter verfolgt. Es wird ein weiterer Parameter der Einsatzzeiterkennung eingeführt, so dass das klassische Design nun aus $3^4 = 81$ Punkten besteht und das Optimierungsbudget somit auf 82 Punkte (ca. $20d$) beschränkt wird. Außerdem wird der Einfluss der Größe und der Zusammensetzung des Startdesigns auf die Güte des Optimierungsergebnisses untersucht. Ein orthogonales Design mit 48 Punkten ($12d$) und das zental-zusammengesetze Design mit innerem Stern (25 Punkte, ca. $6d$) entsprechen jeweils ca. der Hälfte und einem Drittel der Gesamtauswertungen. Für

[5] `http://www.sfu.ca/~ssurjano/branin.html`, Stand: 01.05.2015.

[6] `http://www.sfu.ca/~ssurjano/hart6.html`, Stand: 01.05.2015.

[7] Dabei wurde ein sehr einfaches Algorithmus zur Einsatzzeiterkennung mit nur drei Einflussparametern verwendet.

alle drei genannten klassischen „Lehrbuch“-Pläne werden auch LHS Designs der gleichen Größe betrachtet. Es stellt sich heraus, dass die Optimierung mit LHS Startdesigns bessere y-Werte[8] aufweist, wobei diese auch laut dem Optimierungsverlauf schneller gefunden werden als bei den klassischen Varianten. Weiterhin zeigen LHS Designs mit ca. der Hälfte des Optimierungsbudgets leicht bessere Ergebnisse als die mit ca. einem Drittel[9]. Allerdings sind für die größeren Startdesign insgesamt im Durchschnitt 5 Funktionsauswertungen mehr notwendig, um einen akzeptabel guten y-Wert zu erreichen.

Aufgrund der Komplexität der hier zu optimierenden Anwendung und damit verbundenem hohen Zeitbedarf für die Funktionsauswertungen werden die Größe des Startdesigns auf $5d$ und die Anzahl der sequentiellen Schritte auf $20d$ festgelegt.

6.2.2 Surrogatmodell

Die Wahl des Surrogatmodells ist für die Güte der Optimierung von entscheidender Bedeutung, wobei theoretisch ein beliebiges Regressionsmodell dafür verwendet werden kann. Sasena (2002) fasst einige Surrogatmodelle zusammen, die bereits eine Anwendung in der Optimierung gefunden haben, wie zum Beispiel Spline-Interpolation, Fuzzy-Logik oder Wiener Prozesse. Am populärsten ist allerdings das von Krige (1951) vorgeschlagene Verfahren (das sogenannte Kriging Modell), welches die hochdimensionale multimodale Funktionslandschaft bereits beim Vorliegen von nur wenigen Punkten mit einer ausreichend guten Qualität modellieren kann. Historisch gesehen wurde das Modell für die Vorhersage der Messwerte von Bodenproben an einem beliebigen Ort (d.h. Interpolation) aufgrund der vorhandenen Messwerte aus den benachbarten Bohrungen entwickelt. Entscheidend für den Erfolg dieses Modells ist die Berücksichtigung der räumlichen Abhängigkeit zwischen den Messpunkten.

Bevor im Weiteren das Kriging Modell ausführlich vorgestellt wird, soll hier die Möglichkeit simultaner Verwendung mehrerer Surrogatmodelle erwähnt werden. Die Bildung der Ensembles aus mehreren Modellen bzw. Lernverfahren ist vor allem im Gebiet der überwachten Klassifikation besonders verbreitet (van der Laan et al., 2008). Hess et al. (2013) verwenden mehrere Surrogatmodelle für MBO, wobei in jeder Iteration ein Modell anhand einer Wahrscheinlichkeitsverteilung zufällig gezogen wird. Je effektiver ein Modell in den vorangegangenen Schritten gewesen ist, d.h. je größere Verbesserung der Zielfunktionswerte es bewirken konnte, desto höher ist die Chance, es in der nächsten Iteration erneut zu ziehen. In Bauer et al. (2013a) werden vier Möglichkeiten der Modellkombination für MBO vorgeschlagen, wobei in jeder Iteration alle Surrogatmodelle aus dem Ensemble angepasst und deren Vorhersagen auf verschiedene Weisen miteinander verknüpft

[8] y ist hier gleich dem in Abschnitt 3.9.1 definierten F-Wert.

[9] Diese Tatsache lässt sich vermutlich dadurch erklären, dass in Bauer et al. (2013b) in den meisten Fällen die Vorhersage des Surrogatmodells als Zielkriterium verwendet wurde. Dieses Kriterium kann allerdings schnell zu einer lokalen Konvergenz führen, so dass die größeren Startdesigns die Funktionslandschaft vor dem Beginn der sequentiellen Schritte besser erfasst haben.

werden. Sowohl in Hess et al. (2013) als auch in Bauer et al. (2013a) konnte die Überlegenheit des Kriging Modells beobachtet werden, so dass die Kombination mehrerer Modelle (darunter Kriging) nur selten besser abgeschnitten hat als das einzelne Kriging Modell.

Definition des Kriging Modells. Das universelle Kriging Modell besteht aus zwei Termen, einem polynomiellen Term und einem Fehlerterm:

$$Y(\boldsymbol{x}) = \sum_{j=1}^{k} \beta_j f_j(\boldsymbol{x}) + Z(\boldsymbol{x}). \tag{6.1}$$

Dabei sind f_j die polynomiellen Terme mit den zugehörigen Parametern β_j. $Y(\boldsymbol{x})$ ist eine Zufallsvariable und der Fehlerterm $Z(\boldsymbol{x})$ ist ein Gauss-Prozess, welcher die Unsicherheit in dem Mittelwert von $Y(\boldsymbol{x})$ ausdrückt. Sehr oft wird aber das sogenannte „gewöhnliche“ Kriging Modell (engl. *ordinary Kriging*) verwendet, wobei der erste Summand $\mu \in \mathbb{R}$ ein unbekannter konstanter Trend ist (Picheny et al., 2013):

$$Y(\boldsymbol{x}) = \mu + Z(\boldsymbol{x}). \tag{6.2}$$

In den von Sacks et al. (1989) durchgeführten Experimenten hat die oben vorgestellte Vereinfachung zu keinem signifikanten Verlust der Modellvorhersagegüte geführt. Falls dagegen μ bekannt ist, handelt es sich um das einfache Kriging Modell (engl. *simple Kriging*).

Die wichtigsten Eigenschaften von $Z(\boldsymbol{x})$ sind:

$$\begin{aligned} &\mathbb{E}(Z(\boldsymbol{x})) = 0, \\ &K(\boldsymbol{x},\tilde{\boldsymbol{x}}) = Cov(Z(\boldsymbol{x}), Z(\tilde{\boldsymbol{x}})) = \sigma_z^2 k(\boldsymbol{x},\tilde{\boldsymbol{x}}), \forall \boldsymbol{x},\tilde{\boldsymbol{x}}, \end{aligned} \tag{6.3}$$

wobei $K(\boldsymbol{x},\tilde{\boldsymbol{x}})$ der sogenannte Kovarianz-Kernel, $k(\boldsymbol{x},\tilde{\boldsymbol{x}})$ die räumliche Korrelationsfunktion (engl. *spatial correlation function*) und σ_z^2 ein Parameter der Prozessvarianz sind (Sasena, 2002; Schonlau, 1997). Die Korrelationsfunktion misst die Ähnlichkeit zwischen den Datenpunkten und ist laut Rasmussen und Williams (2006) (S. 79) der entscheidende Bestandteil von Gauss-Prozess-Vorhersagen.

Für die multidimensionalen Daten berechnet sich die Korrealtionsfunktion als Produkt der eindimensionalen Funktionen $k_j(x_j,\tilde{x}_j)$ (engl. *product correlation rule*):

$$k(\boldsymbol{x},\tilde{\boldsymbol{x}}) = \prod_{j=1}^{d} k_j(x_j,\tilde{x}_j). \tag{6.4}$$

Zu bemerken ist, dass $K(\boldsymbol{x},\boldsymbol{x}) = 1$ ist, d.h. dass die Vorhersage für einen bereits evaluierten Punkt nicht von dem Zielfunktionswert in diesem Punkt abweichen wird. Dies kann allerdings für einige Anwendungen kritisch sein, insbesondere wenn die Zielfunktion verrauscht ist.

Korrelationsfunktion. Viele verschiedene Korrelationsfunktionen (KFen) wurden in den letzten Jahrzehnten vorgeschlagen, welche sich grob in drei Gruppen unterteilen lassen: stationäre,

multiplikative (engl. *dot product*) und andere nicht stationäre Korrelationsfunktionen (Rasmussen und Williams, 2006, S. 79 ff.). Sehr verbreitet sind allerdings die (stationären) Klassen der Gauss- und Matérn-Korrelationsfunktionen. Die Gauss-KF (oder auch die quadrierte exponentielle KF genannt) ist definiert durch:

$$k_j(x_j,\tilde{x}_j) = \exp\left(-\frac{|x_j-\tilde{x}_j|^2}{2\theta_j^2}\right). \tag{6.5}$$

Je größer die absolute Distanz zwischen x_j und $\tilde{x}_j$ ist, desto kleiner ist der Wert von $k_j(x_j,\tilde{x}_j)$. Der Einfluss von bereits evaluierten Punkten auf die Vorhersage des neuen Punktes sinkt also mit der wachsender Distanz zwischen denen. Der Parameter θ_j regelt dabei wie stark dieser Einfluss nachlässt: je größer θ_j desto größer ist der Einflussbereich. Rasmussen und Williams (2006) (S. 83) bemerken, dass Gauss-KFen wegen ihrer unendlichen Differenzierbarkeit zu sehr glatten Gauss-Prozessen führen können. Weiterhin zitieren die Autoren die Arbeit von Stein (1991), der diese Annahme der starken Glattheit als unrealistisch für die Modellierung vieler physikalischer Prozesse ansieht und deswegen die Klasse von Matérn-KFen empfiehlt.

Die Matérn-KF wurde von Matérn (1986) eingeführt:

$$k_j^{\nu}(x_j,\tilde{x}_j) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}|x_j-\tilde{x}_j|}{\theta_j}\right)^{\nu} H_{\nu}\left(\frac{\sqrt{2\nu}|x_j-\tilde{x}_j|}{\theta_j}\right). \tag{6.6}$$

ν und θ_j sind positive Konstanten, H_{ν} ist die modifizierte Bessel-Funktion (Abramowitz und Stegun, 1965, S. 375) und $\Gamma(\nu)$ die Gamma-Funktion. Für $\nu\to\infty$ ist die Matérn-KF gleich der Gauss-KF. Für den Fall $\nu = p+\frac{1}{2}$ lässt sich die Matérn-KF als Produkt von exponentiellen und polynomiellen Funktionen aufschreiben (Rasmussen und Williams, 2006, S. 85):

$$k_j^{\nu=p+1/2}(x_j,\tilde{x}_j) = \exp\left(\frac{\sqrt{2\nu}|x_j-\tilde{x}_j|}{\theta_j}\right)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\left(\frac{\sqrt{8\nu}|x_j-\tilde{x}_j|}{\theta_j}\right)^{(p-i)}. \tag{6.7}$$

Laut Rasmussen und Williams (2006) (S. 85) ist von der Wahl $\nu = 1/2$ abzuraten, da der Prozess zu rau werde. Für $\nu \geq 7/2$ werde es dagegen schwer, einen Unterschied zwischen den zugehörigen Matérn-KFen festzustellen, da sie in diesem Bereich der Gauss-KF bereits sehr ähnlich sind. Sehr verbreitet sind daher $\nu = 3/2$ und $\nu = 5/2$ KFen.

Für die meisten in Picheny et al. (2013) behandelten mathematischen Funktionen konnte kein relevanter Unterschied zwischen der Gauss-KF und der Matérn-KF mit $\nu = 3/2$ festgestellt werden. In dieser Arbeit wird die Matérn-KF mit $\nu = 3/2$ verwendet:

$$k_j^{\nu=3/2}(x_j,\tilde{x}_j) = \left(1+\frac{\sqrt{3}|x_j-\tilde{x}_j|}{\theta_j}\right)\exp\left(-\frac{\sqrt{3}|x_j-\tilde{x}_j|}{\theta_j}\right). \tag{6.8}$$

Die KF-Parameter $(\theta_1, \ldots, \theta_d)^T$ sowie die Prozessvarianz σ_z^2 werden mit Hilfe der Maximum-Likelihood-Methode geschätzt. Das Vorgehen wird in Picheny et al. (2013) detailliert erläutert. Im folgenden bezeichnet $\boldsymbol{K}_{\mathscr{D}}(\boldsymbol{x})$ den Vektor von Kovarianz-Kernel zwischen einem Punkt $\boldsymbol{x}$ und n bereits evaluierten Punkten des Designs $\mathscr{D}$:

$$\boldsymbol{K}_{\mathscr{D}}(\boldsymbol{x}) = (K(\boldsymbol{x},\boldsymbol{x}_1), \ldots, K(\boldsymbol{x},\boldsymbol{x}_n))^T \tag{6.9}$$

und $\mathscr{K}_{\mathscr{D}}$ die Kovarianzmatrix des Designs $\mathscr{D}$:

$$\mathscr{K}_{\mathscr{D}} = (K(\boldsymbol{x}_i,\boldsymbol{x}_j))_{1\leq i,j\leq n}. \tag{6.10}$$

Parameterschätzer. Die Definition und Herleitung der Parameterschätzer basiert hier auf Picheny et al. (2013). Sei ein Design $\mathscr{D} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^T$ mit $Y(\boldsymbol{x}_i) = y_i$, $i = 1, \ldots, n$, gegeben. Der Erwartungswert und die Varianz von Y für alle anderen Punkte des Parameterraums können nun mit Hilfe der Bedingung von $Y(\boldsymbol{x})$ auf $\boldsymbol{y}$ berechnet werden:

$$\begin{aligned} \hat{f}(\boldsymbol{x})_{\mathscr{D}} &= \mathbb{E}\left[Y(\boldsymbol{x})|\boldsymbol{y}\right], \\ \hat{s}^2(\boldsymbol{x})_{\mathscr{D}} &= Var\left[Y(\boldsymbol{x})|\boldsymbol{y}\right]. \end{aligned} \tag{6.11}$$

$\hat{f}(\boldsymbol{x})_{\mathscr{D}}$ wird auch Kriging Mittelwert und $\hat{s}(\boldsymbol{x})^2_{\mathscr{D}}$ Kriging Varianz genannt. Die Indizierung mit $\mathscr{D}$ soll verdeutlichen, dass die beiden Funktionen von der Designauswertung abhängen. $\hat{s}(\boldsymbol{x})^2_{\mathscr{D}}$ gibt die Unsicherheit des Modells im Punkt $\boldsymbol{x}$ wieder. Diese Unsicherheit basiert lediglich auf der Entfernung des Punktes $\boldsymbol{x}$ von den bereits evaluierten Punkten. Nicht zu verwechseln ist sie deswegen mit einer möglichen Unsicherheit einer Zielfunktion in bestimmten Bereichen, wo selbst für sehr ähnliche Punkte sehr unterschiedliche Zielfunktionswerte ermittelt werden.

Bezeichnet man mit $\hat{\mu}_{\mathscr{D}}$ den besten linearen unverzerrten Schätzer für μ mit

$$\hat{\mu}_{\mathscr{D}} = \frac{\mathbf{1}_n^T \mathscr{K}_{\mathscr{D}}^{-1} \boldsymbol{y}}{\mathbf{1}_n^T \mathscr{K}_{\mathscr{D}}^{-1} \mathbf{1}_n}, \tag{6.12}$$

wobei $\mathbf{1}_n^T$ der n-dimensionale Einsvektor ist, so können die besten linearen unverzerrten Schätzer für den Erwartungswert und die Varianz wie folgt aufgeschrieben werden (Picheny et al., 2013):

$$\begin{aligned} \hat{f}(\boldsymbol{x})_{\mathscr{D}} &= \hat{\mu}_{\mathscr{D}} + \boldsymbol{K}_{\mathscr{D}}(\boldsymbol{x})^T \mathscr{K}_{\mathscr{D}}^{-1} \left(\boldsymbol{y} - \hat{\mu}_{\mathscr{D}} \mathbf{1}_n\right), \\ \hat{s}^2(\boldsymbol{x})_{\mathscr{D}} &= \sigma_z^2 - \boldsymbol{K}_{\mathscr{D}}(\boldsymbol{x})^T \mathscr{K}_{\mathscr{D}}^{-1} \boldsymbol{K}_{\mathscr{D}}(\boldsymbol{x}) + \frac{\left(1 - \mathbf{1}_n^T \mathscr{K}_{\mathscr{D}}^{-1} \boldsymbol{K}_{\mathscr{D}}(\boldsymbol{x})\right)^2}{\mathbf{1}_n^T \mathscr{K}_{\mathscr{D}}^{-1} \mathbf{1}_n}. \end{aligned} \tag{6.13}$$

Das Kriging Modell ist in seiner ursprünglichen Definition nur für die Optimierung der deterministischen Funktionen geeignet. So schlägt die Parameterschätzung fehl, falls zwei fast identische Punkte im Design existieren. Einen Ausweg bietet die Möglichkeit, beim Modellieren den sogenannten *nugget* Effekt (das Rauschen) mit zu schätzen (Sasena, 2002; Morgan, 2011).

Dummy Kriging vs. naives Kriging. Einer der bedeutendsten Nachteile des Kriging Modells ist seine Beschränkung auf numerische Einflussparameter. Sollen für eine Anwendung auch kategorielle Parameter optimiert werden, bestehen mehrere Möglichkeiten, Kriging dennoch verwenden zu können.

Die einfachste und naive Variante besteht darin, kategorielle Parameter als stetige zu behandeln. In dem Fall der univariaten Einsatzzeiterkennung (s. Tabelle 3.2) wird beispielsweise der Parameter 'Funktion zur Einsatzzeiterkennung' nicht mehr 18 verschiedene diskrete Ausprägungen haben, sondern als eine stetige Größe auf dem Intervall $[1, 18]$ angesehen. Innerhalb der Zielfunktion wird der stetige Wert solcher Parameter gerundet und eine Zuordnung zu dem wahren Parameterwert vorgenommen (z.B. 1 = '*Ampl.Max.Diff*'). Dieses Vorgehen ist aus statistischer Sicht nicht korrekt, weil dadurch sowohl eine Ordnung als auch Abstände zwischen den Ausprägungen definiert werden, die es eigentlich nicht gibt.

Die statistisch korrektere Lösung dieses Problem ist die Verwendung der Dummy-Variablen. Für den oben genannten Parameter würde dies bedeuten, dass er durch 18 Variablen dargestellt wird: $Ampl.Max.Diff, \ldots, Rect.Compl.Domain$. Pro Designpunkt kann nur eine dieser Variablen den Wert 1 haben (die restlichen haben dann den Wert 0). Der Nachteil dieses Vorgehens ist allerdings die höhere Zeit für die Modellanpassung, welche von der Anzahl der Parameter abhängt. Außerdem ist das Kriging Modell nicht für solche Typen von Variablen (mit nur zwei Ausprägungen) konzipiert, so dass hiermit die Modellannahmen verletzt werden.

In dem experimentellen Teil dieser Arbeit werden die beiden Vorgehensweisen – naives Kriging und dummy Kriging – auf das Einsatzzeiterkennungsproblem angewendet und verglichen.

6.2.3 Zielkriterium

Der gängige englische Begriff für das Zielkriterium ist *infill criterion*. Das Zielkriterium gibt die gewünschte Eigenschaft wieder, die für die Auswahl des nächsten Punktes maßgeblich ist. Ein sehr intuitives Zielkriterium ist die Modellvorhersage: die neue Auswertung sollte in dem Punkt mit der kleinsten (im Falle einer Minimierung) Modellvorhersage stattfinden. Der Nachteil dieses Vorgehens ist allerdings die Gefahr der Konvergenz zu einem lokalen Optimum. Ist jedoch das Ziel der Optimierung die Funktionslandschaft möglichst gut zu erforschen, kann die Modellunsicherheit (im Sinne von Abstand zu bereits evaluierten Punkten, $\hat{s}^2(\boldsymbol{x})_{\mathscr{D}}$) ein empfehlenswertes Zielkriterium sein.

Das in Jones et al. (1998) vorgeschlagene Kriterium der erwarteten Verbesserung (engl. *expected improvement*, EI) ist ein gelungener Kompromiss zwischen den beiden genannten Strategien.

EI ist ein Trade-off zwischen „Verbesserung der besten Lösung“ und „Erforschung der Funktionslandschaft“ (engl. *exploitation* vs. *exploration*). Das EI-Kriterium gewährleistet die globale Konvergenz des sequentiellen MBO Verfahrens, d.h. theoretisch kann mit einem unendlichen Budget das wahre Optimum beliebig gut angenähert werden (Jones, 2001; Vazquez und Bect, 2010), und ist zum Standardkriterium für viele Anwendungen geworden. Aus den vorigen Überlegungen ist ersichtlich, dass $Y(\boldsymbol{x})|\boldsymbol{y}$ der univariaten Normalverteilung $N(\hat{f}(\boldsymbol{x})_{\mathscr{D}}, \hat{s}^2(\boldsymbol{x})_{\mathscr{D}})$ folgt. Die erwartete Verbesserung von einem Punkt $\boldsymbol{x}$ ist als Erwartungswert der positiven Verbesserung der Zielfunktion in diesem Punkt definiert (Ginsbourger, 2009):

$$\begin{aligned} \mathrm{EI}(\boldsymbol{x}) &= \mathbb{E}[\max\{0, y_{min} - \hat{f}(\boldsymbol{x})_{\mathscr{D}}\}] \\ &= (y_{min} - \hat{f}(\boldsymbol{x})_{\mathscr{D}})\ \Phi\left(\frac{y_{min} - \hat{f}(\boldsymbol{x})_{\mathscr{D}}}{\hat{s}(\boldsymbol{x})_{\mathscr{D}}}\right) + \hat{s}(\boldsymbol{x})_{\mathscr{D}}\ \phi\left(\frac{y_{min} - \hat{f}(\boldsymbol{x})_{\mathscr{D}}}{\hat{s}(\boldsymbol{x})_{\mathscr{D}}}\right), \end{aligned} \tag{6.14}$$

wobei ϕ and Φ die Dichte und die Verteilungsfunktion der Standardnormalverteilung sind. y_{min} bezeichnet den minimalen bisher erreichten Wert der Zielfunktion. Da die erwartete Verbesserung maximiert werden sollte, sind wir an einer Lösung $\boldsymbol{x}^*$ interessiert mit

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathscr{X}}{\arg\max}\ \mathrm{EI}(\boldsymbol{x}). \tag{6.15}$$

Abbildung 6.1 veranschaulicht die Kriging basierte Optimierung mit Hilfe des EI Kriteriums. Dabei wird die eindimensionale Zielfunktion $y = \sin(2x) + 0.5\cos(x)$ auf dem Intervall $x \in [-1, 8]$ minimiert. Es sind von oben links bis unten rechts die ersten vier Iterationen abgebildet. Pro Iteration werden sowohl die Zielfunktion (durchgehende Linie), das Surrogatmodell (gestrichelte Linie) und die Modellunsicherheit (dunkel graue Fläche) in dem oberen Plot als auch das EI-Kriterium (gepunktete Linie) und dessen maximaler Wert (vertikale gestrichelte Linie) in dem unterem Plot dargestellt. Das Startdesign bestehend aus 5 Auswertungen wird mit roten Punkten markiert, bereits evaluierte Punkte der sequentiellen Optimierung sind grün und der als nächstes vorgeschlagene Punkt (also der mit dem maximalen EI-Wert) ist blau. Außerdem wird noch die Differenz zwischen dem aktuell kleinsten Wert und dem wahren minimalen Wert angezeigt (als *Gap*, s. die Überschrift der Abbildungen). Die Abbildungen wurden mit Hilfe der *plotExampleRun* Funktion aus dem **mlrMBO** R-Paket erzeugt.

In Abbildung 6.1 ist zu sehen, dass in den ersten beiden Iterationen die erwartete Verbesserung in den Randbereichen des zulässigen Intervalls maximal ist. Zum einen liegt dies daran, dass die beiden Optima relativ nah an den Rändern liegen und zum anderen scheint die Modellunsicherheit in den „unerforschten“ Bereichen an den Rändern größer zu sein als in solchen in der Mitte. Bereits in der dritten Iteration wird das rechte Minimum fast genau getroffen und in der vierten Iteration wird der Vorteil des explorativen Vorgehens von EI verdeutlicht: es wird ein Punkt in der Nähe des linken Minimums vorgeschlagen.

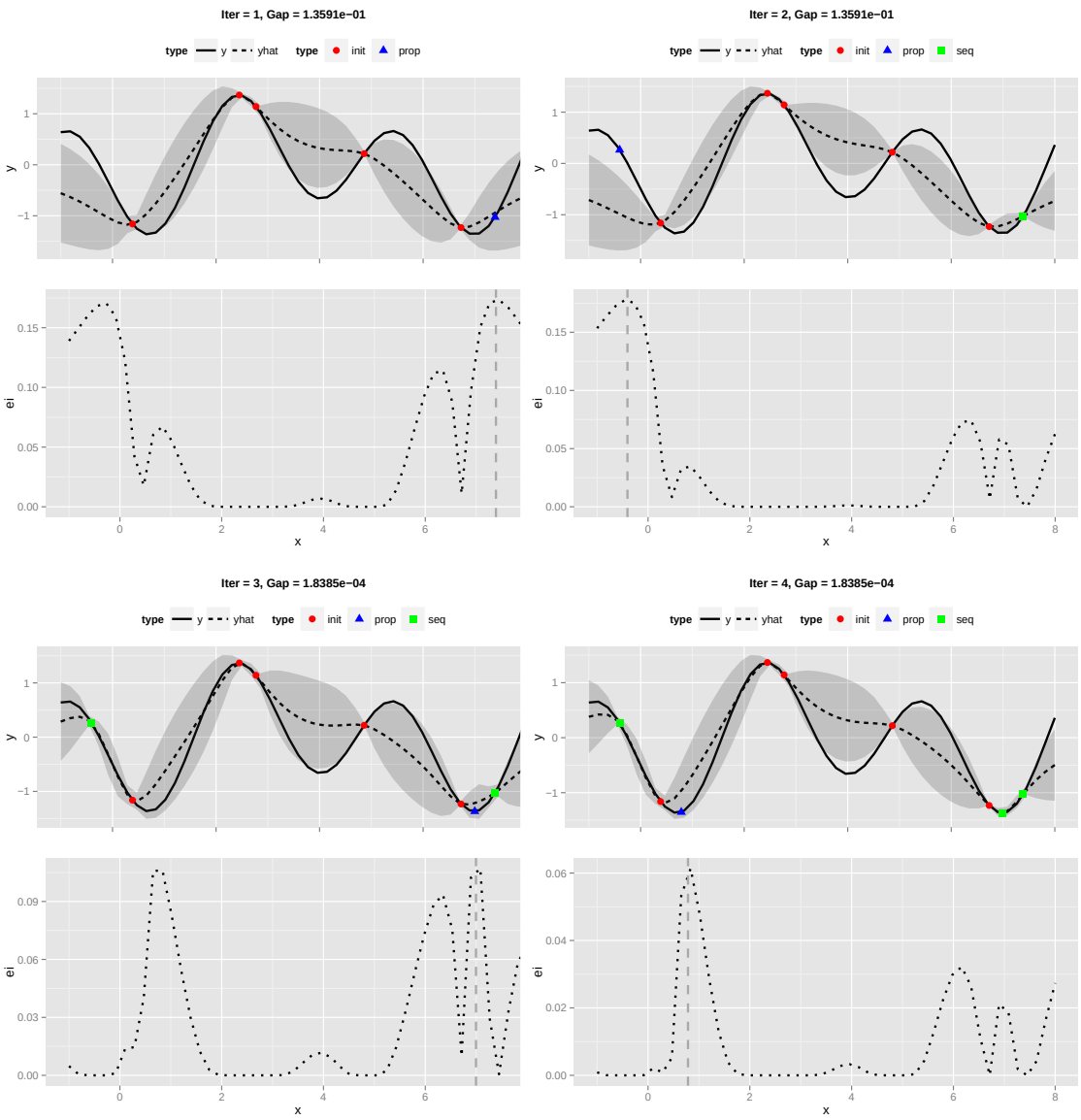

Abbildung 6.1: Die ersten vier Schritte der MBO mit Kriging Surrogatmodell und EI Zielkriterium bei der Minimierung der $y = \sin(2x) + 0.5\cos(x)$ Funktion.

Eine weitere Möglichkeit, Modellvorhersage und -unsicherheit miteinander zu kombinieren, bietet das sogenannte *Lower Confidence Bound* (LCB) Kriterium, welches minimiert werden soll:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathscr{X}}{\arg\min}\, \mathrm{LCB}(\boldsymbol{x}) = \underset{\boldsymbol{x} \in \mathscr{X}}{\arg\min}\, \left(\hat{f}(\boldsymbol{x})_{\mathscr{D}} - \vartheta \hat{s}(\boldsymbol{x})_{\mathscr{D}}\right). \tag{6.16}$$

Der Gewichtsparameter $\vartheta > 0$ regelt dabei das Verhältnis zwischen Ausnutzung (ϑ klein) und Erforschung (ϑ groß). Andere Zielkriterien werden beispielsweise bei Jones (2001) diskutiert. In der bereits oben erwähnten Arbeit von Bauer et al. (2013b) wird zwar $\hat{f}(\boldsymbol{x})_{\mathscr{D}}$ als Zielkriterium verwendet, aber um die Exploration des Parameterraums dennoch nicht zu benachteiligen, ist ein zufallsbedingter Mechanismus eingebaut worden: Mit einer benutzerdefinierten Wahrscheinlichkeit wird in jeder Iteration der nächste Punkt nicht laut der Modellvorhersage ausgewählt, sondern laut seinem zu maximierenden Abstand zu bereits evaluierten Punkten.

An dieser Stelle sei noch erwähnt, dass die Modellunsicherheit nicht nur über die Distanz definiert werden kann. So benutzen Hutter et al. (2011) den Zufallswald (engl. *random forest*) als Surrogatmodell, um das Problem der kategoriellen Parameter im Kriging Modell umzugehen. Dabei definieren sie $\hat{f}(\boldsymbol{x})_{\mathscr{D}}$ and $\hat{s}(\boldsymbol{x})^2_{\mathscr{D}}$ als Mittelwert und Varianz der Vorhersagen der einzelner Bäume.

Auch bei der Wahl des Zielkriteriums soll berücksichtigt werden, ob die Zielfunktion deterministisch oder verrauscht ist. Einen Übersicht über die Zielkriterien für verrauschte Zielgrößen sowie deren Vergleich bietet die Arbeit von Picheny et al. (2013). In dieser Arbeit wird für die Optimierung das EI-Kriterium verwendet.

6.2.4 Optimierung des Zielkriteriums

Wie in Zeile 5 des Algorithmus 6.1 angedeutet ist, soll in jeder MBO-Iteration ein Punkt ausgewählt werden, der das gewünschte Zielkriterium maximiert bzw. minimiert. Dies stellt ein nichtlineares Optimierungsproblem dar, welches auf verschiedenen Wegen gelöst werden kann. In **DiceOptim** R-Paket (Ginsbourger et al., 2013) wird die EI Optimierung mit dem **rgenoud** R-Paket (*GENetic Optimization Using Derivatives*, Sekhon und Mebane, Jr., 1998) realisiert. Wie die meisten genetischen Algorithmen kann dieser Algorithmus nur stetige Parameter optimieren. In Kombination mit dem Kriging Surrogatmodell stellt dies allerdings keine Einschränkung dar.

Bei dem SMAC-Verfahren (*Sequential Model-based optimization for general Algorithm Configuration*) von Hutter et al. (2011) sind die kategoriellen Einflussparameter nicht ausgeschlossen und das EI wird folgendermaßen optimiert: In dem ersten Schritt werden EI-Werte der bereits evaluierten Punkte erneut gerechnet (da $\hat{s}(\boldsymbol{x})^2_{\mathscr{D}}$ nicht kernelbasiert berechnet wird, ist EI von bereits evaluierten Punkten ungleich Null) und die 10 besten Punkte werden als Startpunkte für den nachfolgenden einfachen Suchalgorithmus verwendet, in dem pro Iteration eine Parametereinstellung zufällig variiert wird (engl. *one-exchange neighborhood search*). Das EI-Kriterium wird anschließend sowohl für die Ausgabe des Suchverfahrens als auch für weitere 10 000 zufällig gewählte Punkte berechnet. Anschließend wird ein Set der p Punkte mit den größten EI-Werten als Ergebnis der Zieloptimierung zurückgegeben. Dieses Vorgehen stellt keine konventionelle bzw. intuitive Optimierungsmethode dar, sondern einen eher sehr speziellen heuristischen Ansatz.

Das Paket **mlrMBO** wurde entwickelt, um eine große Flexibilität bei der Optimierung verschiedener komplexer Probleme zu bieten. Es wird davon ausgegangen, dass die Einflussparameter stetig oder kategorial sein können und dass das Surrogatmodell auch mit solchen nicht stetigen Parametern gut angepasst werden kann (z.B. Zufallswald). Aus diesem Grund wurde eine Optimierungsmethode des Zielkriteriums vorgeschlagen – die fokussierte Suche (engl *focus search*) –, die von der Parameterskalierung unabhängig ist. Die grundlegende Idee besteht darin, den Parameterraum sukzessiv auf interessante Bereiche zu fokussieren. Dabei wird im ersten Schritt ein LHS Design der Größe $N_{FS.points}$ auf $\mathscr{X}$ generiert und für jeden Punkt das gewünschte Zielkriterium berechnet[10]. Anschließend wird der beste Punkt ausgewählt und der Parameterraum rund um diesen Punk für jede Dimension reduziert.

Für eine stetige Variable mit zulässigem Bereich $[0,1]$ und dem besten Punkt genau in der Mitte dieses Intervalls würde das zulässige Bereich z.B. auf $[0.25,0.75]$ reduziert werden. Bei kategoriellen Variablen mit mehr als zwei Ausprägungen wird die Ausprägung des besten Punktes beibehalten und zufällig eine der übrigen Ausprägungen entfernt (d.h. in der nächsten Iteration nicht mehr berücksichtigt). Das Fokussierungsvorgehen wird $N_{FS.maxit}$ mal durchgeführt. Da die fokussierte Suche jedoch nur lokal konvergieren kann, sollte sie mehrmals wiederholt werden. Der Parameter $N_{FS.restarts}$ gibt die Anzahl der Wiederholungen wider. Als $\boldsymbol{x}^*$ wird dann der über alle Iterationen und Wiederholungen beste gefundene Punkt zurückgegeben. Für diese Arbeit wurden folgende Einstellungen verwendet: $N_{FS.points} = 10\,000$, $N_{FS.maxit} = 5$ und $N_{FS.restarts} = 3$.

6.2.5 MBO Erweiterungen

Das in Algorithmus 6.1 vorgestellte Schema der modellbasierten Optimierung kann in mehrerer Hinsicht erweitert werden. Zum einen können mehrere Zielgrößen vorliegen, die gleichzeitig zu optimieren sind. In **mlrMBO** R-Paket sind dafür mehrere „State of the Art“ Algorithmen implementiert (z.B. Knowles, 2006; Wagner et al., 2010), wobei an alternativen Vorgehensweisen – angesichts der hohen Relevanz der multikriteriellen Optimierung – zurzeit intensiv geforscht wird (Horn et al., 2015).

Ein weiterer wichtiger Aspekt ist die Möglichkeit, q Punkte pro Iteration gleichzeitig auszuwerten. Technisch ist dies dank der modernen Mehrkernprozessor-Architektur ohne Probleme möglich. Die Herausforderung liegt in der sinnvollen Auswahl von q Punkten. In Ginsbourger et al. (2010) and Janusevskis et al. (2012) wird das q-Punkte EI-Kriterium (bzw. q-EI) aufgestellt, welches die erwartete Verbesserung für $q = 2$ Punkte exakt bestimmt und für $q > 2$ mithilfe einer aufwendigen Monte-Carlo Simulationen schätzt. Chevalier und Ginsbourger (2013) leiten basierend auf der Tallis Formel eine exakte Formel für das q-EI ab, welche allerdings nur für $q < 10$ in realistischer Zeit berechenbar ist. Bischl et al. (2014) schlagen eine Lösung vor, die auf der multikriteriellen evolutionären Suche basiert. Dabei wird neben den gewünschten Zielkriterien wie EI

[10] Da keine echten Zielfunktionsauswertungen benötigt werden, kann und soll $N_{FS.points}$ groß gewählt werden.

oder Modellunsicherheit auch die Distanz zwischen den vorgeschlagenen Punkten berücksichtigt (diese soll maximiert werden). In jeder Iteration wird somit eine Population von q Punkten mit Hilfe eines evolutionären Algorithmus generiert, die die Pareto-Front der ausgewählten Kriterien möglichst gut approximiert. Diese Strategie konnte zum Teil signifikant bessere Ergebnisse als q-EI[11] und andere einfachere Verfahren wie z.B. q-LCB (Hutter et al., 2012) liefern.

Weitere aktuelle Forschungsziele in diesem Bereich sind beispielsweise das Behandeln von sogenannten asymmetrischen Funktionsauswertungen oder bedingten Parameterräumen. Bei asymmetrischen Problemen hängt die Dauer der Funktionsauswertung sehr stark von dem vorgeschlagenen Punkt ab und kann zwischen wenigen Minuten und mehreren Tagen variieren. Werden q Punkte gleichzeitig vorgeschlagen und parallel ausgewertet, so kann die Optimierung nicht vorwärtskommen bis alle Punkte der aktuellen Iteration ausgewertet sind. Es müssen also Strategien entwickelt werden, neue Punkte aufgrund von nicht vollständiger Information aus der aktuellen Iteration vorzuschlagen und das Modell nach Eingang neuer Information adaptiv anzupassen. Über bedingte Parameterräume spricht man wenn der zulässige Bereich bzw. das Vorhandensein eines Parameters von der Einstellung eines anderen Parameters abhängt.

Zusammenfassend lässt sich sagen, dass die Optimierung multikriterieller, verrauschter und sehr zeitintensiver Zielfunktionen immer mehr in den Fokus der Anwendung gelangt und angesichts der modernen Computerressourcen die parallele Auswertung mehrerer Punkte pro Iteration unverzichtbar wird.

6.3 Instanzgebundene modellbasierte Optimierung

Bisher wurde nicht thematisiert, was unter einer Funktionsauswertung verstanden wird. Hier sind zwei Fälle zu unterscheiden: einfache und zusammengesetzte Auswertungen. Bei einer einfachen Auswertung wird nur ein einziges Objekt (wie z.B. eine Simulation) untersucht. Bei einer zusammengesetzten Auswertung, dagegen, sollen mehrere Objekte bzw. Instanzen ausgewertet werden, wobei die Zielfunktion z.B. die mittlere Güte über alle Instanzen ist. Der klare Nachteil der klassischen MBO ist ein fehlendes Konzept für die Behandlung der einzelnen Instanzen.

Das instanzgebundene SMAC-Verfahren von Hutter et al. (2011) unterscheidet sich in mehrerer Hinsicht von dem in Algorithmus 6.1 dargestellten MBO-Schema. Probleminstanzen sind dabei verschiedene Datensätze und optimiert werden die Parameter eines Lernverfahrens (Algorithmuskonfiguration). Es sollen also diejenigen Parameter gefunden werden, für die das Lernverfahren die besten Ergebnisse über alle Datensätze erzielt. Je nach Lernverfahren (z.B. Zufallswald) kann es sich um eine verrauschte Optimierung handeln.

[11] Der Vergleich mit q-EI von Chevalier und Ginsbourger (2013) wurde erst nach der Veröffentlichung von Bischl et al. (2014) durchgeführt. An der Veröffentlichung dieses Ergebnisses wird derzeit gearbeitet.

Es wird vorausgesetzt, dass jede Probleminstanz $I \in \mathscr{I}$ einen Vektor von Merkmalen besitzt, wobei diese Merkmale zusammen mit den zu optimierenden Parametereinstellungen als Einflussgrößen in das Surrogatmodell eingehen (die Autoren nennen es *joint model*). Das Surrogatmodell trägt also die Information, welche Punkte an welchen Instanzen bereits evaluiert wurden. Der Initialisierungsschritt in SMAC besteht im Gegensatz zu MBO in der Evaluierung der Zielfunktion an nur wenigen zufällig ausgewählten Instanzen. Außerdem werden statt nur einem p Punkte (Kandidaten) als Ergebnis der EI-Maximierung vorgeschlagen (s. Abschnitt 6.2.4), und es wird ein zusätzlicher Intensivierungsschritt (engl. *intensify step*) durchgeführt. Bei diesem Schritt wird zuerst der aktuell beste Punkt $\boldsymbol{x}_{best}$ an einer neuen zufällig ausgewählten Instanz ausgewertet und anschließend wird jeder Kandidat an einer Untermenge von Instanzen $I_{iter.nr} \subseteq \mathscr{I}$ iterativ evaluiert bis nachweisbar ist, dass er schlechter als $\boldsymbol{x}_{best}$ ist, oder – wenn das nicht der Fall ist – dieser Kandidat als der neue beste Punkt gesetzt wird.

In dieser Arbeit liegt auch ein instanzgebundenes Optimierungsproblem vor, wobei hier die Musikstücke als Instanzen gelten. Angesicht der Tatsache, dass bei SMAC die Instanz-Merkmale für die Surrogatmodellierung berechnet werden sollen, erweist sich dieser Algorithmus für die angestrebte Anwendung als nicht gut geeignet. Denn zum einen ist die Wahl der geeigneten Musikstück-Merkmale nicht trivial und zum anderen ist unklar, ob bei der Echtzeitanwendung diese Merkmale auch mitbestimmt werden sollten. Wegen der hohen Komplexität des SMAC-Algorithmus und seiner fraglichen Eignung für die Einsatzzeiterkennung wurde in Bauer et al. (2015) ein einfacheres und prinzipiell anderes Vorgehen vorgeschlagen, welches im Folgenden detailliert beschrieben wird. Dieses Vorgehen wurde FMBO genannt (von engl. *Fast Model Based Optimization*) und wird auch hier so abgekürzt. Zwar wird hier die Idee von FMBO für den Fall der Einsatzzeiterkennung motiviert, das Verfahren ist jedoch auch für andere instanzgebundene Optimierungsprobleme geeignet.

Algorithmus 6.2 fasst das Schema von FMBO zusammen, wobei mit M_{sur} das in Abschnitt 6.2.2 behandelte MBO-Surrogatmodell bezeichnet wird. Auch hier wird von einem Minimierungsproblem ausgegangen. Da der F-Wert zu maximieren ist, wird in dieser Arbeit der negierte F-Wert minimiert. An einigen Stellen von FMBO sowie grundsätzlich bei der klassischen MBO wird unter $y = f(\boldsymbol{x})$ der mittlere F-Wert über alle Instanzen an dem Punkt $\boldsymbol{x}$ verstanden.

FMBO basiert auf der Beobachtung, dass sich die „schlechten“ Parametereinstellungen des EZE-Algorithmus bei den meisten Musikstücken in eher kleineren F-Werten widerspiegeln. Somit können solche „schlechten“ Einstellungen bereits nach der Auswertung nur weniger Instanzen erkannt werden. In diesen Fällen soll anstatt der zeitintensiven Auswertung des gesamten Datensatzes lediglich eine Schätzung des mittleren F-Wertes über alle Stücke in die MBO-Hauptschleife zurückgegeben werden.

Sei k_{all} die Gesamtzahl der Probleminstanzen und k_{sel} die Anzahl der Instanzen, die für die Auswahl (engl. *selection*) der „guten“ und der „schlechten“ Punkten berücksichtigt werden. Es können verschiedene Mechanismen verwendet werden, um die k_{sel} Instanzen auszuwählen. In

Algorithmus 6.2: Schnelle instanzgebundene modellbasierte Optimierung (FMBO).

1 Generiere ein Startdesign $\mathscr{D} \subset \mathscr{X}$;
2 Evaluiere die Punkte des Startdesigns auf allen Instanzen: $\boldsymbol{y} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^T$;
3 Clustere k_{all} Instanzen anhand ihrer F- und D-Werte in k_{sel} Cluster (mittels *k-means*);
4 Wähle zufällig eine Instanz aus jedem Cluster aus und bilde aus ihnen die Selektionsdatenbank;
5 Wende Vorwärts-Variablenselektion auf das lineare Modell M_{sel}: '$y \sim$ individuelle F-Werte von selektierten Instanzen' bezüglich R^2_{adj} an. Wähle dabei $k'_{sel} \leq k_{sel}$ Instanzen aus, so dass das Selektionsmodell M'_{sel} mind. $R^2_{adj} = 0.98$ hat;
6 **while** *Optimierungsbudget ist nicht überschritten* **do**
7 Passe das Surrogatmodell M_{sur} auf $\mathscr{D}$ und $\boldsymbol{y}$ an;
8 Bestimme den nächsten Punkt $\boldsymbol{x}^*$ mittels Optimierung des Zielkriteriums;
9 Evaluiere $\boldsymbol{x}^*$ an den Instanzen der Selektionsdatenbank mit k'_{sel} Instanzen;
10 Sage den mittleren F-Wert über alle Instanzen für $\boldsymbol{x}^*$ vorher: $\hat{y}^* = M'_{sel}(\boldsymbol{x}^*)$;
11 Berechne das 99% Konfidenzintervall für die Vorhersage: $[\hat{y}^*_u, \hat{y}^*_o]$;
12 **if** $\hat{y}^*_u \leq y_{min}$ **then**
13 $\boldsymbol{x}^*$ wird als „guter" Punkt klassifiziert;
14 Evaluiere $\boldsymbol{x}^*$ an den restlichen $k_{all} - k'_{sel}$ Instanzen: $y^* = f(\boldsymbol{x}^*)$;
15 Aktualisiere Design und Zielgröße: $\mathscr{D} \leftarrow (\mathscr{D}, \boldsymbol{x}^*)^T$ und $\boldsymbol{y} \leftarrow (\boldsymbol{y}, y^*)^T$;
16 Aktualisiere das Selektionsmodell M'_{sel} durch die neue Beobachtung;
17 Sage die in $\boldsymbol{y}$ enthaltenen Schätzwerte für die „schlechten" Punkte durch das aktualisierte M'_{sel} vorher;
18 **else**
19 $\boldsymbol{x}^*$ wird als „schlechter" Punkt klassifiziert;
20 Aktualisiere Design und Zielgröße: $\mathscr{D} \leftarrow (\mathscr{D}, \boldsymbol{x}^*)^T$ and $\boldsymbol{y} \leftarrow (\boldsymbol{y}, \hat{y}^*)$;
21 **end**
22 **end**
23 **return** $y_{min} = \min(\boldsymbol{y})$ und der zugehörige $\boldsymbol{x}_{min}$.

Bauer et al. (2015) wurde ein clusterbasiertes Verfahren angewendet, das hier etwas erweitert wird. Dabei wird bei der (kompletten) Auswertung des Startdesigns $\mathscr{D}$ für jedes Musikstück nicht nur sein individueller F-Wert sondern auch der individuelle D-Wert gespeichert (D-Wert Erweiterung). Anschließend werden die k_{all} Instanzen aufgrund ihrer F- und D-Werte in k_{sel} Cluster mittels des *k-means* Algorithmus aufgeteilt (Hartigan und Wong, 1979). Die zufällig ausgewählten Repräsentanten aus jedem Cluster (einer pro Cluster) bilden dann die sogenannte Selektionsdatenbank (Zeilen 2 bis 4 von Alg. 6.2). Auf diese Weise wird angestrebt, möglichst große Diversität der Musikstücke in dieser Datenbank zu erreichen.

Aus zahlreichen Vor-Experimenten mit 5% der Daten für k_{sel} hat sich allerdings ergeben, dass das lineare Modell M_{sel} mit individuellen F-Werten von k_{sel} Instanzen als Einflussgrößen und dem mittleren F-Wert über k_{all} Instanzen als Zielgröße sehr häufig ein adjustiertes Bestimmtheitsmaß R^2_{adj} von mehr als 0.99 aufweist. Das bedeutet, dass die Anzahl der selektierten Instanzen reduziert

werden kann, ohne große Verluste in der Modellanpassungsgüte hinnehmen zu müssen. Dadurch wird ein zusätzlicher Schritt als Erweiterung von FMBO motiviert – Vorwärts-Variablenselekiton der ausgewählten k_{sel} Instanzen. Dabei werden so viele Instanzen k'_{sel} in das Modell aufgenommen, bis $R^2_{adj} \geq 0.98$ erreicht wird (Zeile 5). In dem Fall, dass so ein großes R^2_{adj} auch mit allen k_{sel} nicht erreicht wird, gilt: $k'_{sel} = k_{sel}$. Das auf den k'_{sel} Instanzen angepasste Modell (d.h. nach der Variablenselektion) wird mit M'_{sel} bezeichnet.

In den sequentiellen Schritten werden die vorgeschlagenen Punkte zuerst an den Instanzen der Selektionsdatenbank ausgewertet (Zeile 9) und laut dem Selektionsmodell als „gute“ bzw. „schlechte“ Punkte klassifiziert. Auch hier sind verschiedene Ansätze denkbar. Eine einfache Variante ist die konfidenzintervallbasierte Entscheidung: für den Punkt $\boldsymbol{x}^*$ werden die Vorhersage $\hat{y}^*$ und das 99% Konfidenzintervall[12] $[\hat{y}^*_u, \hat{y}^*_o]$ laut dem M'_{sel} berechnet (Zeilen 10 bis 11). Falls die untere Schranke des Konfidenzintervalls kleiner als der bisher erreichte minimale F-Wert ist, dann wird $\boldsymbol{x}^*$ als ein „guter“ Punkt klassifiziert, sonst als ein „schlechter“.

Die „guten“ Punkte werden an den restlichen $k_{all} - k'_{sel}$ Musikstücken des kompletten Datensatzes ausgewertet (Zeile 14), $\mathscr{D}$ und $\boldsymbol{y}$ aktualisiert (Zeile 15) und das Selektionsmodell neu angepasst, um von der neuen Information (zusätzliche Beobachtung) zu profitieren (Zeile 16). Liegt ein „schlechter“ Punkte vor, so wird $y^* = f(\boldsymbol{x}^*)$ lediglich durch $\hat{y}^*$ geschätzt und in die MBO-Schleife zurückgegeben (Zeile 20). Wichtig zu bemerken ist, dass diese Schätzungen nach jeder Neuanpassung des Selektionsmodells aktualisiert werden (Zeile 17).

In dieser Arbeit werden 5% der Musikstücke (der Trainingdatenbank) als k_{sel} Instanzen betrachtet. Die Anzahl der tatsächlich ausgewählten Selektionsinstanzen k'_{sel} (nach der Vorwärts-Variablenselektion) liegt allerdings oft weit unter der genannten Prozentzahl und wird in dem Auswertungsteil dieser Arbeit weiter diskutiert (s. Abschnitt 8.2).

Zu bemerken ist, dass sowohl für den Clusteralgorithmus als auch für das Selektionsmodell auch andere sinnvolle Alternativen denkbar sind. Es würde sich beispielsweise die Entwicklung eines Kaskaden ähnlichen Selektionsmodells anbieten, bei dem zuerst sehr wenige Musikstücke ausgewertet werden und dann je nach deren mittlerem F-Wert und der Höhe des Varianzschätzers für alle Instanzen entweder noch mehr Stücke berücksichtigt werden sollten (bei hohen F-Werten und großer Varianz) oder die Auswertung zu stoppen ist. Auch die Anwendung von Klassifikationsverfahren (wegen Selektion nach „gut“ und „schlecht“) wäre für diese Aufgabe sehr intuitiv.

6.4 Validierung

In den praktischen Anwendungen werden häufig für ein aufgestelltes Problem mehrere geeignete Modelle entwickelt und anschließend die Frage gestellt, welches der angepassten Modelle das „wahre“ Modell am besten approximiert. Methoden der Modellvalidierung ermöglichen es,

[12] In Bauer et al. (2015) wurden 99% und 95% Konfidenzintervalle verglichen. Der 95% Intervall führt dazu, dass mehr Punkte als „schlecht“ gewertet werden. Da der 99% Intervall bessere Ergebnisse lieferte, wird es für diese Arbeit übernommen.

verschiedene Modelle anhand unterschiedlicher interessierender Kriterien (Gütemaße) zu vergleichen. Die Wahl der Gütemaße ist anwendungsbedingt und kann z.B. durch Anpassungsgüte, Prognosefehler oder Komplexität der Modelle ausgedrückt werden.

Nicht nur Modelle, sondern auch komplexe (konkurrierende) Verfahren können validiert werden. Es wird dann nicht mehr von Modell- sondern von Verfahrensparametern gesprochen. Hier wären z.B. das verwendete Surrogatmodell (naives Kriging vs. dummy Kriging) oder das Optimierungsschema (MBO vs. FMBO) solche Parameter. Da in der klassischen Literatur von der Modellvalidierung ausgegangen wird und man ein komplexes Verfahren auch als ein Modell bezeichnen kann, wird in dem allgemein einführenden Teil dieses Abschnittes die Terminologie der Modellvalidierung verwendet. Auf die hier relevante Verfahrensvalidierung wird in dem anschließenden Teil eingegangen.

6.4.1 Grundlagen der Modellvalidierung

Ein oft verwendetes Gütemaß bei der Modellvalidierung ist der Prognosefehler der Modelle (Modellfehler), der widerspiegeln soll, wie gut die Zielfunktionswerte für neue Beobachtungen geschätzt werden. Bei Klassifikationsproblemen werden zur Beurteilung der Modellregeln meistens die so genannten Fehlklassifikationsraten verwendet und bei Regressionen die Fehlerquadratsummen. Bei Hastie et al. (2001) wird (speziell für Fehlerraten) gezeigt, dass falls die Zielvariable auf denselben Daten vorhergesagt wird, auf denen die Modellanpassung erfolgte, der wahre Modellfehler unterschätzt wird. Dies geschieht, weil in diesem Fall das Modell „überspezialisiert“ wird, d.h. die Modellparameter werden so geschätzt, dass sie auf den vorliegenden Daten gut funktionieren.

Methoden. Für die Schätzung des wahren Prognosefehlers kommen oft Trainieren und Testen (engl. *train-and-test*) Methoden zum Einsatz. Dabei wird die vorliegende Stichprobe in einen Trainingsdatensatz, der zum Lernen der Modellregeln dient, und einen Testdatensatz, auf dem das gelernte Modell angewendet wird, nach einem bestimmten Vorgehen aufgeteilt. Von Interesse ist dann der Testdaten-Prognosefehler.

Trainieren und Testen Methoden unterscheiden sich in dem Vorgehen, wie die vorliegende Stichprobe in die Test- bzw. Trainingsdatensätze aufgeteilt wird. Das einfachste Vorgehen ist die Auslassen-Methode (engl. *holdout*), bei der ein gewisser Anteil der Daten (üblicherweise 2/3) als Trainingsdaten und die restlichen Daten als Testdaten verwendet werden. Sehr verbreitet ist allerdings die *k-fache Kreuzvalidierung*, bei der alle Beobachtungen zufällig in k ungefähr gleich große, disjunkte Teildatensätze aufgeteilt werden. Jeder Teildatensatz wird einmal als Testdatensatz gewählt, wobei die restlichen Teildatensätze zum Trainingsdatensatz vereinigt werden. Der „kreuzvalidierte“ Prognosefehler des Modells lässt sich als Mittelwert der k Testdaten-Prognosefehler berechnen.

Ein Spezialfall der k-fachen Kreuzvalidierung ist die *leave-one-out*-Methode, bei der in jedem Durchlauf eine Beobachtung aus dem Datensatz entfernt wird und somit nahezu der gesamte Datensatz für die Modellbildung zur Verfügung steht. Zwar ist diese Methode etwas zeitaufwendig und eignet sich deswegen nur für kleine Datensätze, allerdings liefert sie einen beinahe unverzehrten Schätzer für den wahren Modellfehler (Weiss und Kulikowski, 1991). Es existieren viele Erweiterungen der klassischen k-fachen Kreuzvalidierung wie z.B. r-fach wiederholte k-fache Kreuzvalidierung (Kohavi, 1995), Monte Carlo Kreuzvalidierung (Molinaro et al., 2005), Leave-two-out (Krzanowski und Hand, 1997) oder Delete-d-Jackknife Sampling (Weihs, 1993).

Eine weitere populäre Trainieren und Testen Methode ist die *Bootstrap-Methode*, die in vielen Variationen zu finden ist (Efron und Tibshirani, 1997). Besteht der Datensatz aus n Beobachtungen, so setzt sich beispielsweise bei der *e0-Bootstrap-Methode* der Trainingsdatensatz aus n-mal zufällig mit Zurücklegen ausgewählten Beobachtungen zusammen. Diejenigen Beobachtungen, die nicht im Trainingsdatensatz vorkommen, bilden den Testdatensatz (Weiss, Kulikowski, 1990). Außerdem existieren viele Kombinationen von Kreuzvalidierung und Bootstrap-Methoden (Fu et al., 2005).

Auswertung. Ein allgemeines Problem bei der Kreuzvalidierung ist die Tatsache, dass die Fehlerraten, die an den jeweiligen Testdatensätzen ermittelt werden, miteinander korreliert sein können, da die Modelle an sehr ähnlichen Trainingsdaten gelernt werden (Bengio und Grandvalet, 2004; Bischl et al., 2012). Dies stellt ein Problem dar, wenn man die Validierungsergebnisse mehrerer Modelle oder Verfahren miteinander mittels statistischer Test vergleichen möchte, da die Unabhängigkeit der Beobachtungen meist eine wichtige Voraussetzung ist. Als ein Ausweg bietet sich die mehrfache Wiederholung der Kreuzvalidierung an, die allerdings – je nach Anwendungsproblem – sehr zeitintensiv sein kann. In Bengio und Grandvalet (2004), Demšar (2006) und Dietterich (1998) werden verschiedene parametrische und nicht-parametrische Tests für dieses Problem diskutiert und in Demšar (2006) wird auf das Einhalten des multiplen Testniveaus bei multiplen Vergleichen eingegangen.

6.4.2 Validierung der Optimierungsverfahren

In diesem Abschnitt wird die konkrete Umsetzung der Validierung von Optimierungsverfahren beschrieben, die auf die interessierende Anwendung zugeschnitten ist. Die allgemeinen Ideen sind allerdings auf beliebige andere Validierungsprobleme übertragbar.

Um das oben genannte Problem der abhängigen Stichproben zu vermeiden, wird hier die Auslassen-Methode (im Weiteren als Holdout-Methode bezeichnet) r mal wiederholt. Dabei werden die Daten r mal zufällig in Training- und Testdaten aufgeteilt (im 2/3 zu 1/3 Verhältnis). Auf den Trainingsdaten werden die gewünschten zu vergleichenden Optimierungsverfahren ausgeführt und die jeweiligen besten Parametereinstellungen des EZE-Algorithmus mittels eines Kriteriums (hier: der F-Wert) identifiziert.

Die identifizierte optimale Parametereinstellung wird dann auf den Testdaten validiert, wobei sich daraus das endgültige Gütemaß des Verfahrens in der r-ten Wiederholung ergibt. Hier ist insbesondere der F-Wert als Gütemaß von Interesse, der D-Wert wird jedoch bei der Auswertung zusätzlich analysiert und ist dann für die Empfehlung der besten EZE-Einstellung über alle Wiederholungen von Interesse. Um die Ergebnisse noch besser vergleichbar zu machen, wird in jeder Holdout-Wiederholung jeweils das gleiche Startdesign für alle Optimierungsstrategien verwendet.

Somit ergibt sich für jede Optimierungsstrategie ein r-dimensionaler Stichprobenvektor mit unabhängigen Beobachtungen: Gütemaß der Einsatzzeiterkennung auf den jeweiligen Testdaten. Da hier keine Normalverteilungsannahme gemacht werden kann, werden Stichprobenpaare mit Hilfe des Wilcoxon-Rangsummentests (Siegel und Castellan, 1988, S. 128 ff.) miteinander verglichen. Dieser Test setzt unabhängige Stichprobenvariablen voraus, die die gleiche Verteilungsfunktion besitzen, sich allerdings um einen Verschiebungsparameter unterscheiden. Bei einem zweiseitigem Test wird getestet, ob dieser Verschiebungsparameter gleich Null ist. Die beiden Stichprobenausprägungen werden zu einer gemeinsamen Stichprobe kombiniert und geordnet. Die Teststatistik ist dann die Summe der Ränge der ersten Stichprobe.

Hinsichtlich der Stichprobengröße zeigen Siegel und Castellan (1988) (S. 132), dass bereits bei $r > 10$ bzw. $r > 20$ eine ausreichende asymptotische Annäherung der Teststatistik an die Standardnormalverteilung gewährleistet wird, was für die Berechnung der kritischen Werte von Bedeutung ist. Aus diesem Grund wird in dieser Arbeit $r = 30$ gewählt, um einerseits eine ausreichend große Stichprobe zu erreichen, andererseits aber die Rechenzeit nicht zu stark zu erhöhen.

Es ist eine Besonderheit bei der Validierung der multivariaten Einsatzzeiterkennung zu berücksichtigen: Die Evaluierung der Zielfunktion auf dem Testdatensatz unterscheidet sich leicht von dem in Kapitel 4 erläuterten Vorgehen. Und zwar wird hier das Klassifikationsmodell nicht auf einem Teil der Teststücke neu gelernt, sondern das zu dieser Parametereinstellung im Optimierungsschritt gelernte Modell auf alle Musikstücke der Testdatenbank angewendet. Das Klassifikationsmodell ist nämlich – wie auch die Parametereinstellungen – ein Ergebnis der Optimierung und soll in der Testphase nicht mehr verändert werden, da sonst die Echtzeitanwendung-Fähigkeit des Verfahrens nicht gewährleistet wäre.

6.4.3 Realisierung der Experimente

Sowohl die Anwendungsalgorithmen (EZE) als auch die Optimierungsverfahren sind in der Programmiersprache R (R Core Team, 2014) implementiert. Für einzelne Aufgaben wurden außerdem Matlab (Matlab, 2013) und Python[13] Programme verwendet. Die Experimente wurden auf dem Linux-HPC Cluster System der TU Dortmund[14] durchgeführt. Parallele Auswertungen der Experimente wurden mit Hilfe des R-Pakets **BatchJobs** (Bischl et al., 2015b) realisiert.

[13] Version 2.7.0, `https://www.python.org`, Stand: 01.05.2015.

[14] `http://lidong.itmc.tu-dortmund.de/ldw/index.php?title=Systemoverview&oldid=259`.

Kapitel 7

Problemstellung

7.1 Forschungsfragen und Planung der Experimente

In den vorherigen Kapiteln dieser Arbeit wurden die Optimierungsverfahren (MBO und FMBO, s. Kapitel 6) sowie das Anwendungsproblem (univariate und multivariate EZE, s. Kapitel 3, 4) ausführlich vorgestellt. Es werden nun folgende aufeinander aufbauende Forschungsfragen aufgestellt, welche sich in zwei Klassen unterteilen lassen: Optimierungs- und Anwendungsfragen.

- **Optimierungsfrage 1**: Ist das naive Kriging Surrogatmodell genau so gut wie das statistisch korrektere aber Zeit aufwändigere dummy Kriging Surrogatmodell für MBO?
- **Optimierungsfrage 2**: Ist FMBO trotz weniger Funktionsauswertungen genau so gut wie MBO? Wie sieht das Zeitersparnis-Güteverlust-Verhältnis aus?
- **Anwendungsfrage 1**: Liefern die durch Optimierung gefundenen Parametereinstellungen des EZE-Algorithmus bessere F-Werte als die „State of the Art“ Empfehlungen?
- **Anwendungsfrage 2**: Gibt es einen relevanten Unterschied zwischen der erreichten Güte bei der *online*, *pseudo-online* bzw. *offline* EZE?
- **Anwendungsfrage 3**: Spiegelt sich die Erzeugungsart der Musikaufnahmen (echt vs. synthetisch) in den F-Werten bzw. den optimalen EZE-Algorithmus Einstellungen wieder?
- **Anwendungsfrage 4**: Erzielt die multivariate EZE bessere Ergebnisse als die univariate EZE?

Für die Planung der Experimente ist zuerst ein kurzer Überblick über die zu vergleichenden Optimierungsstrategien sowie EZE-Methoden notwendig:

- 2 Optimierungsstrategien: MBO und FMBO (Abschnitte 6.2 und 6.3),
- 2 Surrogatmodelle: naives Kriging und dummy Kriging (Abschnitt 6.2.2),
- 4 EZE-Konzepte: univariat (Kapitel 3) und multivariat mit drei Klassifikatoren (*logReg*, *randForest* und *svm*, Kapitel 4),
- 3 EZE-Arten: *online*, *pseudo-online* und *offline* (Abschnitt 3.1),
- 2 Datenbanken: WAV- und MIDI-Datenbank (Abschnitt 5.2).

Insgesamt ergeben sich 96 Verfahren-Problem-Kombinationen. Für jede Kombination sollte eine 30-fach wiederholte Holdout-Validierung, wie in Abschnitt 6.4.2 beschrieben, angewendet werden. Dies würde 2 880 Optimierungsläufen entsprechen. Außerdem werden noch einige Läufe mit Referenzstrategien (z.B. Zufallssuche als Referenz zu MBO, realisiert durch ein entsprechend großes LHS-Design) durchgeführt. Je nach Verfahren-Problem-Kombination kann ein Optimierungslauf weniger als einen Tag (bei FMBO) aber auch mehrere Wochen (multivariate EZE mit *svm* Klassifikator) Rechenzeit in Anspruch nehmen.

Es erweist sich also praktisch als unmöglich, alle möglichen Kombinationen zu berücksichtigen. Um den Rechenaufwand zu reduzieren wird hier eine stufenweise Beantwortung der Forschungsfragen vorgeschlagen. Dabei werden basierend auf den Ergebnissen der bereits durchgeführten Experimente die nächsten Verfahren-Problem-Kombination geplant. Im Weiteren werden diese Stufen erläutert. Es werden hier jedoch nur die oben definierten Fragestellungen angesprochen. Die etwas breitere Auswertung der Ergebnisse in Kapitel 8 umfasst zusätzlich einige weitere Aspekte, welche an dieser Stelle nicht relevant sind.

Erste Stufe der Experimente. In der ersten Stufe werden 12 Verfahren-Problem-Kombinationen betrachtet: Die MBO Optimierungsstrategien mit naivem Kriging bzw. dummy Kriging als Surrogatmodelle werden für die drei Arten der univariate EZE (*online*, *pseudo-online* und *offline*) auf den beiden Datenbanken (WAV und MIDI) angewendet. Dies ergibt $12 \cdot 30 = 360$ Optimierungsexperimente und $6 \cdot 30 = 180$ Referenzexperimente für MBO (Zufallssuche). Die Ergebnisse der Experimente sollen zu der Beantwortung der ersten Optimierungsfrage sowie der ersten drei Anwendungsfragen beitragen.

Zweite Stufe der Experimente. In der zweiten Stufe geht es hauptsächlich um den Vergleich von MBO mit FMBO. Dafür werden 6 zusätzliche Verfahren-Problem-Kombinationen betrachtet: FMBO (mit naivem Kriging oder dummy Kriging Surrogatmodell, je nach den Ergebnissen der ersten Stufe), angewendet auf die drei Arten der univariaten EZE und die beiden Datenbanken. Somit ergeben sich $6 \cdot 30 = 180$ Optimierungsläufe. Außerdem wird ein Referenzverfahren für FMBO mit 30 Läufen implementiert (für *online* EZE auf der WAV-Datenbank). Diese Stufe dient der Beantwortung der zweiten Optimierungsfrage. Weiterhin sollte das Zeitersparnis-Güteverlust-Verhältnis der FMBO Strategie analysiert und eine Entscheidung getroffen werden, mit welcher Optimierungsstrategie die nächste Stufe der Experimente durchgeführt werden soll.

Dritte Stufe der Experimente. Anschließend soll die letzte Anwendungsfrage beantwortet werden. Da die Optimierung der multivariaten EZE besonders zeitaufwändig ist, wird sie mit Hilfe nur einer Optimierungsstrategie (ermittelt in Stufe zwei) und nur auf der WAV-Datenbank (wegen Vergleichbarkeit zu „State of the Art“) durchgeführt. Im Fall der *online* EZE kommen alle drei Klassifikatoren zum Einsatz, was insgesamt $3 \cdot 30 = 90$ neue Experimente ergibt. Die Optimierung der *offline* EZE (weitere 30 Optimierungsläufe) wird lediglich mit dem besten Klassifikator laut der Ergebnisse der *online* EZE Optimierung realisiert.

7.2 Behandlung des multiplen Testproblems

In Abschnitt 6.4.2 wurde bereits die Wahl des Wilcoxon-Rangsummentests für Stichprobenvergleiche motiviert. Eine Stichprobe besteht dabei aus 30 F-Werten des entsprechenden Verfahrens (ein Wert pro Holdout-Wiederholung). Offensichtlich werden einige Stichproben mehrmals für das Testen verwendet, da sie für mehrere Fragestellungen relevant sind. Es entsteht somit die Problematik des multiplen Testens, so dass eine angemessene Adjustierung des Testniveaus notwendig ist. Dies ist insbesondere dann wichtig, wenn man die Wahrscheinlichkeit mindestens eine Nullhypothese fälschlicherweise abzulehnen durch ein festgelegtes globales Signifikanzniveau α begrenzen möchte. In dieser Arbeit wird $\alpha = 5\%$ verwendet. Dabei soll zuerst eine Gruppe von k Nullhypothesen ($k \leq$ Gesamtzahl der Nullhypothesen) Nullhypothesen festgelegt werden, für die diese Begrenzung gelten soll.

Eine sehr verbreitete Methode für eine solche Adjustierung ist das Bonferroni-Holm-Verfahren (Holm, 1979). Für jede einzelne Hypothese wird zuerst der p-Wert ermittelt. Anschließend werden die p-Werte in aufsteigender Reihenfolge sortiert und mit lokalen Signifikanzniveaus verglichen. Für den kleinsten p-Wert ist das lokale Signifikanzniveaus mit α/k gegeben, für den zweitkleinsten mit $\alpha/(k-1)$ usw. Der größte p-Wert wird mit α verglichen. Abgelehnt werden alle Nullhypothesen, deren p-Werte kleiner als die jeweiligen lokalen Signifikanzniveaus sind unter der Bedingung, dass alle Nullhypothesen mit noch kleineren p-Werten auch abgelehnt wurden.

Die Problematik in dieser Arbeit besteht darin, die Gruppe der interessierenden Hypothesen, für die das Niveau adjustiert werden sollte, zu definieren. Die Gesamtzahl der durchzuführenden Tests ist viel höher als die maximale Anzahl des Heranziehens einer konkreten Stichprobe für das Testen. Dies erklärt sich damit, dass man nicht alle paarweisen Vergleiche durchführt. Es ist somit nicht günstig, k als die Gesamtzahl der Tests zu definieren. Hier wird also ein Kompromiss vorgeschlagen, der auf der Adjustierung des Niveaus innerhalb der Gruppenvergleiche basiert.

Aus dem obigen Abschnitt geht hervor, dass z.B. die Ergebnisse der besten MBO-Strategie für univariate *online* EZE auf der WAV-Datenbank sechs mal zum Testen verwendet werden und zwar bei allen Anwendungs- und Optimierungsfragen. Es entsteht somit eine Gruppe von sechs Nullhypothesen, für die das Niveau adjustiert werden sollte. Jede einzelne Stichprobe bildet eine eigene Gruppe, kommt aber auch in mehreren anderen Gruppen vor. Das lokale Niveau für die Tests, die diese Stichprobe berücksichtigen, wird allerdings nach derjenigen Gruppe bestimmt, die die meisten Testprobleme behandeln.

Die Hypothesen werden im Laufe der Analyse in Kapitel 8 sequentiell vorgestellt. Da zum Zeitpunkt des Testens nicht klar ist, wie klein der ermittelte p-Wert im Vergleich zu den anderen p-Werten der jeweiligen Gruppe ist, werden die p-Werte aller Tests in dieser Arbeit in Tabelle B.1 im Anhang zusammengefasst. Für jede Hypothese wird also die zugehörige größte Gruppe ermittelt, die Anordnung des p-Wertes in dieser Gruppe vorgenommen und das entsprechende lokale Signifikanzniveau für den Vergleich bestimmt. Zwecks besserer Übersicht wird jede Stichprobe mit einem Hochindex versehen, der die Anzahl ihres bisherigen Verwendens für das Testen angibt.

Kapitel 8

Auswertung der Experimente

8.1 Erste Stufe der Experimente

In diesem Abschnitt wird die erste Stufe der Experimente (s. Abschnitt 7.1) behandelt. Unter anderen ist dabei interessant, ob die durch Modell basierte Optimierung gefundenen Parametereinstellungen mit den „State of the Art“ Ergebnissen konkurrieren können (Anwendungfrage 2 aus Abschnitt 7.1). Für diesen Zweck werden die Empfehlungen von Böck et al. (2012) auf denselben Teststücken in jeder Holdout-Wiederholung evaluiert wie auch bei der MBO-Validierung (s. Validierungsschema für Optimierungsstrategien in Abschnitt 6.4.2). Die Empfehlungen von Böck et al. (2012) sind durch folgende Parametereinstellungen gegeben:

- *online* EZE: $N = 2048$ Samples, $h = 441$ Samples, *wind.func* =„Hanning“, *spec.filter* =„Ja“, *spec.log* =„Ja“, $\ell = 1$, *odf* = *Spec.Flux*, $\alpha = 1$, *mov.fun* =„mean“, *mov.par* $= 0$, $\delta = 2.5$, $t(l_T) = 0.1$ s, $t(l_O) = 0.03$ s, $t(r_T) = 0$ s, $t(r_O) = 0$ s, $t(min.dist) = 0.03$ s, *onset.shift* $= 0.01$ s;
- *pseudo-online* EZE: wie *online*, die Amplitude des Signal wird aber vorher auf das Intervall $[-1,1]$ umskaliert;
- *offline* EZE: wie *pseudo-online*, außer $t(r_T) = 0.1$ s und $t(r_O) = 0.03$ s.

Außerdem wird die Zufallssuche als ein Referenzverfahren zu MBO implementiert: ein LHS-Design mit $25d$ Punkten (d.h. genau so viele Auswertungen wie bei MBO, d ist dabei die Anzahl der zu optimierenden Parameter). Die besten Parametereinstellungen der jeweiligen Suchläufe in 30 Wiederholungen werden unter gleichen Bedingungen wie bei MBO validiert.

8.1.1 Optimierungsfrage 1: naives Kriging vs. dummy Kriging

Die erste Optimierungsfrage (s. Abschnitt 7.1) wird auf beiden Datensätzen (WAV und MIDI) sowie für alle drei EZE-Arten (*online*, *pseudo-online* und *offline*) untersucht. Im Vordergrund steht hier die Frage, ob MBO mit naivem Kriging Surrogatmodell ähnlich gut funktioniert wie die statistisch korrektere Variante mit dummy Kriging Modell. Zuerst werden die Ergebnisse deskriptiv betrachtet: Abbildung 8.1 für die WAV-Datenbank und Abbildung 8.2 für die MIDI-Datenbank. Jeder Boxplot stellt die Verteilung der mittleren F-Werte in 30 Wiederholungen der Holdout-Methode dar. Die Farbe der Boxplots gibt die verwendete Strategie wieder und auf der x-Achse ist

notiert, um welche EZE-Art es sich handelt. Die Boxfarben in Abbildung 8.2 (MIDI-Datenbank) sind etwas dunkler als die in Abbildung 8.1 (WAV-Datenbank). Dies dient einer bessere Unterscheidung zwischen den beiden Datensätzen und erleichtert spätere grafische Vergleiche.

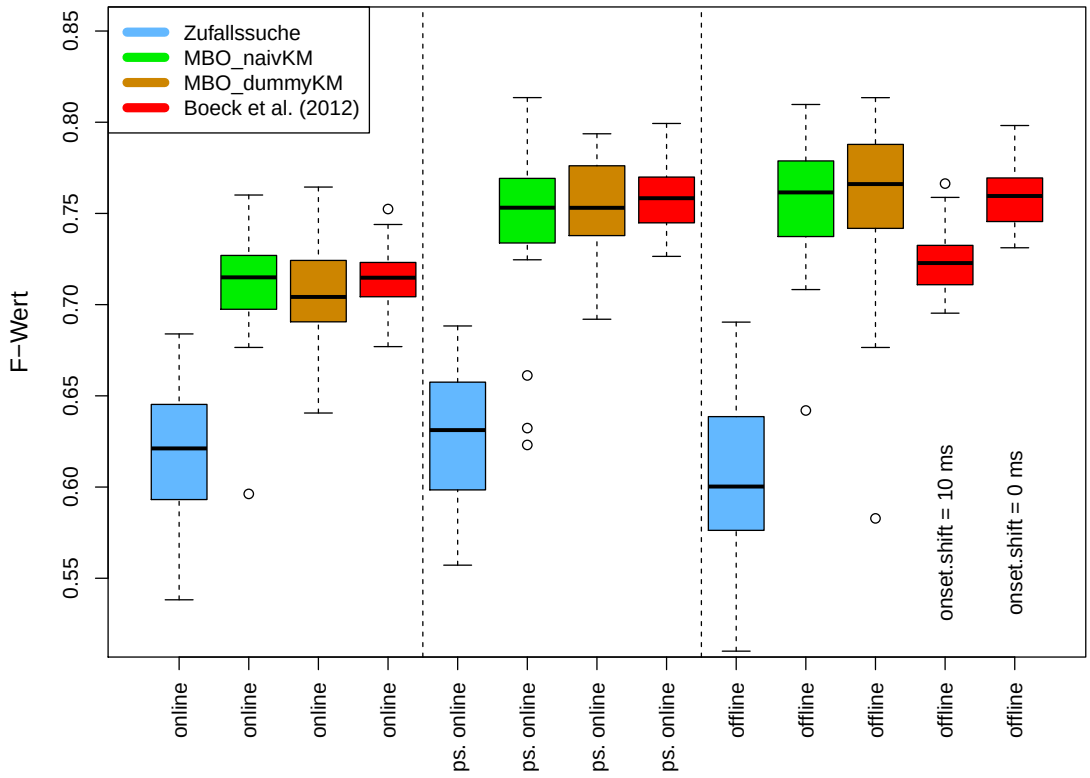

Abbildung 8.1: *Ergebisse der univariaten EZE auf der WAV-Datenbank.*

Da die Empfehlung von Böck et al. (2012) im *offline* Fall bei der WAV-Datenbank schlecht abgeschnitten hat (mit $onset.shift = 0.01$ s), wurde eine modifizierte Variante ohne Zeitverschiebung (d.h. mit $onset.shift = 0$ s) ausprobiert, welche offensichtlich bessere Ergebnisse aufweist. Auf die grauen Boxplots (Empfehlungen aus Tabelle 8.2) in Abbildung 8.2 wird in Abschnitt 8.1.6.2 ausführlich eingegangen.

Als erstes ist aus den beiden Abbildungen ersichtlich, dass die Optimierung sowie die „State of the Art“ Empfehlungen von Böck et al. (2012) bessere Parametereinstellungen liefern als die durch Zufallssuche gefundenen Parametersets. Rein deskriptiv scheint es keinen systematischen Unterschied zwischen MBO mit naivem Kriging und dummy Kriging Surrogatmodellen zu geben, abgesehen von der *pseudo-online* EZE auf der MIDI-Datenbank, wo das erste Surrogatmodell etwas besser abschneidet. Weiterhin fällt auf, dass die Verbesserung zwischen der *online* und *pseudo-online* EZE auf der MIDI-Datenbank nicht so stark ausgeprägt ist wie auf der WAV-Datenbank.

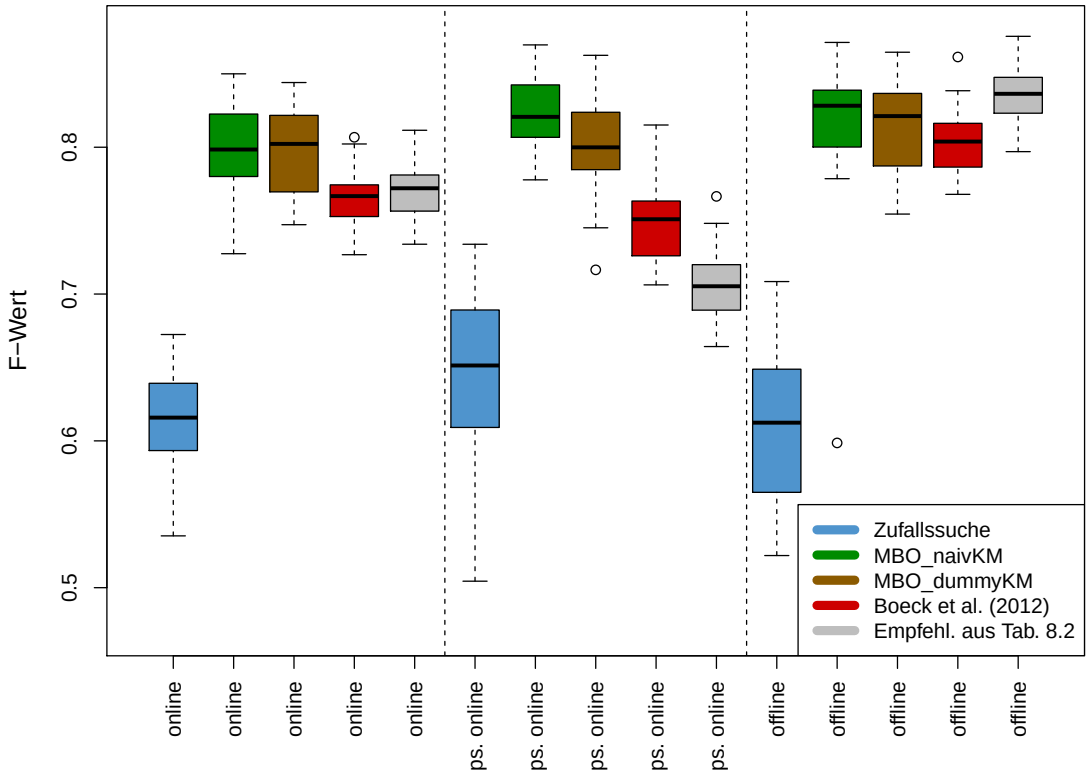

Abbildung 8.2: *Ergebnisse der univariaten EZE auf der MIDI-Datenbank.*

Die der ersten Anwendungsfrage entsprechenden Nullhypothesen werden unten aufgeführt. Außerdem werden direkt die entsprechenden p-Werte der Wilcoxon-Rangsummentests und Testentscheidungen notiert. Hier und im Weiteren steht der untere Index OF bei der Hypothesenbezeichnung für die Optimierungsfrage und AF für die Anwendungsfrage (jeweils versehen mit der entsprechenden Nummer). Der obere Index notiert die Datenbank bzw. die EZE-Art. Unter $MBO\_dummyKM\_wav\_online$ ist beispielsweise der Erwartungswert der entsprechenden Variable zu verstehen. Wie bereits in Abschnitt 7.2 erwähnt wurde, markiert der obere Index bei den Stichproben die Anzahl ihrer bisherigen Einsätze für das Testen.

- $H_{OF1}^{online.WAV}$: $MBO\_dummyKM\_wav\_online^{(1)} = MBO\_naivKM\_wav\_online^{(1)}$,
  - p-Wert = 0.4492 (nicht ablehnen);
- $H_{OF1}^{ps.online.WAV}$: $MBO\_dummyKM\_wav\_ps.online^{(1)} = MBO\_naivKM\_wav\_ps.online^{(1)}$,
  - p-Wert = 0.7082 (nicht ablehnen);
- $H_{OF1}^{offline.WAV}$: $MBO\_dummyKM\_wav\_offline^{(1)} = MBO\_naivKM\_wav\_offline^{(1)}$,
  - p-Wert = 0.5619 (nicht ablehnen);

- $H_{OF1}^{online.MIDI}$: $MBO\_dummyKM\_midi\_online^{(1)} = MBO\_naivKM\_midi\_online^{(1)}$,
  - p-Wert = 0.8315 (nicht ablehnen);
- $H_{OF1}^{ps.online.MIDI}$: $MBO\_dummyKM\_midi\_ps.online^{(1)} = MBO\_naivKM\_midi\_ps.online^{(1)}$,
  - p-Wert = 0.0041 (**ablehnen**);
- $H_{OF1}^{offline.MIDI}$: $MBO\_dummyKM\_midi\_offline^{(1)} = MBO\_naivKM\_midi\_offline^{(1)}$,
  - p-Wert = 0.2601 (nicht ablehnen).

Bis auf bei der Hypothese $H_{OF1}^{ps.online.MIDI}$ ergeben sich bei allen Nullhypothesen p-Werte größer als 0.05, d.h. es besteht dort kein signifikanter Unterschied in der MBO-Güte zwischen den beiden betrachteten Surrogatmodellen. Die Stichprobe *MBO_naivKM_midi_ps.online* wird insgesamt fünf mal in dieser Arbeit zum Testen herangezogen (s. Tabelle B.1), wobei der hier notierte p-Wert der drittkleinste ist. Nach dem Bonferroni-Holm-Verfahren (s. Abschnitt 7.2) ist also das lokale Signifikanzniveau für dieses Testproblem gleich $\alpha/3 = 0.0167$. Hier und auch bei allen nachfolgenden Tests ist die zweite Bedingung des Bonferroni-Holm-Verfahrens (alle Nullhypothesen mit noch kleineren p-Werten sollen abgelehnt werden) für das Ablehnen erfüllt. Somit kann die Nullhypothese $H_{OF1}^{ps.online.MIDI}$ abgelehnt werden. Die MBO mit naivem Kriging Surrogatmodell zeigt also in diesem Fall eine signifikant besser Güte als die mit dummy Kriging.

Was die Laufzeit der Surrogatmodell-Anpassung angeht, kann aus Tabelle 8.1 (Abschnitt 8.1.5) abgelesen werden, dass pro Optimierungslauf für eine dummy Kriging Modellanpassung und anschließende EI-Optimierung im Mittel 3.5 bzw. 4.2 Stunden mehr für *online* bzw. *offline* EZE benötigt werden als für eine naive Kriging Modellanpassung und anschließende EI-Optimierung. Da die EI-Optimierungszeit von der Anzahl der Variablen anhängt (welche bei dummy Kriging höher ist), wurde diese hier auch mitberücksichtigt. Aus den Ergebnissen der Signifikanztests und der Zeitanalyse geht hervor, dass das naive Kriging Surrogatmodell wegen seiner Einfachheit und Effizienz zu bevorzugen ist. Aus diesem Grund werden weitere Experimente (in der zweiten und dritten Experimentstufe, s. Abschnitte 8.2 und 8.3) ausschließlich mit naivem Kriging Surrogatmodell durchgeführt.

Eine Frage bleibt allerdings noch offen: Warum funktioniert naives Kriging trotz Verletzung der Normalverteilungsannahme so gut? Vor allem betrifft das die nominal skalierte Variable EZE-Merkmal mit 18 Ausprägungen. Dies könnte an der Größe des Startdesigns liegen: $5d$, was im *online* Fall 75 Punkten entspricht. Nach der Konstruktion des verwendeten LHS Designs kommt somit jedes EZE-Merkmal ca. 4 vor. Anscheinend reicht dies aus, um das Kriging Modell ausreichend gut anzupassen, denn bereits in den ersten sequentiellen MBO-Schritten wird in der Nähe des numerischen Wertes des besten Merkmals evaluiert. Interessant wäre die Erforschung der Güte von MBO mit naivem Kriging in Abhängigkeit von der Größe des Startdesigns (die von der Anzahl weiterer numerischer Parameter abhängt). Auf diese Thematik wird im Ausblick dieser Arbeit noch eingegangen.

8.1.2 Anwendungsfrage 1: optimierte Parameter vs. „State of the Art“

Laut der Schlussfolgerungen aus dem obigen Abschnitt wird das Optimierungsverfahren MBO mit naivem Kriging Surrogatmodell als Referenz zu den Empfehlungen von Böck et al. (2012) betrachtet. Auch hier ist der Vergleich für alle EZE-Arten interessant. Allerdings ist dieser Vergleich nur für die WAV-Datenbank sinnvoll, da sich die Empfehlungen von Böck et al. (2012) auf diese Datenbank stützen. Folgende Nullhypothesen werden getestet:

- H_{AF1}^{online}: $Boeck\_wav\_online^{(1)} = MBO\_naivKM\_wav\_online^{(2)}$,
  - p-Wert = 0.8430 (nicht ablehnen);
- $H_{AF1}^{ps.online}$: $Boeck\_wav\_ps.online^{(1)} = MBO\_naivKM\_wav\_ps.online^{(2)}$,
  - p-Wert = 0.3659 (nicht ablehnen);
- $H_{AF1}^{offline}$: $Boeck\_wav\_offline^{(1)} = MBO\_naivKM\_wav\_offline^{(2)}$,
  - p-Wert = 0.9941 (nicht ablehnen).

Für diese Fragestellung kann keine der Nullhypothesen abgelehnt werden. Bei der Stichprobe *Boeck_wav_offline* handelt es sich um die bessere Einstellung mit $onset.shift = 0$ s. Die MBO-Ergebnisse scheinen also nicht schlechter zu sein als das Referenzverfahren von Böck et al. (2012). Anzumerken ist allerdings, dass pro Optimierungslauf immer eine andere optimale Parametereinstellung gefunden wird. Eine detailliere Analyse der optimalen Einstellungen findet in Abschnitt 8.1.6 statt. Weiterhin ist darauf hinzuweisen, dass die Empfehlungen von Böck et al. (2012) auf einer von den Autoren durchgeführten internen Optimierung basieren und deshalb gerade auf der WAV-Datenbank gute Ergebnisse erwartet werden. Wie Abbildung 8.2 zeigt, liefern diese Empfehlungen auf anderen Daten (MIDI-Datenbank) zwar immer noch zufriedenstellende F-Werte, sind aber (erwartungsgemäß) den auf dieser Datenbank optimierten Einstellungen deutlich unterlegen.

8.1.3 Anwendungsfrage 2: online EZE vs. offline EZE

Der Vergleich der EZE-Arten wird auf den beiden Datensätzen für die Optimierungsstrategie MBO mit naivem Kriging Surrogatmodell vollzogen. Dabei werden *online* vs. *pseudo-online* EZE und *pseudo-online* vs. *offline* EZE getestet. Die Testhypothesen mit entsprechenden p-Werten und Testentscheidungen lauten wie folgt:

- $H_{AF2}^{online/ps.online.WAV}$: $MBO\_naivKM\_wav\_online^{(3)} = MBO\_naivKM\_wav\_ps.online^{(3)}$,
  - p-Wert = $5.633 \cdot 10^{-07}$ (**ablehnen**);
- $H_{AF2}^{ps.online/offline.WAV}$: $MBO\_naivKM\_wav\_ps.online^{(4)} = MBO\_naivKM\_wav\_offline^{(3)}$,
  - p-Wert = 0.3980 (nicht ablehnen);
- $H_{AF2}^{online/ps.online.MIDI}$: $MBO\_naivKM\_midi\_online^{(2)} = MBO\_naivKM\_midi\_ps.online^{(2)}$,
  - p-Wert = 0.0023 (**ablehnen**);
- $H_{AF2}^{ps.online/offline.MIDI}$: $MBO\_naivKM\_midi\_ps.online^{(3)} = MBO\_naivKM\_midi\_offline^{(2)}$,
  - p-Wert = 0.8087 (nicht ablehnen).

Unter Berücksichtigung der Tabelle B.1 beträgt das lokale Signifikanzniveau für die Hypothese $H_{AF2}^{online/ps.online.WAV}$ $\alpha/5 = 0.01$ und für die Hypothese $H_{AF2}^{online/ps.online.MIDI}$ $\alpha/4 = 0.0125$. Die beiden Hypothesen können also nach dem Bonferroni-Holm-Verfahren abgelehnt werden.

Der Wilcoxon-Rangsummentest bestätigt die Beobachtung aus Abbildung 8.1, dass die *pseudo-online* EZE deutlich besser als die *online* Variante abschneidet. Die Umskalierung der Signalamplituden bewirkt die Umskalierung der spektralen Amplituden, so dass die spektralbasierten Merkmale meistens in einem anderen Zahlenintervall liegen als ohne eine solche Umskalierung. Offenbar funktioniert die Einsatzzeitlokalisierung in dem *pseudo-online* Fall besser, obwohl sie von der Skalierung der Merkmale unabhängig sein sollte. Interessant ist, dass für die MIDI-Datenbank dieser Unterschied zwar auch signifikant, jedoch nicht mehr so stark ausgeprägt ist.

Wenn man die maximale Amplitude des Signals frühzeitig und verlässlich schätzen kann, ergibt sich ein großes Verbesserungspotenzial für die *online* EZE. Man könnte also mit den optimalen Parametereinstellungen die Erkennung im *online* Fall starten, nach wenigen Sekunden die maximale Amplitude schätzen, weitere Amplitudenwerte anschließend durch den geschätzten Betrag teilen und auf die hier gefundenen optimalen Einstellungen der *pseudo-online* Erkennung umschalten. Eventuell sollte man die Amplitudenschätzung in bestimmten Zeitperioden aktualisieren, da sich die Lautstärke des Signals ändern kann. Dies wird in dem Ausblick dieser Arbeit noch einmal aufgegriffen.

8.1.4 Anwendungsfrage 3: WAV-Datenbank vs. MIDI-Datenbank

Die dritte Anwendungsfrage dient der Untersuchung der Musikdatenbank-bedingten Unterschiede in den erreichten optimalen F-Werten. Konkret wird hier die Performance der MBO auf der WAV- und der MIDI-Datenbank verglichen. In Abbildung 8.3 ist die Verteilung der optimalen F-Werte, die mit der Optimierungsmethode MBO mit naivem Kriging Surrogatmodell in 30 Wiederholungen gefunden wurden, für die beiden Datenbanken sowie alle EZE-Arten dargestellt.

Es ist deutlich zu sehen, dass auf der MIDI-Datenbank bessere F-Werte erreicht werden können. Folgende Nullhypothesen werden mit Hilfe des Wilcoxon-Rangsummentests analysiert:

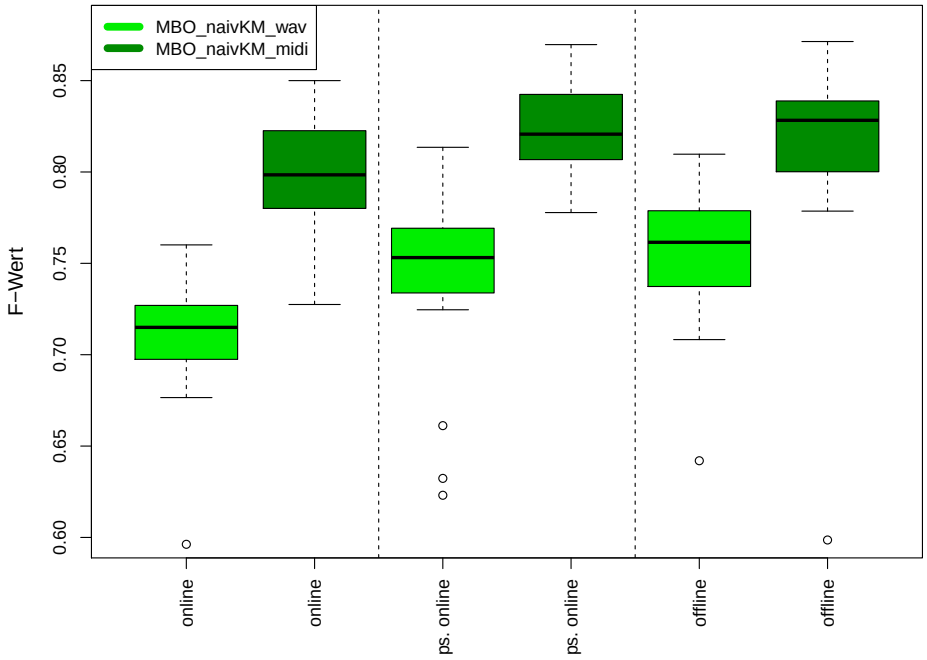

Abbildung 8.3: *Gegenüberstellung der optimalen F-Werte, die bei der Optimierungsstrategie MBO mit naivem Kriging Surrogatmodell jeweils auf der WAV- und MIDI-Datenbank erreicht wurden.*

- H_{AF3}^{online}: $MBO\_naivKM\_wav\_online^{(4)} = MBO\_naivKM\_midi\_online^{(3)}$,
  - p-Wert = $1.133 \cdot 10^{-15}$ (**ablehnen**);
- $H_{AF3}^{ps.online}$: $MBO\_naivKM\_wav\_ps.online^{(5)} = MBO\_naivKM\_midi\_ps.online^{(4)}$,
  - p-Wert = $2.050 \cdot 10^{-14}$ (**ablehnen**);
- $H_{AF3}^{offline}$: $MBO\_naivKM\_wav\_offline^{(4)} = MBO\_naivKM\_midi\_offline^{(3)}$,
  - p-Wert = $9.536 \cdot 10^{-10}$ (**ablehnen**).

Die p-Werte sprechen eindeutig für die Ablehnung der Nullhypothesen (auch unter Berücksichtigung des multiplen Niveaus). Somit können auf den MIDI-Stücken signifikant bessere F-Werte erreicht werden als auf den „per Hand“ annotierten Aufnahmen. Dieses Ergebnis erstreckt sich aber nur auf die betrachteten Datenbanken und könnte lediglich als eine begründete Vermutung für andere Datenbanken gelten. Als eine Erklärung soll erwähnt werden, dass eine künstlich generierte Aufnahme niemals die Komplexität einer echten Aufnahme erreichen wird, wodurch die Toneinsätze unter Umständen leichter zu erkennen sind. Dieses Phänomen wurde bereits in Bauer et al. (2010) untersucht: die spektrale Struktur der langen echten Klaviertöne zeigt in regelmäßigen Abständen eine starke Veränderung auf zeitlicher sowie spektraler Ebene, so dass diese (durch bestimmte Merkmale) fälschlicherweise als neue Toneinsätze erkannt werden. Für künstlich generierte Töne konnte dies jedoch nicht beobachtet werden.

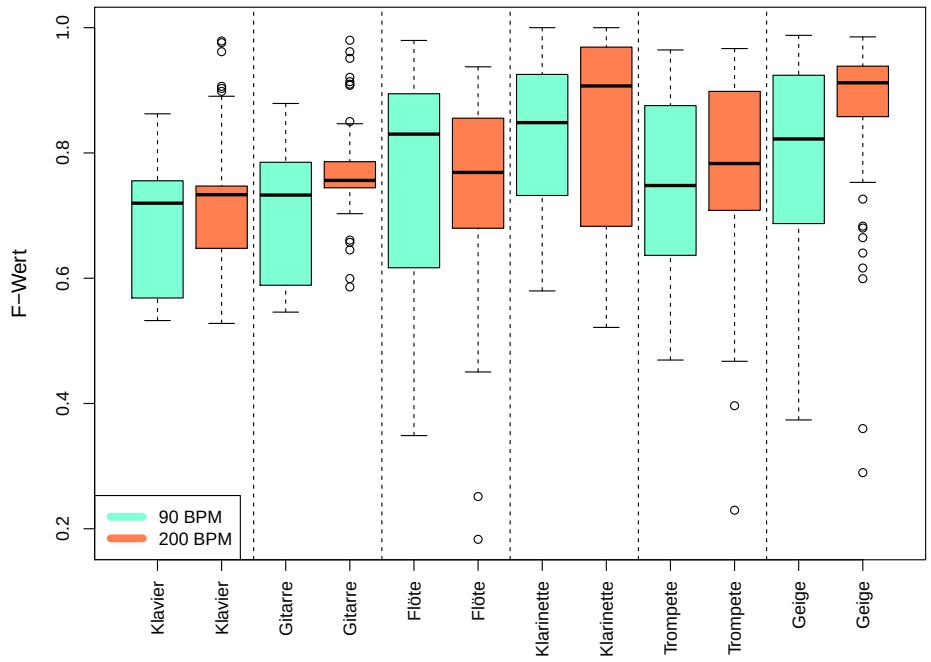

Abbildung 8.4: *Verteilung der F-Werte bei optimalen Parametereinstellungen für die Musikstücke der Volkslied-Datenbank.*

Die Anwendungsfrage 3 wird nun für die MIDI-Datenbank etwas detaillierter untersucht. Dabei handelt es sich um die Frage, ob musikalische Charakteristiken die Güte der EZE beeinflussen. Die in der MIDI-Datenbank enthaltene Volkslied-Datenbank (Abschnitt 5.2) ermöglicht eine Analyse des Einflusses von Tempo (schnell, langsam) und Musikinstrument (Klavier, Geige, Flöte, Klarinette, Trompete und Geige) auf die Verteilung der F-Werte. Zur Erinnerung, hier werden zwei Tonsequenzen mit den Tempi 90 BPM und 200 BPM für jedes Musikinstrument erzeugt. Pro Optimierungslauf werden dann F-Werte der optimalen Parametereinstellung für diese Musikstücke notiert. Abbildung 8.4 veranschaulicht die Verteilung dieser F-Werte über 30 Wiederholungen der MBO mit naivem und dummy Kriging Surrogatmodellen (zusammengefasst). Es ist etwas problematisch, Strukturen in dieser Abbildung zu erkennen. Nichtsdestotrotz können zwei Tendenzen beobachtet werden: Die Ergebnisse von Klavier und Gitarre fallen eher schlechter aus als die der drei betrachteten Blasinstrumente sowie Geige. Weiterhin scheinen die schnellen Tonabfolgen (200 BMP) besser abzuschneiden als die langsamen (90 BPM), wobei bei Flöte die umgekehrte Tendenz zu sehen ist.

Die erste Beobachtung kann durch das EZE-Merkmal erklärt werden. In Rahmen einer Vorstudie von Bauer et al. (2012) wurde der Einfluss der Merkmale auf die Güte der EZE für verschiedene Musikinstrumente untersucht. Dabei konnte festgestellt werden, dass spektralbasierte Merkmale für Klavier und Gitarre schlechtere Ergebnisse liefern als amplidudenbasierte Merkmale. Vermutlich liegt das an dem stark ausgeprägten Amplitudenanstieg, der bei diesen beiden

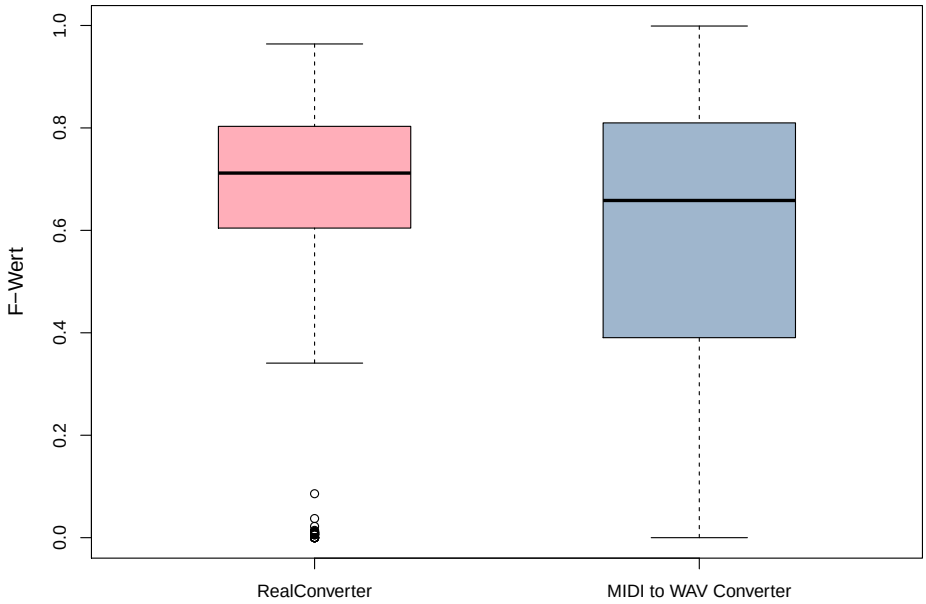

Abbildung 8.5: *Verteilung der F-Werte bei optimalen Parametereinstellungen für die Musikstücke der Musikepochen-Datenbank. Unterschieden wird zwischen mittels des RealConverters generierten Stücke und mittels des MIDI to WAV Converters generierten Aufnahmen.*

Instrumenten während eines Tonanschlages stattfindet. Bei Blasinstrumenten und Geige, dagegen, waren die spektralbasierten Merkmale den amplitudenbasierten stark überlegen. Da die besten F-Werte mit dem spektralbasierten Merkmal 'spektraler Fluss' ermittelt wurden, können die schlechteren Ergebnisse von Klavier und Gitarre gegenüber den anderen Instrumenten dadurch begründet sein.

Die zweite Beobachtung könnte mit der Konstruktion des F-Wertes und den damit verbundenen Nachteilen (Abschnitt 3.9.1) zusammenhängen. Da bei schnelleren Stücken mehr Toneinsätze pro Sekunde vorkommen (ca. 3), liegen die außerhalb des Toleranzintervalls geschätzten Einsätze womöglich direkt in den Toleranzintervallen der benachbarten Toneinsätze, so dass sie nicht mehr als falsch negative Einsätze gezählt werden. Auf eine genauere Untersuchung dieser Vermutung wird allerdings in dieser Arbeit verzichtet.

Die 22 Musikstücke der Musikepochen-Datenbank (s. Abschnitt 5.2), die ebenso in der MIDI-Datenbank enthalten sind, wurden einmal mittels *RealConverter* (mit echten Tönen) und einmal mittels *MIDI to WAV Converter* (mit synthetischen Tönen) generiert. Dadurch bietet sich der Vergleich der F-Werte auf diesen Stücken besonders an. Das Vorgehen ist hier analog zu der obigen Instrument-Tempo-Analyse. Abbildung 8.5 zeigt die Verteilung der F-Werte mit den jeweiligen optimalen EZE-Einstellungen über 60 Läufe (MBO mit naivem und dummy Kriging Surrogatmodellen).

Die F-Werte von den Aufnahmen, die mittels echter Töne generiert wurden, sind leicht besser als die der rein synthetischen Stücke. Das scheint der oben formulierten Vermutung zu widersprechen, dass die synthetischen Stücke wegen nicht perfekter Tonmodellierung grundsätzlich bessere EZE-Güte aufweisen sollten als die echten Stücke. Allerdings soll verdeutlicht werden, dass die mittels echter Töne generierten Stücke den Klang der echten Aufnahmen zwar sehr gut approximieren könnten, viele Aspekte wie Legato (Spielen der Tonabfolge ohne akustischer Unterbrechungen) aber nicht wiedergeben können. Vermutlich haben solche Aufnahmen zwei Vorteile: einzelne Töne besitzen die Klangeigenschaften der echten Töne und die Töne haben einen eindeutig definierten Anfang- und Endzeitpunkt.

8.1.5 Analyse der MBO-Optimierungszeit

Es ist nun von Interesse, wie die Gesamtzeit der Optimierung auf die drei wichtigsten Komponenten verteilt ist: Anpassung des Surrogatmodells, Optimierung des EI-Kriteriums und Funktionsauswertungen (s. Abschnitt 6.2). In Tabelle 8.1 ist die mittlere Zeit für einen MBO-Lauf in Stunden angegeben. Dabei wird die Zeit der Funktionsauswertungen für das Startdesign nicht berücksichtigt. Es geht hier also nur um die Proportion innerhalb der sequentiellen MBO-Schritte. Die Rechenzeitanalyse erfolgt hier lediglich auf der WAV-Datenbank.

Es wurde bereits oben bemerkt, dass die Surrogatmodellanpassung und EI-Optimierung für das dummy Kriging Modell spürbar mehr Zeit kosten als für das naive Kriging Modell. Anteilig gesehen werden im *online* Fall mit naivem Kriging Surrogatmodell 4.6% der Gesamtzeit für Modellanpassung, 10.8% für EI-Optimierung und 84.6% für Funktionsauswertungen gebraucht. Im *offline* Fall ist die Rechenzeit wie folgt verteilt: 9.8% für Modellanpassung, 19.2% für EI-Optimierung und 71% für Funktionsauswertung.

Tabelle 8.1: Mittlere Zeit in Stunden pro MBO-Optimierungslauf für Anpassung des Surrogatmodells, EI-Optimierung und Funktionsauswertungen. Grundlage der Optimierung: WAV-Datenbank.

EZE-Art	*Modellanpassung*		*EI-Optimierung*		*Funktionsauswertungen*	
	naivKM	*dummyKM*	*naivKM*	*dummyKM*	*naivKM*	*dummyKM*
online	1.866	3.602	4.348	6.081	34.214	33.274
offline	3.028	5.251	5.922	7.942	21.876	23.580

Auffällig ist zum einen die gesunkene Zeit für Funktionsauswertungen im *offline* EZE, welche durch tendenziell zeit-günstigere Parametereinstellungen während der Optimierung erklärt werden kann. Der maßgebende Parameter an dieser Stelle ist die Sprungweite: je größer h ist, desto weniger Fenster werden gebildet (Folge: schnellere Funktionsauswertungen). Auch die Anwendung des spektralen Filters und der Logarithmierung könnte die Zeit relevant beeinflussen.

Die Laufzeit für Funktionsauswertungen sollte unabhängig von dem Surrogatmodell sein. Nichtsdestotrotz sind relevante Unterschiede zwischen den beiden Modellen zu sehen. Dies könnte wiederum durch eher zeit-günstigere Parametereinstellungen beim Verwenden einer Methode gegenüber der anderen begründet sein.

8.1.6 Analyse der optimalen Parametereinstellungen

In diesem Abschnitt wird eine Analyse der gefundenen optimalen Parametereinstellungen von MBO für die univariate Einsatzzeiterkennung durchgeführt. Dabei werden MBO mit den beiden Surrogatmodellen sowie alle drei EZE-Arten gleichzeitig betrachtet. Auf der WAV-Datenbank wird diese Analyse jedoch etwas detaillierter erfolgen als auf der MIDI-Datenbank. Anschließend werden unter Berücksichtigung einiger wichtiger Parameter sowie F- und D-Werten Empfehlungen für die *online*, *pseudo-online* und *offline* EZE für die beiden Datenbanken gegeben.

8.1.6.1 WAV-Datenbank

Als erstes werden hier optimale Parametereinstellungen von MBO auf der Grundlage der WAV-Datenbank analysiert. Es werden die einzelnen Schritte des EZE-Schemas (s. Algorithmus 3.1 in Abschnitt 3.2) behandelt und die Verteilungen der optimalen Einstellungen graphisch dargestellt.

Fensterung des Signals. Hier sind zwei Parameter von Interesse: Fensterlänge N und Sprungweite h (Abschnitt 3.3). Die Fensterlänge ist besonders für *online* Anwendungen ein wichtiger Gesichtspunkt, denn sie gibt die Mindestlatenzzeit des Verfahrens wieder. Böck et al. (2012) empfehlen $N = 2048$ Abtastwerte, was ca. 46 ms bei einer Abtastrate von 44.1 kHz entspricht. Diese Zeit soll idealerweise verkürzt werden, so dass eher kleinere Fensterlängen von Interesse sind. In Abbildung 8.6 ist die Verteilung der optimalen Einstellungen für N über 30 Läufe der jeweiligen Verfahren dargestellt. Hier und in vielen weiteren Abbildungen ist auf der x-Achse das gesamte Optimierungsintervall dargestellt, auch wenn einige Ausprägungen gar nicht ausgewählt werden.

Aus Abbildung 8.6 ist ersichtlich, dass für *online* Anwendungen auch sehr kleine Fensterlängen (N = 512) gelegentlich ausgewählt werden, allerdings kommt die Einstellung N = 2048 am häufigsten vor. Für die *pseudo-online* und *offline* EZE wird die Fensterlänge von 2048 Abtastwerten ebenso häufig als optimal ausgewählt. Weiterhin ist zu bemerken, dass hier hinsichtlich der Verteilung der optimalen Einstellungen kein systematischer Unterschied zwischen den beiden Surrogatmodellen zu sehen ist.

In Abbildung A.5 im Anhang ist die Verteilung des Parameters Sprungweite dargestellt. Im *online* Fall kommen bei beiden Surrogatmodellen die Werte aus dem Intervall $[200, 600]$ und im *offline* Fall aus dem Intervall $[400, 800]$ am häufigsten vor. Bei der *pseudo-online* EZE gibt es jedoch einen großen Unterschied zwischen naivem Kriging und dummy Kriging: im ersten Fall konzentrieren sich die optimalen Einstellungen hauptsächlich im Intervall $[200, 400]$, während sie im zweiten Fall um das Intervall $[400, 600]$ symmetrisch verteilt sind. Ob für *online* Anwendungen

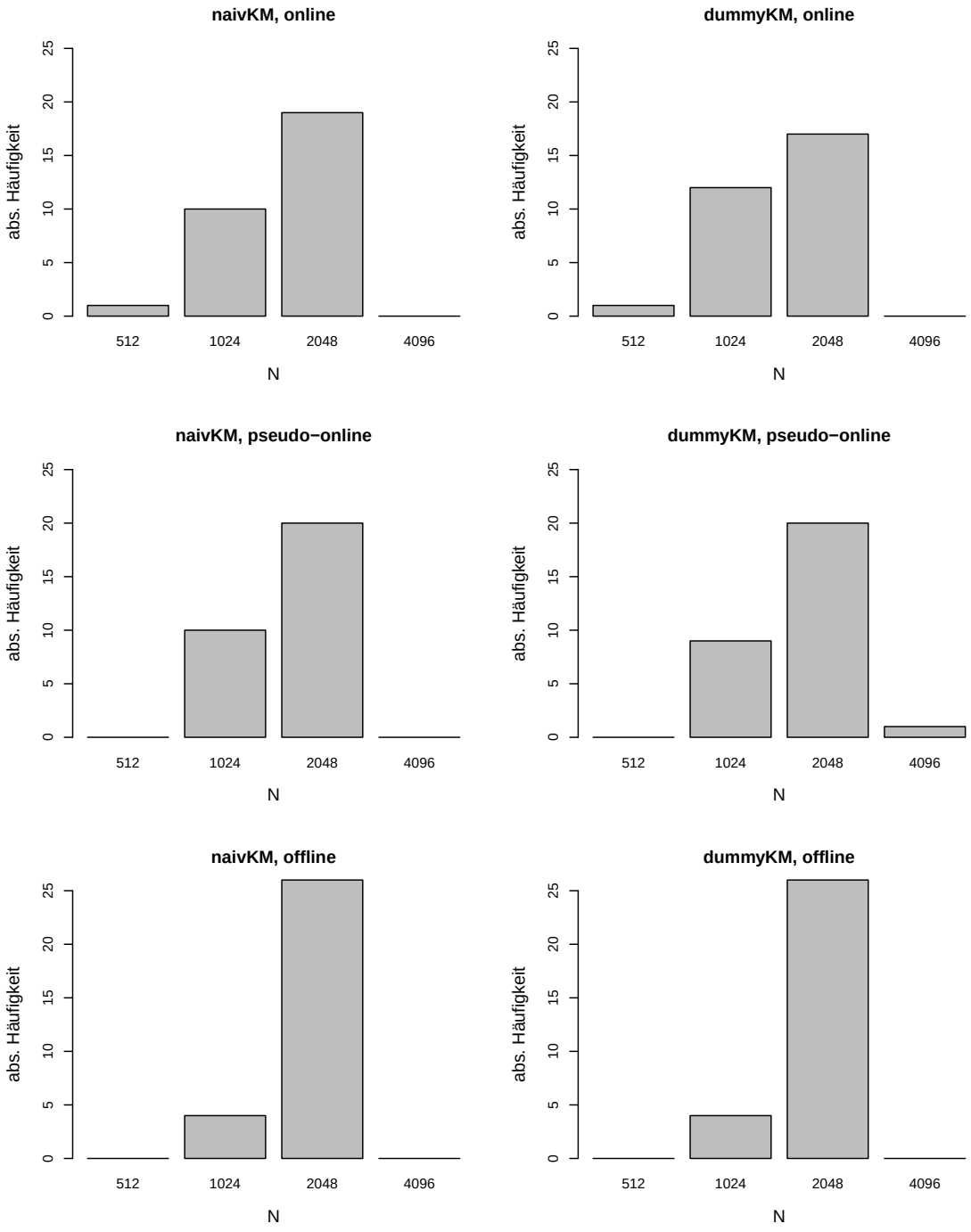

Abbildung 8.6: *Verteilung der optimalen Einstellungen für den Parameter N (Fensterlänge) in Anzahl der Abtastwerte. Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

eher größere oder kleinere Sprungweiten von Vorteil sind, lässt sich nicht allgemein beantworten. Einerseits, wenn Sprungweite klein ist und für ein Merkmal ein Fenster in die Zukunft „geschaut“ werden sollte, ist eine kleine Sprungweite bezüglich der Latenzzeit vorteilhaft. Andererseits aber erhöht sich der Rechenaufwand, denn je kleiner die Sprungweite, desto mehr überlappende Fenster pro Sekunde werden erzeugt. Wenn in jedem Fenster ein oder mehrere Merkmale bestimmt werden sollen, spiegelt sich eine große Anzahl an Fenstern negativ in der Bearbeitungszeit wieder. Die richtige Wahl ist somit ein Trade-off zwischen der Verzögerung verursacht durch das Abwarten der nötigen Signalinformation (bei großen Sprungweiten) und der Verzögerung verursacht durch die Rechenzeit des Algorithmus (bei kleinen Sprungweiten).

Vorverarbeitung des Signals. Zu den Vorverarbeitungsschritten zählen die Anwendungen einer Fensterfunktion auf die Signalamplitude, die spektrale Filterung und die Logarithmierung der spektralen Amplitude (Abschnitt 3.4). Die Verteilung der optimalen Ausprägungen des Parameters *window.fun* (Fensterfunktion) ist in Abbildung A.6 im Anhang dargestellt. Während bei MBO mit naivem Kriging für alle EZE-Arten die Hanning Fensterfunktion deutlich dominiert, kommt bei dummy Kriging auch die Blackman Fensterfunktion relativ oft vor. In keinem der Fälle wird die Rectangular Fensterfunktion (d.h. unverändertes Signal) ausgewählt.

Die Filterung des Spektrums (Parameter *spec.filter*) scheint laut Abbildung A.7 nutzbringend zu sein, was bei MBO mit dummy Kriging deutlicher zu sehen ist als bei MBO mit naivem Kriging. Auch wird die Dominanz der Filterung von der *online* zur *offline* EZE ausgeprägter. Die Anwendung des spektralen Filters benötigt zwar zusätzliche Rechenzeit, reduziert sie aber bei Berechnung der EZE-Merkmale, da deutlich weniger Frequenzlinien berücksichtigt werden müssen. Nichtsdestotrotz kann die Filterung des Spektrums einen zusätzlichen Rechenaufwand mit sich bringen, was für *online* Anwendungen immer diskutiert werden sollte. Der größere Zeitaufwand für Funktionsauswertungen bei dummy Kriging Modell in Tabelle 8.1 könnte z.B. durch häufigere Anwendung der Filterung erklärt werden.

Bei der Logarithmierung des Spektrums (Parameter *spec.log*) ergibt sich ein etwas differenzierteres Bild als bei der Filterung (s. Abbildung A.8): bei naivem Kriging für die *online* und *pseudo-online* EZE sind die Anteile von „Ja“ und „Nein“ relativ ausgeglichen. Bei den anderen Optimierungsstrategien ist ersichtlich, dass die Logarithmierung eher angewendet werden sollte.

Die Verteilung der optimalen Einstellungen des Komprimierungsparameters der Logarithmierung ℓ, welche in Abbildung A.9 dargestellt ist, lässt vermuten, dass dieser Parameter keinen Einfluss auf das Endergebnis hat. Die optimalen Werte sind nämlich fast in allen Fällen relativ gleich auf das zulässige Intervall verteilt.

Berechnung eines EZE-Merkmals in jedem Fenster. Einer der wichtigsten Bausteine der EZE ist das verwendete Merkmal. In dieser Arbeit werden 18 Merkmale vorgestellt und als Ausprägungen des Parameters *odf* zugelassen. In Abbildung 8.7 ist nun die Verteilung der in den jeweiligen MBO Läufen als optimal gefundenen Merkmale veranschaulicht. Im Grunde handelt

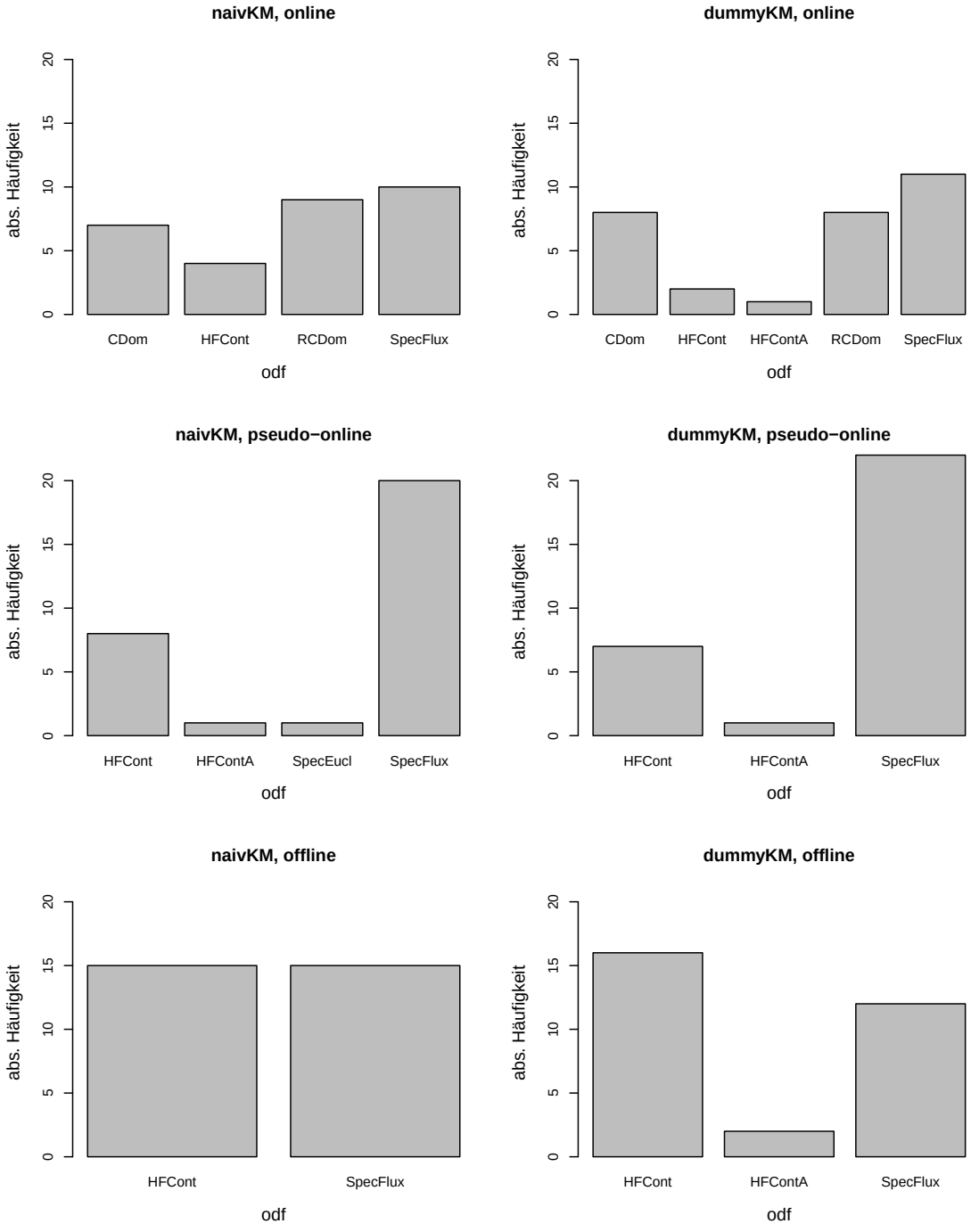

Abbildung 8.7: *Verteilung der optimalen Einstellungen für den Parameter odf (EZE-Merkmal). Merkmale auf der x-Achse: CDom (Compl.Domain), RCDom (Rect.Compl.Domain), HFCont (High.Freq.Cont.Diff), HContA (High.Freq.Cont.Abs.Diff), SpecFlux (Spec.Flux) und SpecEucl (Spec.Euclid), s. Abschnitt 3.5. Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

es sich um drei spektral basierte Merkmalspaare (s. Abschnitte 3.5.2, 3.5.3 und 3.5.4): ‘gewichtete spektrale Energie’ (einmal mit und einmal ohne Berücksichtigung der Tonende-Information), ‘spektraler Fluss’ und ‘spektraler euklidischer Abstand’ (Berücksichtigung der Tonende-Information) sowie ‘komplexe Domäne’ (Berücksichtigung der Tonende-Information) und ‘gerichtete komplexe Domäne’. Insgesamt scheinen die Merkmale ‘spektraler Fluss’ und ‘gewichtete spektrale Energie’ (beide ohne Berücksichtigung der Tonende-Information) am erfolgreichsten zu sein. Am günstigsten bezüglich der Rechenzeit ist die Berechnung der gewichteten spektralen Energie und des ‘spektralen Flusses’, während das ‘komplexe Domäne’ Merkmal etwas zeitintensiver ist.

Normalisierung der Merkmalswerte. In Abschnitt 3.6 wurden verschiedene Normalisierungsmethoden vorgestellt, welche sich allerdings nicht für *online* Anwendungen eignen. Aus diesem Grund wurde lediglich die exponentielle Glättung der Merkmalswerte mit dem Parameter α in Betracht gezogen (s. Abschnitt 3.6). Die Verteilung der optimalen Werte dieses Parameters ist in Abbildung A.10 dargestellt. Dabei zeichnet sich über alle Optimierungsstrategien ein ähnliches Bild ab: bevorzugt werden eher größere α-Werte (im Intervall $[0.6, 1]$), was einer mittleren bis schwachen Glättung entspricht.

Schwellenwertfunktion. Die Berechnung der Schwellenwertfunktion hat fünf Parameter (s. Abschnitt 3.7). Zuerst ist interessant, welche der drei gleitenden Funktionen – Mittelwert, Median und Quantil – am häufigsten ausgewählt werden. Die zugehörigen Verteilungen sind Abbildung 8.8 zu entnehmen. Bei dem Surrogatmodell naives Kriging wird die Mittelwert-Funktion bevorzugt, wobei dies von *online* zu *offline* immer deutlicher wird. Bei dummy Kriging, dagegen, ist die Präferenz der Median-Funktion im *online* Fall eindeutig, während für die *pseudo-online* und *offline* EZE die Funktionen eher gleich verteilt ausgewählt werden.

Die Verteilung des Parameters *mov.par*, welcher je nach Schwellenwertfunktion entweder in die Berechnung des multiplikativen Parameters λ oder des p-Quantils eingeht, ist über alle sechs Verfahren eindeutig: Es werden eher kleinere Werte bevorzugt, meistens im Intervall $[0, 0.2]$. Die entsprechende Darstellung ist in Abbildung A.11 im Anhang zu finden. Die Verteilung des additiven Parameters der Schwellenwertfunktion lässt hingegen keine deutlichen Schlüsse auf deren optimalen Einstellungen ziehen: die Werte sind relativ gleich bzw. unsystematisch über den gesamten zulässigen Bereich verteilt (s. Abbildung A.12). Besonders im *pseudo-online* bzw. *offline* Fall ist diese Tatsache etwas unerwartet, da die Merkmalswerte nach der Amplituden-Umskalierung eher in einem schmaleren Bereich liegen sollten, so dass die Schwankung des additiven Parameters einen relevanten Effekt haben sollte.

Weiterhin interessant ist die Tatsache, dass die Verteilungen der Parameter $t(l_T)$ (Zeit in die Vergangenheit) und $t(r_T)$ (Zeit in die Zukunft, nur für die *offline* EZE) der Schwellenwertbestimmung wiederum sehr unsystematisch sind (s. Abbildungen A.13 und A.14 im Anhang). Die optimalen Werte streuen innerhalb der zulässigen Bereiche relativ gleichmäßig. Dies gilt für die

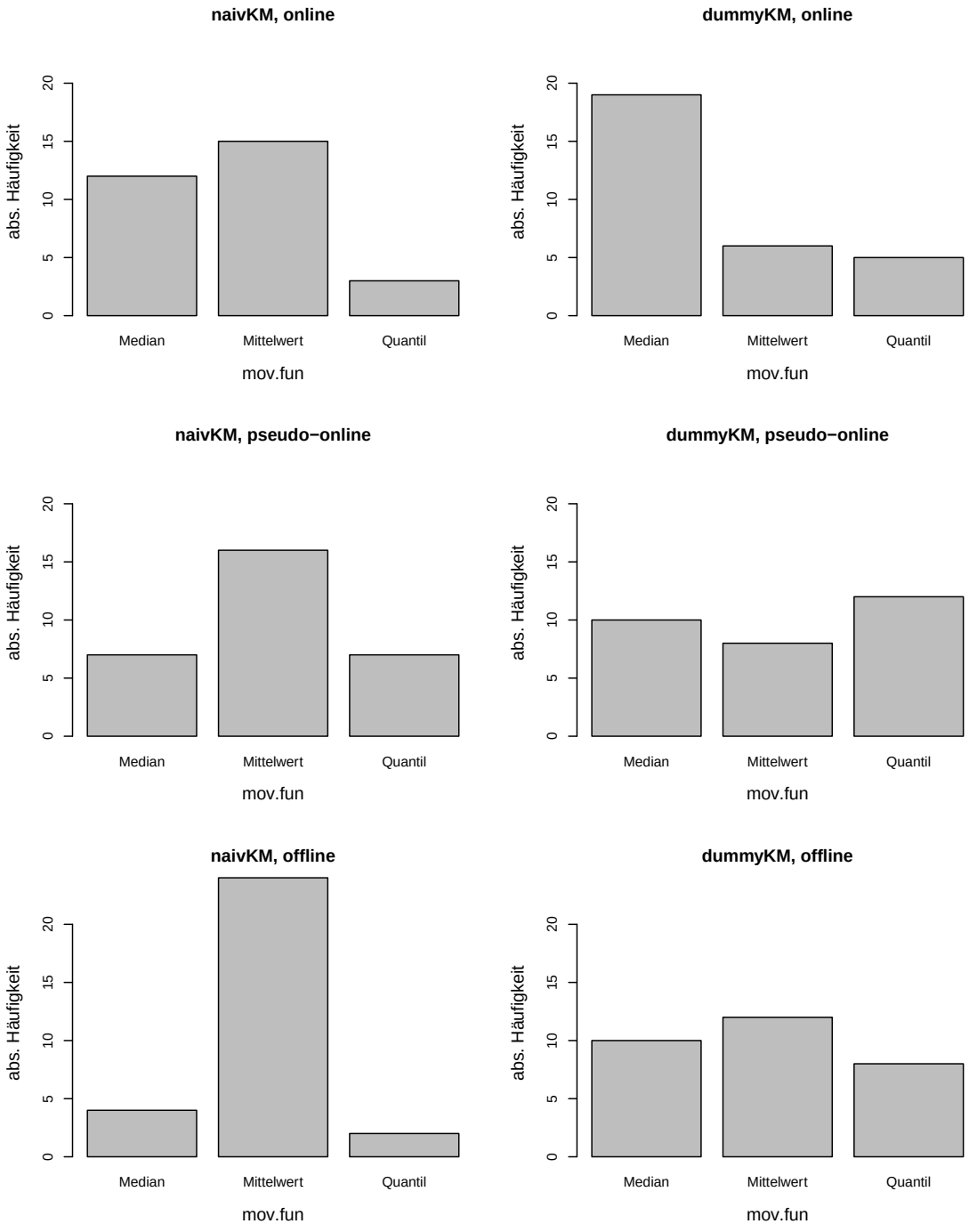

Abbildung 8.8: *Verteilung der optimalen Einstellungen für den Parameter mov.fun (Schwellenwertfunktion). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

beiden Surrogatmodelle und für alle Erkennungsarten. Es lässt sich somit vermuten, dass auch diese Parameter keinen wesentlichen Effekt auf die EZE-Güte haben. Aus Rechenzeitgründen würde man in diesem Fall eher kleinere Werte für die beiden Parameter bevorzugen.

Lokalisierung der Einsatzzeiten. In dem letzten Schritt der EZE werden die Zeiten der geschätzten Toneinsätze bestimmt. An dieser Stelle werden vier Parameter für die Optimierung herangezogen (s. Abschnitt 3.8). Die Verteilungen der Parameter $t(l_O)$ (Zeit in die Vergangenheit) und $t(r_O)$ (Zeit in die Zukunft, nur für die *offline* EZE) sind in Abbildungen A.15 und A.16 in Anhang veranschaulicht. Im Gegensatz zu den oben diskutierten Zeit-Parametern der Schwellenwertfunktion ist die Verteilung der optimalen Zeiten für die Einsatzzeitlokalisierung sehr eindeutig: die meisten Werte liegen im Intervall $[0, 0.1]$ s (für $t(l_O)$) bzw. ausschließlich in diesem Intervall (für $t(r_O)$). Das beutetet, dass das Fenster für die Bestimmung der lokalen Maxima eher klein sein sollte. Allerdings sollte dabei der Parameter $t(\mathit{min.dist})$, welcher die minimale Distanz zwischen zwei aufeinander folgenden Toneinsätzen vorgibt, in eher größerem Zeitbereich gewählt werden: im Intervall $[0.04, 0.05]$ s (s. Abbildung A.17). Diese Einschränkung scheint wichtig für die *online* und *pseudo-online* EZE zu sein. Im *offline* Fall spielt $t(\mathit{min.dist})$ offenbar keine wichtige Rolle mehr, was sich in seiner gleichmäßigen Verteilung über den zulässigen Bereich widerspiegelt.

Anschließend wird untersucht, ob die Verschiebung der geschätzten Toneinsätze einen Vorteil bringt. Die Verteilung der optimalen Einstellungen des Parameters *onset.shift* kann Abbildung 8.9 entnommen werden. Für die *online* und *pseudo-online* EZE weisen die Verteilungen zwar keine deutlichen Spitzen auf, insgesamt liegen die meisten Werte aber im Intervall $[-0.005, 0.005]$ s, was einer sehr kleinen bis keiner Verschiebung entspricht. Für die *offline* EZE, hingegen, ist eine negative Verschiebung aus dem Intervall $[-0.01, -0.005]$ s ratsam. Dies kann damit erklärt werden, dass durch die Berücksichtigung des zukünftigen Signals die „wahren“ lokalen Maxima des Merkmals gefunden werden, die allerdings etwas später als der wahre Toneinsatz vorkommen.

Empfehlung der optimalen EZE-Einstellungen für die WAV-Datenbank. Oben wurden die optimalen Einstellungen aller Parameter über 30 Läufe der jeweiligen Verfahren unabhängig voneinander diskutiert. Nun sollen drei Parametersets (eine pro EZE-Art) identifiziert werden, die als Empfehlung für die weiteren Anwendungen (wie Optimierung der Hörgeräte oder Musiktranskription) gelten sollen. Momentan liegen pro Verfahren 30 als optimal gefundene Einstellungen mit zugehörigen mittleren F-Werten auf den Testdaten vor. Dabei können diese Einstellungen nicht miteinander verglichen werden, weil sie auf unterschiedlichen Testdaten ermittelt wurden. Aus diesem Grund werden sie auf allen Musikstücken der WAV-Datenbank erneut validiert und die zugehörigen mittleren F-Werte miteinander verglichen. Die mittleren D-Werte werden auch zusätzlich notiert.

In Tabelle 8.2 werden die 5 erfolgreichsten Parametereinstellungen für *online*, *pseudo-online* und *offline* Erkennung vorgestellt, wobei die beiden Surrogatmodelle gleichzeitig betrachtet werden. Je nach Schwellenwertfunktion wird direkt entweder der multiplikative Parameter λ oder das p-Quantil angegeben (anstatt der *mov.par* Einstellung). ‘Spektraler Fluss’ erweist sich dabei

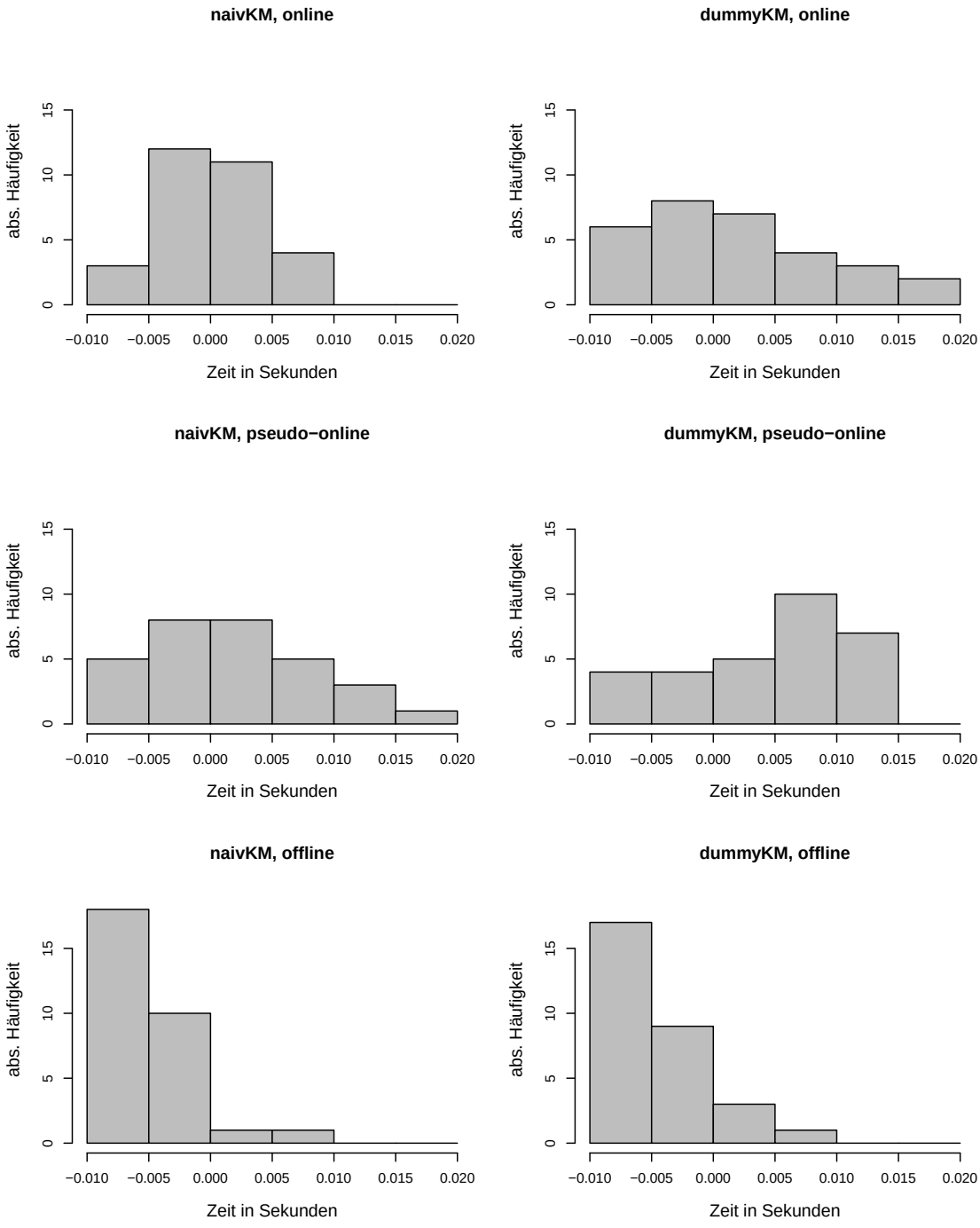

Abbildung 8.9: *Verteilung der optimalen Einstellungen für den Parameter onset.shift (Verschiebung der geschätzten Toneinsätze) in Sekunden. Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit Dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

als das beste EZE-Merkmal für alle Erkennungsarten. Weiterhin wird die spektrale Filterung und Logarithmierung bei allen besten Parametereinstellungen angewendet. Der additive Parameter der Schwellenwertfunktion der besten Einstellungen streut nicht mehr sehr breit innerhalb des zulässigen Intervalls (wie das aus Abbildung A.12 zu sehen war), sondern konzentriert sich hauptsächlich in $[0,2]$ Intervall.

Der klare Favorit für die *online* EZE ist die erste Parametereinstellung, welche einen deutlich besseren F- sowie D-Wert als die nachfolgenden Einstellungen aufweist. Die besten Einstellungen (außer Median als Schwellenwertfunktion) ähneln sehr den empfohlenen Einstellungen von Böck et al. (2012) (s. Abschnitt 8.1). Es kommt lediglich die relativ große Fensterlänge von 2048 Abtastwerten bei den erfolgreichen *online* EZE-Einstellungen vor, was nicht vorteilhaft für die Latenzzeit des Verfahrens ist. Umso bedeutender ist es, dass die beste Einstellung der *pseudo-online* Erkennung (bezüglich des F- und D-Wertes) eine kleinere Fensterlänge von 1024 Abtastwerten benötigt. Somit spiegelt sich die Amplituden-Umskalierung nicht nur in signifikant besseren F-Werten wider, sondern verkürzt deutlich die Latenzeit der Einsatzzeiterkennung. Dies ist ein weiteres wichtiges Argument, um eine adaptive Schätzung der Signalamplitude bei echten *online* Anwendungen zu implementieren. Die zweitbeste *pseudo-online* EZE-Einstellung ist die einzige der top fünf Einstellung, welche das gleitende Quantil (vorgeschlagen in dieser Arbeit) als Schwellenwertfunktion verwendet.

Was die *offline* Erkennung angeht, so scheinen die drei ersten Einstellungen bezüglich ihrer F-Werte sehr ähnlich zu sein, wobei die dritte Einstellung bezüglich des D-Wertes am besten abschneidet und die zweite die größte Sprungweite hat (was weniger Rechenzeit bedeutet). Allgemein wird hier die erste Einstellung empfohlen, wobei je nach Anwendungsanforderungen auch andere Einstellungen sinnvoll sein könnten.

Es stellt sich die Frage, wie sensibel die empfohlenen Einstellung gegenüber kleinen Veränderungen sind. Es ist also interessant, ob sich z.B. der mittlere F-Wert stark ändert, wenn anstatt von $h = 389$ Abtastwerten $h = 441$ verwendet werden. Die Beantwortung dieser Fragen bedarf einer detaillierten Analyse, die auf einem zusätzlichen Datensatz erfolgen sollte, um eine Überanpassung zu vermeiden. Auch die Frage nach einer wesentlichen Verbesserung bzw. Verschlechterung der Performance sollte vorab geklärt werden (ob dafür z.B. statistische Test verwendet werden sollten). In dieser Arbeit wird daher auf eine solche Sensibilitätsanalyse verzichtet.

8.1.6.2 MIDI-Datenbank

Bevor hier die optimalen EZE-Einstellungen auf der MIDI-Datenbank analysiert werden, wird – wie bereits angekündigt – auf Abbildung 8.2 (s. Abschnitt 8.1.1) erneut eingegangen. Dort ist nämlich zusätzlich zu den Optimierungsergebnissen von MBO mit naivem und dummy Kriging als Surrogatmodelle sowie der Zufallssuche auch die Güte der Empfehlungen von Böck et al. (2012) und der in Unterabschnitt 8.1.6.1 formulierten Empfehlungen auf der MIDI-Datenbank veranschaulicht.

Tabelle 8.2: Die fünf besten Parametereinstellungen für *online*, *pseudo-online* und *offline* EZE, die mittels MBO mit naivem Kriging bzw. dummy Kriging als Surrogatmodell auf der WAV-Datenbank ermittelt wurden.

EZE	Sur. Modell	F-Wert	D-Wert	N	h	*window.fun*	*spec.filter*	*spec.log*	ℓ	*odf*	α	*mov.fun*	λ / p-Quantil	δ	$t(l_T)$	$t(r_T)$	$t(l_O)$	$t(r_O)$	$t(min.dist)$	*onset.shift*
online	*dummy*	0.763	0.012	2048	389	Hann	Ja	Ja	0.085	*SFlux*	0.699	Median	1.180	1.634	0.403	0	0.030	0	0.042	0.008
	naiv	0.754	0.015	2048	706	Hann.	Ja	Ja	6.791	*SFlux*	0.860	Mittel.	1.115	2.096	0.372	0	0.037	0	0.045	0.009
	naiv	0.744	0.015	2048	688	Hann.	Ja	Ja	17.89	*SFlux*	0.874	Mittel.	1.345	1.184	0.179	0	0.020	0	0.046	0.009
	dummy	0.743	0.013	2048	694	Black.	Ja	Ja	12.35	*SFlux*	0.802	Median	1.468	1.730	0.408	0	0.022	0	0.048	0.003
	dummy	0.743	0.014	2048	607	Black.	Ja	Ja	8.410	*SFlux*	0.772	Median	1.614	1.260	0.364	0	0.015	0	0.036	0.005
ps.-online	*naiv*	0.791	0.009	1024	644	Hann.	Ja	Ja	0.965	*SFlux*	0.845	Median	1.285	1.366	0.237	0	0.029	0	0.039	0.015
	dummy	0.786	0.010	2048	570	Black.	Ja	Ja	0.302	*SFlux*	0.788	Quantil	0.809	0.947	0.330	0	0.036	0	0.038	-0.003
	dummy	0.782	0.012	2048	606	Hann.	Ja	Ja	0.394	*SFlux*	0.953	Mittel.	1.175	0.630	0.311	0	0.093	0	0.044	0.006
	dummy	0.780	0.010	1024	626	Hann.	Ja	Ja	5.487	*SFlux*	0.627	Median	1.260	1.493	0.274	0	0.048	0	0.049	0.013
	dummy	0.780	0.011	2048	743	Hann.	Ja	Ja	1.747	*SFlux*	0.866	Median	1.309	2.359	0.275	0	0.034	0	0.032	0.007
offline	*dummy*	0.800	0.009	2048	563	Hann.	Ja	Ja	4.174	*SFlux*	0.771	Median	1.342	1.580	0.395	0.452	0.029	0.051	0.041	-0.009
	naiv	0.799	0.011	2048	813	Hann.	Ja	Ja	5.022	*SFlux*	0.913	Mittel.	1.139	1.547	0.289	0.215	0.014	0.062	0.027	-0.007
	dummy	0.799	0.009	2048	686	Hann.	Ja	Ja	10.02	*SFlux*	0.820	Median	1.243	1.975	0.316	0.248	0.013	0.030	0.004	-0.005
	dummy	0.798	0.010	1024	474	Hann.	Ja	Ja	2.515	*SFlux*	0.828	Mittel.	1.140	1.352	0.351	0.132	0.017	0.029	0.045	-0.006
	dummy	0.798	0.010	2048	787	Black.	Ja	Ja	4.074	*SFlux*	0.982	Median	1.277	2.675	0.116	0.250	0.018	0.050	0.012	0.001

Zum einen liefern die beiden Empfehlungen immer schlechtere F-Werte als die auf der MIDI-Datenbank optimierten Einstellungen. Zum anderen ist das Verhalten der in Unterabschnitt 8.1.6.1 empfohlenen Einstellungen sehr auffällig: genau so gut wie die Empfehlungen von Böck et al. (2012) bei der *online* EZE, deutlich schlechter bei *pseudo-online* und deutlich besser bei *offline*. Insgesamt deutet das darauf hin, dass die auf der WAV-Datenbank gefundenen besten Einstellungen nicht genau so gut auf anderen bzw. anders generierten Datensätzen zu funktionieren scheinen. Das könnte in diesem Fall an verschiedenen Arten der markierten Toneinsätze liegen – wahrnehmbare bzw. physikalische Toneinsätze (wie bereits in Abschnitt 5.2 erwähnt wurde).

Bei der Analyse der optimalen Einstellungen auf der MIDI-Datenbank wird auf die Veranschaulichung der Verteilung einzelner Algorithmusparameter verzichtet. Analog zu Tabelle 8.2 zeigt Tabelle 8.3 die besten Einstellungen des EZE-Algorithmus für die MIDI-Datenbank (für alle drei EZE-Arten). Auch hier wird fast ausschließlich die Fensterlänge von 2048 Abtastwerten, Hanning Fensterfunktion, 'spektraler Fluss' EZE-Merkmal, Anwendung der spektralen Filterung und der spektralen Logarithmierung über alle Erkennungsarten ausgewählt. Die optimalen Werte der numerischen Parameter der fünf besten Einstellungen scheinen sehr nah aneinander zu liegen, genauso wie die besten F-Werte. Die auffällig viel größeren D-Werte in Tabelle 8.3 (im Vergleich mit Tabelle 8.2) können dadurch erklärt werden, dass, bedingt durch seine Konstruktion, der D-Wert von der Dauer des Musikstückes und der Anzahl der Toneinsätze abhängt. Die MIDI-Datenbank schließt dabei ca. doppelt so viele Toneinsätze wie die WAV-Datenbank ein.

Weiterhin ist hervorzuheben, dass hier meistens der Median als gleitende Funktion für den Schwellenwert ausgewählt wird. Auch die optimalen Sprungweite-Werte sind deutlich höher gewählt als bei der WAV-Datenbank. Die optimalen Einstellungen des Parameters *onset.shift* liegen ausschließlich im negativen Bereich (obgleich auch sehr klein). Dies erklärt sich durch die Tatsache, dass bei den MIDI-Stücken die physikalischen Toneinsätze notiert sind, welche erst mit einer Verzögerung erkannt werden können.

Hier korrelieren die besten F- und D-Werte nicht mehr stark miteinander, im Gegensatz zu den Ergebnissen auf der WAV-Datenbank. So könnte im *online* Fall entweder die erste Parametereinstellung (mit dem besten F-Wert von 0.835 aber nur dem zweitbesten D-Wert von 0.180) oder die fünfte Einstellung (mit etwas schlechterem F-Wert von 0.830 aber dafür dem besten D-Wert von 0.173) empfohlen werden. Bei der *pseudo-online* EZE hat die drittbeste (bezüglich des F-Werts) Einstellung den Vorteil, dass die Fensterlänge mit 1024 Abtastwerten um die Hälfte kleiner als bei anderen Einstellungen ist und sie den besten D-Wert aufweist. Daher könnte sie für einige Anwendungen besonders interessant sein. Hinsichtlich der *offline* Erkennung scheint die erste Einstellung trotz des nicht besten D-Wertes empfehlenswert zu sein.

Tabelle 8.3: Die fünf besten Parametereinstellungen für *online*, *pseudo-online* und *offline* EZE, die mittels MBO mit naivem Kriging bzw. dummy Kriging als Surrogatmodell auf der MIDI-Datenbank ermittelt wurden.

EZE	Sur. Modell	F-Wert	D-Wert	N	h	*window.fun*	*spec.filter*	*spec.log*	ℓ	*odf*	α	*mov.fun*	λ / %-Quantil	δ	$t(l_T)$	$t(r_T)$	$t(l_O)$	$t(r_O)$	$t(min.dist)$	*onset.shift*
online	*naiv*	0.835	0.180	2048	821	Hann.	Ja	Ja	12.38	*SFlux*	0.743	Median	1.233	2.350	0.214	0	0.034	0	0.045	-0.007
	naiv	0.832	0.182	2048	846	Hann.	Ja	Ja	15.67	*SFlux*	0.842	Median	1.175	3.255	0.314	0	0.051	0	0.047	-0.004
	dummy	0.831	0.194	2048	590	Hann.	Ja	Ja	10.03	*SFlux*	0.832	Median	1.488	1.485	0.174	0	0.036	0	0.040	-0.008
	naiv	0.831	0.187	2048	764	Hann.	Ja	Ja	7.262	*SFlux*	0.745	Median	1.271	1.960	0.341	0	0.029	0	0.037	-0.006
	dummy	0.830	0.173	2048	702	Hann	Ja	Ja	8.266	*SFlux*	0.706	Median	1.253	2.714	0.184	0	0.059	0	0.045	-0.006
ps.-online	*dummy*	0.846	0.157	2048	654	Hann.	Ja	Ja	4.601	*SFlux*	0.840	Median	1.129	2.487	0.458	0	0.068	0	0.045	-0.009
	naiv	0.846	0.153	2048	687	Hann.	Ja	Ja	2.647	*SFlux*	0.785	Median	1.368	1.365	0.179	0	0.045	0	0.026	-0.007
	naiv	0.840	0.134	1024	602	Hann.	Ja	Ja	3.350	*SFlux*	0.896	Mittel.	1.203	1.930	0.105	0	0.060	0	0.039	-0.007
	naiv	0.840	0.150	2048	576	Hann.	Ja	Ja	3.741	*SFlux*	0.713	Mittel.	1.241	0.775	0.310	0	0.041	0	0.050	-0.005
	naiv	0.838	0.150	2048	655	Black.	Ja	Ja	2.431	*SFlux*	0.892	Median	1.240	2.337	0.184	0	0.059	0	0.027	-0.007
offline	*naiv*	0.844	0.149	2048	970	Hann.	Ja	Ja	17.35	*SFlux*	0.954	Median	1.163	2.790	0.210	0.179	0.070	0.052	0.003	-0.010
	dummy	0.840	0.148	2048	726	Hann.	Ja	Ja	3.281	*SFlux*	0.988	Median	1.671	1.631	0.278	0.204	0.086	0.008	0.043	-0.005
	dummy	0.839	0.158	2048	1103	Hann.	Ja	Ja	9.680	*SFlux*	0.873	Median	1.417	1.428	0.219	0.102	0.064	0.035	0.028	-0.007
	naiv	0.839	0.135	2048	875	Hann.	Ja	Ja	17.35	*SFlux*	0.972	Mittel.	1.217	2.062	0.391	0.318	0.069	0.061	0.045	-0.007
	naiv	0.839	0.134	2048	700	Hann.	Ja	Ja	1.996	*SFlux*	0.990	Median	1.212	3.342	0.303	0.188	0.038	0.074	0.031	-0.008

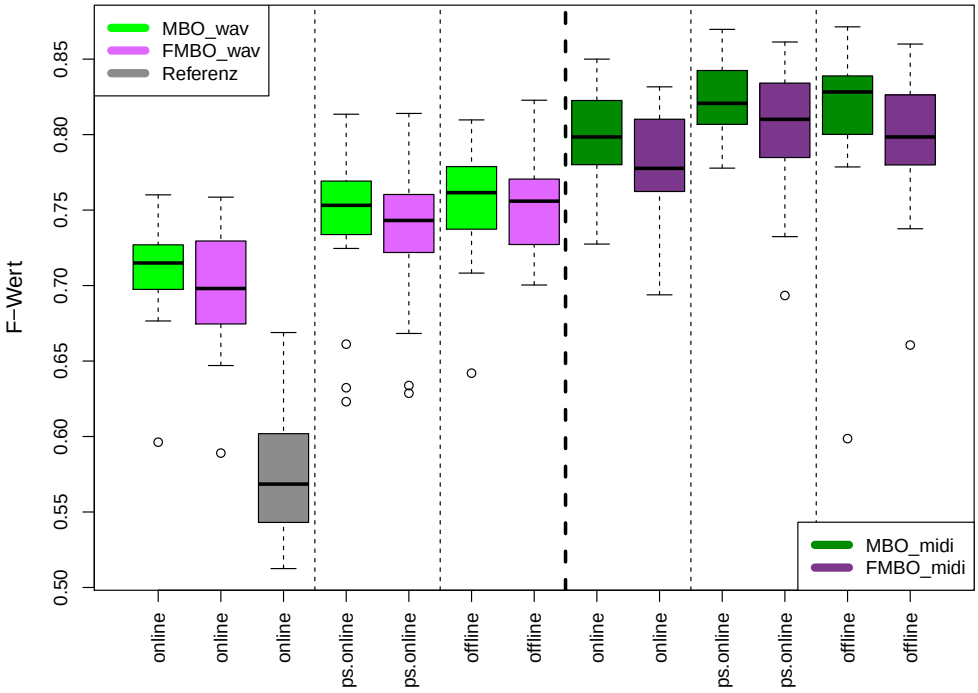

Abbildung 8.10: *Gegenüberstellung der Optimierungsergebnisse von MBO und FMBO mit naivem Kriging Surrogatmodell. Die linke sieben Boxplots entsprechen den Läufen auf der WAV-Datenbank (inklusive eines Referenzverfahrens für die online EZE) und die rechten sechs Boxplots den Läufen auf der MIDI-Datenbank.*

8.2 Zweite Stufe der Experimente und Optimierungsfrage 2: MBO vs. FMBO

In diesem Abschnitt soll die Effektivität der FMBO Strategie (instanzgebundene modellbasierte Optimierung, s. Abschnitt 6.3) überprüft werden. Es sind hier zwei Fragen interessant: wie groß ist die Zeitersparnis von FMBO gegenüber MBO und ob durch diese Ersparnis eine signifikante Verschlechterung der Optimierungsgüte zustande kommt.

Zuerst wird die zweite Fragestellung (Optimierungsfrage 2, s. Abschnitt 7.1) untersucht. Für diesen Zweck wird die FMBO Strategie mit Surrogatmodell naives Kriging zur Optimierung aller drei EZE-Arten auf den beiden Datensätzen herangezogen. Da es sich hierbei um das gleiche Validierungsschema wie bei MBO handelt (s. Abschnitt 6.4.2), sind die mittleren F-Werte auf den 30 Testdatensätzen direkt miteinander vergleichbar.

Abbildung 8.10 stellt die Ergebnisse der Experimente grafisch dar. Es ist dabei zu sehen, dass FMBO immer eine etwas schlechtere Performance aufweist als MBO. Der graue Boxplot zeigt das Ergebnis eines einfachen Referenzverfahrens für FMBO: es wird nur an den im Selektionsschritt ausgewählten k Stücken optimiert. Dieses Referenzverfahren ist konstruktionsbedingt schneller als alle anderen Optimierungsstrategien, liefert aber deutlich schlechtere F-Werte (hier nur für die *online* EZE auf der WAV-Datenbank veranschaulicht).

Mit Hilfe des Wilcoxon-Rangsummentests sollte die Frage untersucht werden, ob der Güteverlust von FMBO gegenüber von MBO (für jeden Paarvergleich) signifikant ist. Es ergeben sich somit sechs Nullhypothesen:

- $H_{OF2}^{online.WAV}$: $MBO\_naivKM\_wav\_online^{(5)} = FMBO\_naivKM\_wav\_online^{(1)}$,
  - p-Wert = 0.1870 (nicht ablehnen);
- $H_{OF2}^{online.MIDI}$: $MBO\_naivKM\_midi\_online^{(4)} = FMBO\_naivKM\_midi\_online^{(1)}$,
  - p-Wert = 0.0370 (nicht ablehnen);
- $H_{OF2}^{ps.online.WAV}$: $MBO\_naivKM\_wav\_ps.online^{(6)} = FMBO\_naivKM\_wav\_ps.online^{(1)}$,
  - p-Wert = 0.1970 (nicht ablehnen);
- $H_{OF2}^{ps.online.MIDI}$: $MBO\_naivKM\_midi\_ps.online^{(5)} = FMBO\_naivKM\_midi\_ps.online^{(1)}$,
  - p-Wert = 0.0800 (nicht ablehnen);
- $H_{OF2}^{offline.WAV}$: $MBO\_naivKM\_wav\_offline^{(5)} = FMBO\_naivKM\_wav\_offline^{(1)}$,
  - p-Wert = 0.3738 (nicht ablehnen).
- $H_{OF2}^{offline.MIDI}$: $MBO\_naivKM\_midi\_offline^{(4)} = FMBO\_naivKM\_midi\_offline^{(1)}$,
  - p-Wert = 0.0415 (nicht ablehnen).

Die p-Werte der Hypothesen $H_{OF2}^{online.MIDI}$ und $H_{OF2}^{offline.MIDI}$ sind kleiner als 0.05. Allerdings wird hier das Signifikanzniveau, wie in Abschnitt 7.2 beschrieben, adjustiert. Aus Tabelle B.1 geht hervor, dass die beiden p-Werte nicht ausreichend klein sind, um nach dem Bonferroni-Holm-Verfahren die entsprechenden Nullhypothesen abzulehnen. Der erste p-Wert sollte dafür nämlich kleiner als $\alpha/2 = 0.025$ und der zweite kleiner als $\alpha/3 = 0.0167$ sein.

Nichtsdestotrotz, die beiden kleinen p-Werte sowie die Beobachtungen aus Abbildung 8.10 lassen vermuten, dass es wohl eine Tendenz für leicht schlechtere Güte von FMBO gegenüber MBO gibt. Da FMBO lediglich als Approximation von MBO gesehen werden soll, ist dieser Güteverlust auch zu erwarten. Ob er relevant ist, sollte für jede einzelne Anwendung individuell entschieden werden.

Abbildung 8.11 zeigt die Zeitersparnis bei Funktionsauswertungen von FMBO gegenüber von MBO. Abgebildet ist dabei die summierte Zeit für Funktionsauswertungen während der sequentiellen Schritte eines Optimierungslaufs (Startdesign wurde ausgelassen). Es ist eine deutliche Zeitersparnis zu sehen: ca. 23 Stunden im Durchschnitt für die Optimierung auf der WAV-Datenbank und ca. 52 Stunden für die Optimierung auf der MIDI-Datenbank. Es bleibt somit eine Frage der Priorität – leicht besser Güte vs. relevante Zeitersparnis – wenn man sich für MBO bzw. FMBO als Optimierungsstrategie entscheidet.

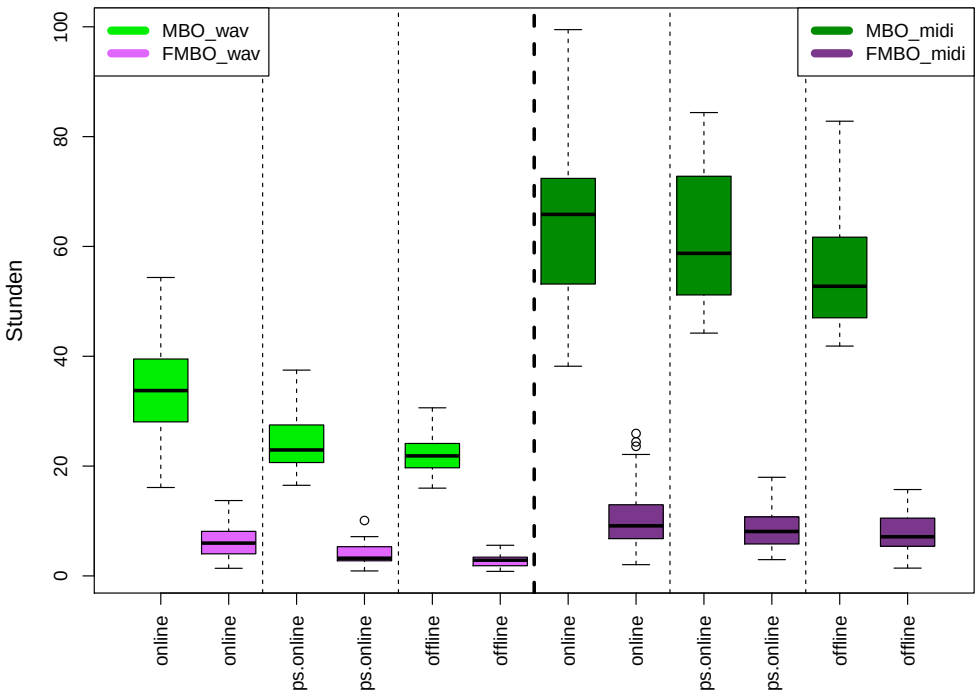

Abbildung 8.11: *Summierte Zeit in Stunden für Funktionauswertungen während der sequentiellen Schritte (d.h. ohne Berücksichtigung des Startdesigns) bei MBO und FMBO mit naivem Kriging Surrogatmodell. Die linken sechs Boxplots entsprechen den Läufen auf der WAV-Datenbank und die rechten sechs Boxplots den Läufen auf der MIDI-Datenbank.*

Die Zeitersparnis hängt zum größten Teil von der Anzahl der „guten“ Punkte ab, in denen eine Komplettauswertung der Daten stattfindet (je mehr solche Punkte, desto geringer ist die Ersparnis). Allerdings beeinflusst auch die Anzahl der Instanzen in der Selektionsdatenbank leicht die Optimierungszeit. In Zeile 4 des Algorithmus 6.2 werden aus k_{sel} Stücken (5% der Daten) k'_{sel} Stücke so ausgewählt, dass das verwendete Selektionsmodell mindestens ein adjustiertes Bestimmtsheitsmaß von 0.98 aufweist (falls dieses erreichbar ist). Es hat sich für die beiden Datenbanken ergeben, dass durch diesen Schritt die Anzahl der selektierten Musikstücke von 5% auf durchschnittlich 3.4% gesenkt wird, ohne einen Verlust in der Modellgüte zu verzeichnen.

Die durchgeführte Analyse sollte einen Beitrag für die Wahl der Optimierungsstrategie in der nächsten Experimentenstufe leisten. Einerseits konnte kein signifikanter Unterschied zum globalen Signifikanzniveau zwischen FMBO und MBO nachgewiesen werden. Andererseits jedoch lagen die FMBO Ergebnisse im Median immer unter denen von MBO. Da für die Optimierung der multivariaten EZE ausreichend Zeit vorhanden war sowie das Interesse an der bestmöglichen Performance dieser neuen Strategie lag, wurde an dieser Stelle von der Autorin dieser Arbeit entschieden, multivariate EZE mit Hilfe von MBO mit naivem Kriging Sorrogatmodell zu optimieren.

8.3 Dritte Stufe der Experimente und Anwendungsfrage 4: univariate vs. multivarite EZE

Eine der wichtigsten Fragestellungen dieser Arbeit – Anwendungsfrage 4 (s. Abschnitt 7.1) – lautet, ob die hier vorgeschlagene Methode der multivariaten Einsatzzeiterkennung (Kapitel 4) bessere Erkennungsgüte liefert als das univariate Verfahren. Zur Erinnerung, bei der multivariaten Erkennung werden nicht nur ein, sondern alle oben definierten EZE-Merkmale (Abschnitt 3.5) berücksichtigt, wobei ein Klassifikationsmodell für die Vorhersage der Wahrscheinlichkeit eines Toneinsatzes in einem Fenster verwendet wird. Es wurden drei Klassifikationsmodelle in Betracht gezogen: das relativ einfache Modell der logistischen Regression (*logReg*), der Zufallswald (*randForest*) als ein Ensemble-Verfahren und ein allgemein erfolgreiches aber sehr zeitintensives SVM-Modell (*svm*).

Angesichts der langen Optimierungszeit der multivariaten EZE, konnte die obige Fragestellung nur auf der WAV-Datenbank und nur für die *online* und *offline* Fälle untersucht werden. Um unnötige Tests und Optimierungsläufe zu vermeiden, wird zuerst deskriptiv das beste der drei Klassifikationsmodelle im *online* Fall ausgewählt und gegen die univariate *online* EZE getestet. Die Optimierungsläufe für die *offline* EZE werden dann nur mit diesem ausgewählten Klassifikationsmodell gestartet. Wie im vorherigen Abschnitt begründet, wird hier MBO mit naivem Kriging Sorrogatmodell als Optimierungsstrategie verwendet.

Abbildung 8.12 vergleicht die Ergebnisse der multivariaten EZE mit der univariaten Variante. Am besten hat das Klassifikationsmodell *randForest* abgeschnitten, gefolgt von dem *logReg* Modell. Das *svm* Modell zeigt keine Verbesserung gegenüber der univariaten Erkennung. Die Performance des *randForest* Modells im *offline* Fall ist auch sehr zufriedenstellend: es ist eine deutliche Güteverbesserung zu sehen.

Nun wird mittels des Wilcoxon-Rangsummentests untersucht, ob die deskriptiv wahrgenommenen Verbesserungen auch statistisch signifikant sind. Die beiden Nullhypothesen mit jeweiligen p-Werten und Testentscheidungen sind unten aufgelistet:

- H_{AF4}^{online}: $MBO\_naivKM\_wav\_online^{(6)} = MBO\_multi.randForest\_naivKM\_wav\_online^{(1)}$,
  - p-Wert = $1.252 \cdot 10^{-4}$ **(ablehnen)**;
- $H_{AF4}^{offline}$: $MBO\_naivKM\_wav\_offline^{(6)} = MBO\_multi.randForest\_naivKM\_wav\_offline^{(1)}$,
  - p-Wert = $2.913 \cdot 10^{-6}$ **(ablehnen)**.

Die beiden p-Werte sind kleiner als die entsprechenden lokalen Signifikanzniveaus ($\alpha/4 = 0.0125$ für H_{AF4}^{online} und $\alpha/5 = 0.01$ für $H_{AF4}^{offline}$), so dass die Nullhypothesen zum globalen Testniveau von 5% (nach Bonferroni-Holm-Verfahren) abgelehnt werden. Somit weist die multivariate EZE mit Klassifikationsmodell *randForest* signifikant bessere F-Werte auf als die univariate EZE sowohl im *online* und als auch im *offline* Fall.

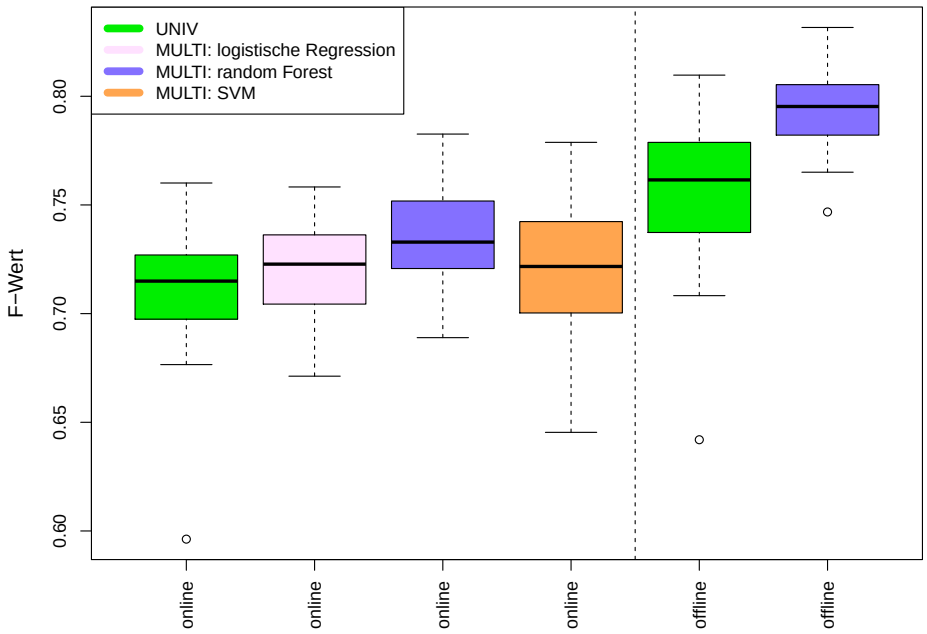

Abbildung 8.12: *Gegenüberstellung der Optimierungsergebnisse von univariater (UNIV) und multivariater (MULTI) Einsatzzeiterkennung mit verschiedenen Klassifikationsmodellen. Optimierungsverfahren: MBO mit naivem Kriging Surrogatmodell, Datenbank: WAV.*

Die verwendeten Klassifikationsmodelle zeigen ein sehr unterschiedliches Verhalten bezüglich der Modellanpassungszeit. In Abbildung 8.13 ist die Verteilung der Gesamtzeit für die Funktionsauswertungen in den sequentiellen Schritten von MBO veranschaulicht. Die extrem lange Modellanpassungszeit (und sehr große Streuung) von *svm* basierter multivariater EZE fällt dabei direkt auf. Die längste Iteration (d.h. ein MBO-Schritt) beim *svm* Modell dauerte mehr als 11 Tage. Die Anpassungszeiten von *randForest* und *logReg* zeigen eine kleinere Streuung, was die Planung der Dauer der Experimente wesentlich erleichtert.

Verglichen mit der *online* EZE benötigt die *randForest* basierte multivariate EZE im *offline* Fall deutlich mehr Zeit für die Funktionsauswertungen. Dies liegt offenbar daran, dass die Dimension der zusammengesetzten Lernmatrix D_{lern} (s. Algorithmus 4.1) unter Umständen deutlich größer wird (gegenüber von *online* Fall), was sich in der Rechenzeit für die Variablenselektion direkt widerspiegelt.

Es soll an dieser Stelle noch einmal verdeutlicht werden, dass die aufwändige Variablenselektion sowie die Modellanpassung nur in der Optimierungsphase notwendig ist. In der Validierungsphase sowie bei den eigentlichen Anwendungen wird nur das zu der besten Parametereinstellung zugehörige Modell für die Vorhersage verwendet, welches auch als Ergebnis der Optimierung zu verstehen ist. Solche Modellvorhersagen sind zeit-günstig und können in Echtzeit-Anwendungen problemlos eingesetzt werden.

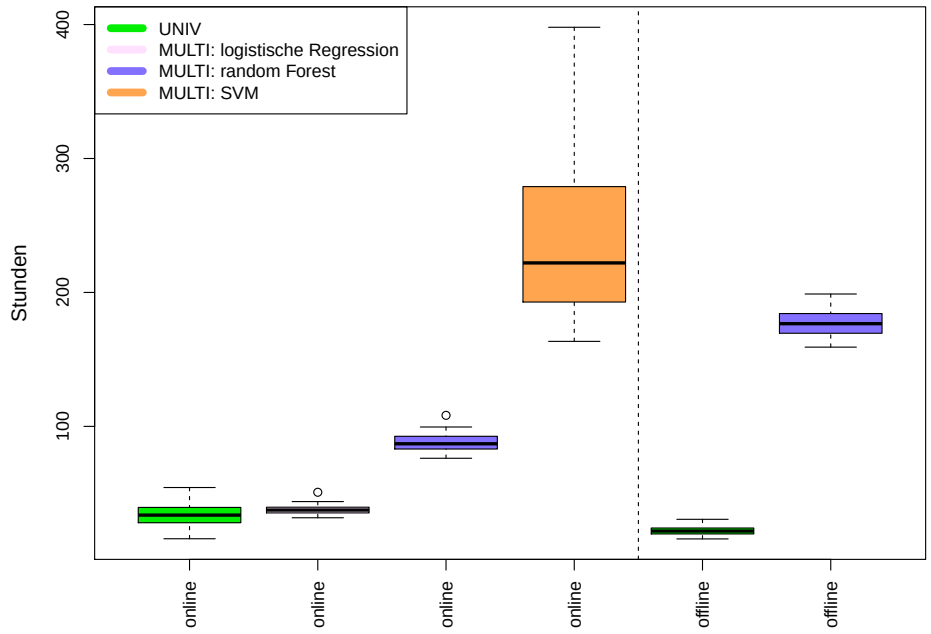

Abbildung 8.13: *Summierte Zeit in Stunden für Funktionsauswertungen pro ein Optimierungslauf für die sequentiellen Schritte (d.h. ohne Berücksichtigung des Startdesigns) bei univariater (UNIV) und multivariater (MULTI) Einsatzzeiterkennung mit verschiedenen Klassifikationsmodellen. Optimierungsverfahren: MBO mit naivem Kriging, Datenbank: WAV.*

Während die besten Einstellungen der zu optimierenden Parameter der multivariaten EZE in tabellarischer Form vorgestellt werden können (s. Tabelle 8.4 für die *randForest* basierte multivariate EZE), ist eine eindeutige Parametrisierung des *randForest* Klassifikationsmodells nicht möglich. Aus diesem Grund werden die Klassifikationsmodelle, die den fünf besten in Tabelle 8.4 aufgeführten Parametereinstellungen entsprechen (jeweils für die *online* und *offline* EZE), auf der beiliegenden CD im Ordner `randomForestModels` zur Verfügung gestellt. Analog zu dem Vorgehen bei der univariaten EZE werden die besten Parametereinstellungen in den jeweiligen Holdout-Wiederholung auf allen Musikstücken evaluiert, um bessere Vergleichbarkeit der EZE-Güte zu ermöglichen.

Neben der Tatsache, dass die multivariate EZE mit *randForest* Modell signifikant besser als die univariate ist, verkürzt sie auch die Latenzzeit bei den *online* Anwendungen. Aus Tabelle 8.4 geht nämlich hervor, dass alle top fünf Parametereinstellungen die Fensterlänge von 1024 Abtastwerten aufweisen. Somit kann die Latenzzeit von 46 auf 23 ms halbiert werden. Sowohl im *online* als auch im *offline* Fall überschreitet die optimale Sprungweite die halbe Fensterlänge, was eine Fensterüberlappung von ca. 30% (*online* EZE) bzw. 50% (*offline* EZE) bedeutet. Je kleiner die Überlappung, desto weniger Rechenzeit ist erforderlich.

Tabelle 8.4: Die fünf besten Parametereinstellungen für die *online* und *offline* multivariate EZE, die mittels MBO mit naivem Kriging und *randForest* Klassifikationsmodell auf der WAV-Datenbank ermittelt wurden.

EZE	*F*-Wert	*D*-Wert	N	h	*window.fun*	*spec.filter*	*spec.log*	ℓ	$t(l_O)$	$t(r_O)$	$t(min.dist)$	*classif.thresh*
online	0.787	0.009	1024	816	Hann.	Ja	Ja	19.25	0.027	0	0.025	0.310
	0.781	0.010	1024	686	Hann.	Ja	Ja	10.91	0.001	0	0.024	0.382
	0.781	0.010	1024	667	Hann.	Ja	Ja	15.16	0.032	0	0.029	0.365
	0.780	0.009	1024	744	Hann.	Ja	Ja	6.362	0.001	0	0.034	0.328
	0.779	0.012	1024	787	Hann.	Ja	Ja	5.900	0	0	0.035	0.342
offline	0.838	0.009	2048	1043	Black.	Ja	Ja	1.017	0	0.052	0.037	0.546
	0.836	0.008	2048	1054	Hann.	Ja	Ja	4.122	0	0.039	0.018	0.471
	0.833	0.008	2048	1024	Hann.	Ja	Ja	1.036	0.008	0.032	0.031	0.510
	0.828	0.008	2048	1026	Hann.	Ja	Ja	1.630	0.023	0.022	0.018	0.564
	0.823	0.008	2048	1036	Hann.	Ja	Ja	3.600	0.013	0.057	0.050	0.430

Wie im univariaten Fall ist hier die Anwendung der Hanning Fensterfunktion, der spektralen Filterung und der spektralen Logarithmierung empfehlenswert. Besonders interessant ist, dass bei der *online* Erkennung die optimale Klassifikationsgrenze (für Toneinsatz / kein Toneinsatz Entscheidung) mit ca. 0.35 gegeben ist, was deutlich unter der Standardeinstellung von 0.5 liegt. Bei der *offline* EZE, dagegen, entspricht diese Grenze ungefähr der Standardeinstellung.

Abschließend wird untersucht, welche EZE-Merkmale zur Modellbildung (beim *randForest* Modell) besonders häufig ausgewählt werden. Dabei handelt es sich um den Variablensselektionsschritt aus Zeile 4 des Algorithmus 4.1 (s. Kapitel 4). In Abbildung 8.14 ist dargestellt, wie viele Merkmale zur Modellbildung jeweils bei der *online* und *offline* EZE in 30 Wiederholungen ausgewählt werden. Bei der *offline* EZE stehen mehr Merkmale zur Verfügung, wodurch sich auch die tendenziell größere Anzahl an ausgewählten Variablen erklären lässt. Die Anzahl der Variablen wird nach dem Selektionsschritt insgesamt um ein Vielfaches reduziert, was zum einen die Modellierungszeit in der Lernphase enorm verkürzt und zum anderen die Zeit für die Merkmalsberechnung in echten Anwendungen günstig beeinflusst. In manchen Fällen handelt es sich bei mehreren ausgewählten Merkmalen im Grunde um das gleiche Merkmal: um seine Werte im aktuellen, vorherigen bzw. nächsten Fenster.

Tabelle 8.5 fasst die am häufigsten ausgewählten Merkmale zusammen. Hier sind die Namen der Merkmale aus Tabelle 3.2 verwendet, abgesehen von der zusätzlichen *onset* / *offset* Markierung. Dabei bedeuten die Positionen der Fenster wie '_a', '_links_1' oder '_rechts_2', dass es sich jeweils um das Merkmal in dem aktuellen, dem ersten von links bzw. dem zweiten von rechts Fenster handelt (siehe Parameter l_M und r_M in Abschnitt 4.2). Die genannte Tabelle besteht aus drei Teilen: in dem oberen Teil sind die am häufigsten ausgewählten Merkmale (d.h. diejenige, die in mindestens 10% der Wiederholungen ausgewählt wurden) bei der *online* EZE in geordneter Reihenfolge aufgelistet. In der rechten Spalte wird die Häufigkeit der Auswahl dieser Merkmale

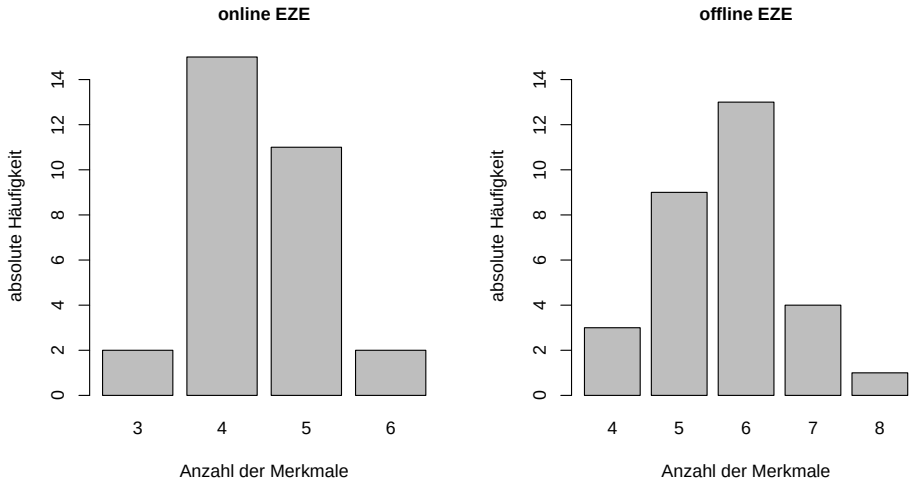

Abbildung 8.14: *Anzahl der ausgewählten Mermale in dem Variablenselektionsschritt (s. Zeile 4 des Algorithmus 4.1) bei der online und offline multivariaten EZE mit randForest Modell.*

bei der *offline* EZE notiert. In dem mittleren Teil der Tabelle sind häufig ausgewählte Merkmale der *offline* EZE aufgeführt, die nur aktuelle bzw. linke Fenster berücksichtigen, welche jedoch bei der *online* EZE nicht häufig oder gar nicht ausgewählt wurden. Anschließend werden in dem letzten Tabellenteil häufig ausgewählte Merkmale der *offline* EZE dargestellt, die die zukünftige Fenster berücksichtigen und bei der *online* EZE daher gar nicht vorkommen können.

In Tabelle 8.5 ist zu erkennen, dass im *online* Fall das ‘spektraler Fluss’ Merkmal des aktuellen Fensters sowie das ‘spektraler euklidischer Abstand’ Merkmal des ersten vorherigen Fensters besonders oft ausgewählt werden. Bei der *offline* EZE scheint bei allen Wiederholungen das ‘spektraler Fluss’ Merkmal des ersten zukünftigen Fensters besonders wichtig zu sein. Mit großem Abstand folgen die Merkmale ‘spektraler Fluss’ und ‘spektraler euklidischer Abstand’ der benachbarten Fenster. Zur Erinnerung, die beiden Merkmale unterscheiden sich lediglich in der Hinsicht, dass bei dem spektralen euklidischen Abstand die Tonendeinformation verwendet wird. Außerdem werden in ca. einem Drittel der Fälle bei den beiden EZE-Arten die ‘gewichtete spektrale Energie’ und ‘komplexe Domäne’ Merkmale (in verschiedenen Variationen) ausgewählt. Unterrepräsentiert bzw. gar nicht ausgewählt werden dagegen die amplitudenbasierten Merkmale sowie die auf der spektralen Verteilung basierenden Merkmale wie ‘spektraler Schwerpunkt’, ‘spektrale Streuung’ und ‘spektrale Schiefe’.

Tabelle 8.5: Die am häufigsten ausgewählten Merkmale in dem Variablenselektionsschritt (s. Zeile 4 des Algorithmus 4.1) bei der *online* und *offline* multivariaten EZE mit *randForest* Modell. Es wird prozentuell angegeben, wie oft das Merkmal in 30 Wiederholungen des jeweiligen Verfahrens ausgewählt wurde.

Merkmal	*Position des Fensters*	*Häufigkeit in %*	
		online EZE	*offline EZE*
Spec.Flux	aktuell	97	47
Spec.Euclid	links_1	67	30
Rect.Compl.Domain	aktuell	27	3
High.Freq.Cont.Diff	aktuell	27	7
Gauss.Freq.Cont.Diff	links_1	23	7
Spec.Flux	links_1	23	23
Compl.Domain	aktuell	20	3
Compl.Domain	links_3	17	10
Spec.Centroid.Abs.Diff	aktuell	13	0
Norm.Weight.Phase.Dev	aktuell	10	7
Rect.Compl.Domain	links_3	10	3
Ampl.Energy.Abs.Diff	aktuell	10	0
Ampl.Energy.Diff	links_1	10	0
Rect.Compl.Domain	links_1	0	17
Norm.Weight.Phase.Dev	links_1	0	10
Spec.Flux	links_2	0	10
Spec.Flux	rechts_1	0	100
Spec.Euclid	rechts_1	0	33
Spec.Flux	rechts_3	0	33
Spec.Flux	rechts_2	0	27
Spec.Euclid	rechts_2	0	23
High.Freq.Cont.Diff	rechts_2	0	23
Ampl.Max.Diff	rechts_2	0	23
High.Freq.Cont.Diff	rechts_3	0	20
Rect.Compl.Domain	rechts_2	0	10
Gauss.Freq.Cont.Diff	rechts_3	0	10

Kapitel 9

Ausblick

Ziel dieses Kapitels ist es, zum einen, alle bisher in der Arbeit erwähnten Vorschläge für die Verbesserung einzelner Probleme zusammenzufassen. Zum anderen werden hier auch neue Ideen vorgestellt und diskutiert. Das Kapitel ist in mehrere Absätze unterteilt, in denen die jeweiligen Aspekte behandelt werden.

MBO-Erweiterungen. Wie jedes Optimierungsverfahren hängt MBO (modellbasierte Optimierung, s. Abschnitt 6.2) von einer Reihe von Parametern ab, die die Güte des Verfahrens beeinflussen können. Dazu zählen unter anderem die Wahl des Surrogatmodells, des Zielkriteriums und der Größe des Startdesigns. Hier wurden nur konkrete Einstellungen (wie Kriging Modell oder EI-Zielkriterium) dieser Parameter berücksichtigt. Interessant wäre zu überprüfen, ob mit anderen Einstellungen die behandelten Anwendungsprobleme besser gelöst werden können. Insbesondere sollte die Problematik der kategoriellen Einflussgrößen größere Aufmerksamkeit bekommen. Denn die beiden hier vorgeschlagenen Lösungen – naives Kriging und dummy Kriging – sind trotz guter Ergebnisse nur sub-optimal. Es ist dabei nicht abzusehen, wie das naive Kriging Surrogatmodell abschneiden würde, wenn nicht alle Kategorien der betroffenen Einflussgrößen im Startdesign vertreten sein würden.

FMBO-Erweiterungen. Die instanzgebundene modellbasierte Optimierung (FMBO, s. Abschnitt 6.3) hängt von einem noch größeren Parameterset als MBO ab. Zusätzliche Parameter beziehen sich auf die Auswahl der Selektionsdatenbank und auf das Selektionsmodell. Insbesondere wäre die Anwendung von FMBO auf andere instanzgebunde Optimierungsprobleme interessant. Ein ähnliches Problem ist z.B. das Finden einer optimalen Konfiguration eines Klassifikationsverfahrens, welches auf einer großen Menge verschiedener Klassifikationsprobleme (also Datenbanken) im Mittel gut funktionieren soll. Aber auch eine Simulationsstudie, bei der die Ähnlichkeit der Performance eines zu optimierenden Algorithmus auf einer Menge von Probleminstanzen beliebig variiert werden kann, wäre an dieser Stelle sehr angebracht.

Güte der Einsatzzeiterkennung. Üblicherweise wird die Güte der Einsatzzeiterkennung (EZE) mit Hilfe des F-Werts gemessen (s. Abschnitt 3.9.1). In dieser Arbeit wurden einige Nachteile dieses Gütemaßes genannt und anhand eines Beispiels veranschaulicht. Essentiell ist dabei die fehlende Berücksichtigung der sogenannten „true negatives“ (richtige Entscheidung, dass kein

Toneinsatz zu einem Zeitpunkt vorliegt) und die Abhängigkeit von dem vorgegebenen Toleranzintervall. Es wurde anschließend der Versuch unternommen, den *D*-Wert als alternatives Gütemaß vorzuschlagen. Dieses Maß basiert auf der absoluten zeitlichen Abweichung zwischen den geschätzten und den wahren Einsatzzeiten. Der *D*-Wert hat allerdings theoretisch keine obere Grenze und ist nur auf dem gleichen Set von Musikstücken vergleichbar. Die Notwendigkeit eines sinnvollen alternativen Gütemaßes bleibt nach wie vor bestehen.

***Pseudo-online* EZE.** Wie in Abschnitt 8.1.3 gezeigt wurde, weist die *pseudo-online* EZE signifikant bessere F-Werte auf als die *online* EZE. Der einzige Unterschied zwischen den beiden Verfahren ist die Tatsache, dass bei der *pseudo-online* EZE die Signalamplitude im Voraus durch das absolute Maximum geteilt wird. Somit wird sie auf das Intervall $[-1,1]$ skaliert. Ein weiterer Vorteil der *pseudo-online* EZE ist, dass es mit halb so großer Latenzzeit funktionieren kann wie die besten Einstellungen des *online* EZE Algorithmus (23 ms anstatt 46 ms). Aus diesem Grund sollte bei *online* Anwendungen eine Schätzung der maximalen Signalamplitude (z.B. nach wenigen Sekunden und dann regelmäßig im Laufe der Aufnahme) erfolgen, um von den Vorteilen der *pseudo-online* EZE zu profitieren.

Sensibilität der optimalen EZE-Einstellungen. In dieser Arbeit wurden optimale Parametersets für die jeweiligen Anwendungsprobleme genannt. Allerdings ist interessant, wie stark sich die Güte der entsprechenden Algorithmen variiert, wenn man diese optimalen Einstellungen leicht verändert. Dadurch könnte z.B. festgestellt werden, welche Parameter keinen bzw. nur sehr geringen und welche einen entscheidenden Einfluss auf die Zielgröße haben. Hierfür sollten allerdings sinnvolle Kriterien für die Messung der Verbesserung bzw. Verschlechterung der Performance definiert werden. Besonders interessant wäre die Zulassung von beliebigen Fensterlängen – im Gegensatz zu den üblichen Potenzen von 2. Dies ist zwar technisch möglich, nimmt aber mehr Rechenzeit in Anspruch (weil schnelle Fourier Transformation, s. Abschnitt 2.4, nicht mehr angewendet werden kann). Außerdem ist die Akzeptanz solcher beliebigen Fensterlängen unter den Anwendern und Entwicklern der EZE-Algorithmen fraglich, denn die Potenzen von 2 als Fensterlängen haben sich inzwischen sehr stark etabliert.

Nutzen der zeitlichen Struktur der Musikstücke. Bisher wurde in „State of the Art“ Literatur zu EZE ignoriert, dass bei einer Musikaufnahme ein Toneinsatz theoretisch nicht zu einem beliebigen Zeitpunkt anfangen kann. Denn jedes Musikstück hat eine eigene zeitliche Struktur (Takt), welche sich im Laufe des Stückes verändern kann. Die Noten – Grundbausteine der Musik – unterteilen sich bezüglich ihrer Länge in sieben grundlegende Klassen[1], wie in Abbildung 9.1 veranschaulicht wird[2]. Allerdings hängt es von dem Tempo des Stückes ab, wie lange z.B. die ganze Note gespielt wird. Es stellt sich somit die Frage, wie diese Information verwendet werden kann, um die EZE zu verbessern.

[1] Punktierungsregeln werden später angesprochen.

[2] Quelle: `https://de.wikipedia.org/wiki/Notenwert`, Stand: 01.05.2015.

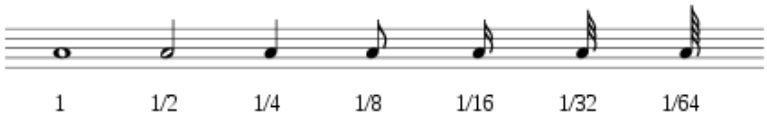

Abbildung 9.1: *Notenlägen-System: schrittweise Halbierung der Notenlängen.*

Zuerst wird aber exemplarisch untersucht, ob die Notenlängen der Musikaufnahmen – echter (WAV-Datenbank) und synthetischer (MIDI-Datenbank) – eine Struktur aufweisen. Dafür werden je Datenbank zwei Beispiele veranschaulicht. Abbildung 9.2 stellt Zeitreihen der ersten Differenzen jeweils der wahren und der geschätzten Toneinsätze vier ausgewählter Musikstücke dar. Die geschätzten Einsatzzeiten wurden mit Hilfe der Empfehlungen von Böck et al. (2012) für *offline* EZE berechnet (s. Abschnitt 8.1). Interessant ist, dass je mehr Struktur bei den wahren Toneinsätzen vorhanden ist, desto mehr Struktur liegt auch bei den geschätzten Toneinsätzen vor. Auch der F-Wert der EZE ist bei solchen strukturierten Aufnahmen meist relativ groß.

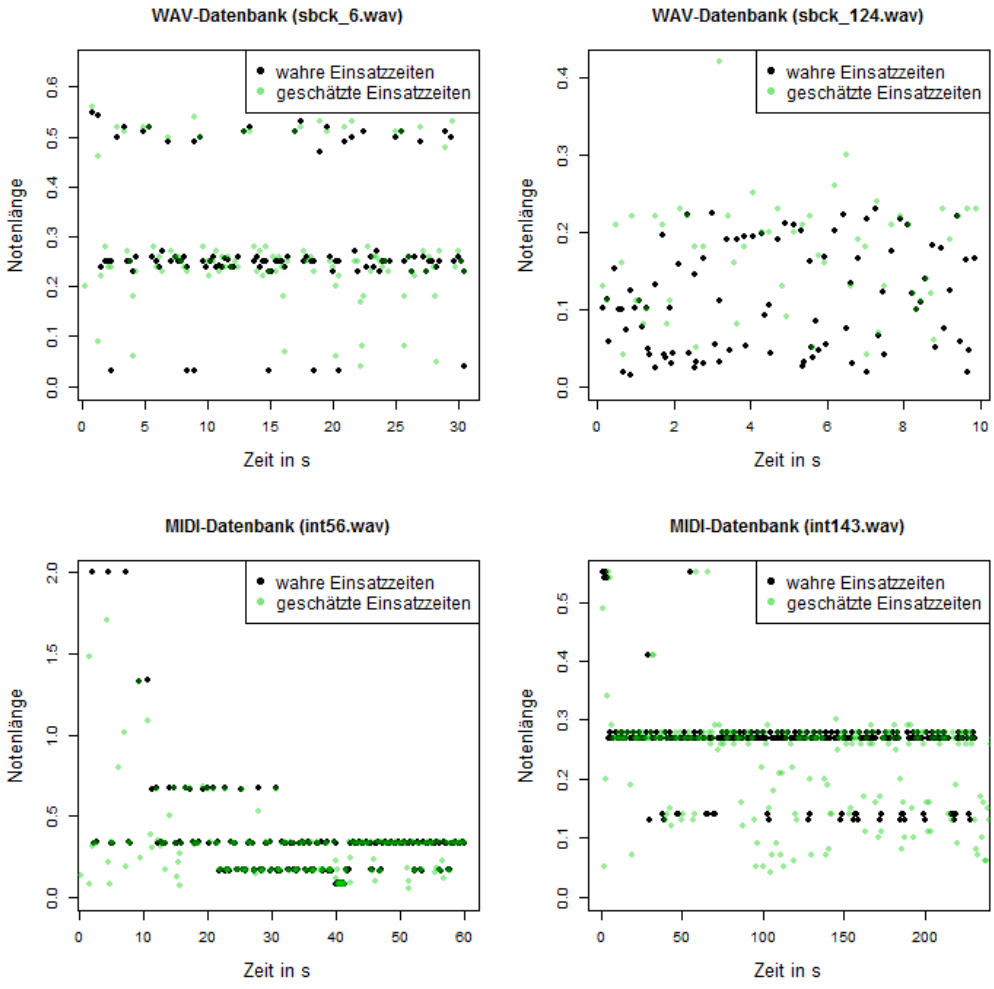

Abbildung 9.2: *Verteilung der Variable Notenlänge der wahren (schwarz) und den geschätzten (grün) Einsatzzeiten für jeweils zwei Musikstücke aus der WAV-Datenbank (oben) und der MIDI-Datenbank (unten). In Klammern ist der Name der entsprechenden Audiodatei angegeben.*

Da es sich bei der WAV-Datenbank um die echten Aufnahmen mit manuell notierten Toneinsätzen handelt, ist zu erwarten, dass die im Notenblatt vorgesehenen Toneinsatzzeiten nicht exakt wiedergegeben werden (Fehlerterme verursacht durch menschliche Faktoren). Größtenteils kann dennoch eine Struktur erkannt werden, wie in der oberen linken Darstellung in Abbildung 9.2 zu sehen ist. Hier sind drei Cluster bezüglich der Tonlänge der wahren Einsätze zu sehen: in der Nähe von 0.03 s, von 0.25 s und von 0.5 s. Bei der ersten Klasse handelt es sich vermutlich um Toneinsätze, die gleichzeitig stattfinden sollten. Zur Erinnerung, folgend dem Vorschlag von Böck et al. (2012) wurden alle Einsätze, die in weniger als 0.03 s stattfinden, zu einem Toneinsatz aggregiert. Es könnte somit sinnvoll sein, diese Grenze auf 0.05 s zu erhöhen. Die zweite Klasse würde die kleinste Längeneinheit repräsentieren und die dritte genau das doppelte der zweiten Klasse. Somit könnte das Notenlängen-System aus Abbildung 9.1 eingehalten werden.

Wie aus dem oberen rechten Plot in Abbildung 9.2 ersichtlich ist, kann bei manchen Musikstücken kein Notenlängen-System beobachtet werden. In diesem Fall handelt es sich um ein Latino-Amerikanisches Stück, gespielt mit mehreren Instrumenten. Vermutlich werden bei solchen Aufnahmen, die zum Teil auf spontanen Improvisationen basieren, die klassischen Regeln der abendländischen Musik nicht immer eingehalten. Nichtsdestotrotz, ein Algorithmus zur EZE sollte auch in diesem Fall die Toneinsätze möglichst korrekt schätzen.

Die beiden unteren Plots in Abbildung 9.2 entsprechen zwei MIDI-Stücken und repräsentieren typische Notenlängen-Muster in dieser Datenbank. Die Stücke werden dabei so programmiert, dass sie möglichst exakt nach dem vorgegebenen Notenblatt abgespielt werden. Es ist jedoch zu sehen, dass die theoretische Struktur der Notenlängen-Verdoppelung nur mit Ausnahmen eingehalten wird. So scheinen die Cluster der schwarzen Punkte auf dem unteren rechten Bild äquidistant platziert zu sein. Dabei bedeutet dies nicht automatisch die Verletzung des in Abbildung 9.1 dargestellten Systems. Denn es existiert auch Punktierungsregeln für die Verlängerung der Notenlängen. So ist z.B. eine Viertelnote mit einem Punkt die Dreiachtelnote. Diese Tatsache wird aber vorerst nicht berücksichtigt.

Um die Notenlängen-Information für die Verbesserung der EZE zu verwenden, sollten zwei wesentliche Aspekte berücksichtigt werden. Zum einen soll das Notenlängen-Schema eines Musikstücks ermittelt werden und zwar nach möglichst wenigen Toneinsätzen (falls von *online* Anwendungen ausgegangen wird). Darunter ist zu verstehen, dass z.B. die kleinste Notenlänge bestimmt wird und – entsprechend der Verdoppelungsstruktur – andere theoretisch möglichen Notenlängen festgelegt werden. Zum anderen stellt sich die Frage, wie diese Information weiter verarbeitet werden kann. Während hier für das erste Problem eine einfache Idee vorgeschlagen und evaluiert wird, werden für das zweite Problem lediglich einige Ansätze diskutiert.

Bezüglich der ersten Herausforderung wird folgendes Vorgehen vorgeschlagen: als erstes werden die ersten q Toneinsatzzeiten und die zugehörigen Tonlängen bestimmt. Im zweiten Schritt wird diejenige Tonlänge bestimmt, die sich am besten als Referenz bzw. Grundlage für die anschließende Schätzung des Tonlängen-Schemas eignet.

Dafür wird zuerst für die i-te (Referenz-)Tonlänge l_i^* das i-te Schema $S^i \in R^{13}$ bestimmt:

$$S_k^i = 2^{k-7} \cdot l_i^*,\ i = 1,\ldots,q. \tag{9.1}$$

Anschließend werden absolute Abweichung der restlichen Tonlängen l_j zu den am nähesten liegenden Zeitlinien dieses Schemas berechnet. Hier ist anzumerken, dass bei den längeren Tönen größere Abweichungen toleriert werden können als bei den kürzeren. Da es sich hier ein Verdoppelungsschema handelt, werden sowohl die restlichen Tonlängen als auch die Zeitlinien des zugehörigen Schemas logarithmiert (mit Basis 2), bevor die absoluten Abweichungen berechnet werden. Dauert ein Ton genau die doppelte Zeit als z.B. vorgesehen, wird die Differenz der logarithmierten Tonlängen genau 1 betragen. Die Abweichungen werden dann aufsummiert:

$$d_i = \sum_{j=1}^{q} \min_{1 \le k \le 13} |\log_2(l_j / S_k^i)|. \tag{9.2}$$

Das Schema mit der kleinsten summierten Abweichung d_i wird als Notenlängen-Schema der Musikaufnahme betrachtet.

Da es sich bei den Referenztonlängen um jede beliebige Notenlänge handeln kann, wird das Schema so konstruiert, dass auch die Extremfälle – kleinste bzw. größte Notenlänge – einbezogen werden. Dadurch enthält das Schema unumgänglich solche Tonlängen, die theoretisch nicht möglich sind. Dies ist jedoch weniger problematisch, als umgekehrt.

Um die Güte dieses Vorschlages zu überprüfen, wird für jedes Stück der WAV- und MIDI-Datenbank das Notenlängen-Schema anhand der ersten $q = 10$ wahren Toneinsatzzeiten (falls zehn vorhanden) geschätzt. Es werden die wahren Einsatzzeiten verwendet, um erstmal das „Best Case“-Szenario zu überprüfen. Anschließend werden Abweichungen der restlichen Töne der Musikaufnahme von dem geschätzten Schema berechnet (analog wie oben beschrieben) und der Anteil der Töne bestimmt, die von diesem Schema abweichen. Eine logarithmierte Tonlänge weicht dabei von dem Schema ab, falls ihre absolute Differenz zu der am nähesten liegenden logarithmierten Zeitlinie des Schemas mehr als $\log_2(1.1)$ beträgt[3].

Abbildung 9.3 stellt die Verteilung der Anteile der von dem geschätzten Schema abweichenden Tonlängen dar, sowohl für die WAV- als auch für die MIDI-Datenbank. Dabei ist ein großer Unterschied zwischen WAV- und MIDI-Files zu sehen. Bei den echten Aufnahmen scheint das anhand von 10 ersten Tönen geschätzte Notenlängen-Schema nicht für das restliche Stück geeignet zu sein, denn bei der Mehrheit der Stücke weicht mehr als die Hälfe der restlichen Töne von diesem Schema ab. Vermutlich liegt dies an der Definition einer Tonlängenabweichung (max. 10% von der vorgesehenen Länge), die angesichts der menschlichen Fehler sowohl bei der Musikkomposition als auch bei der nachträglichen manuellen Notation der Toneinsätze zu strikt zu sein

[3] Sollte ein Ton z.B. eine Sekunde dauern, so wäre hiermit eine absolute Abweichung von 100 ms tolerabel. Allgemein wird dadurch eine Abweichung von maximal 10% von der vorgesehen Tonlänge toleriert.

scheint. Bei den meisten Stücke aus der MIDI-Datenbank, dagegen, erklärt das geschätzte Schema fast perfekt die Notenlängenverteilung der restlichen Töne.

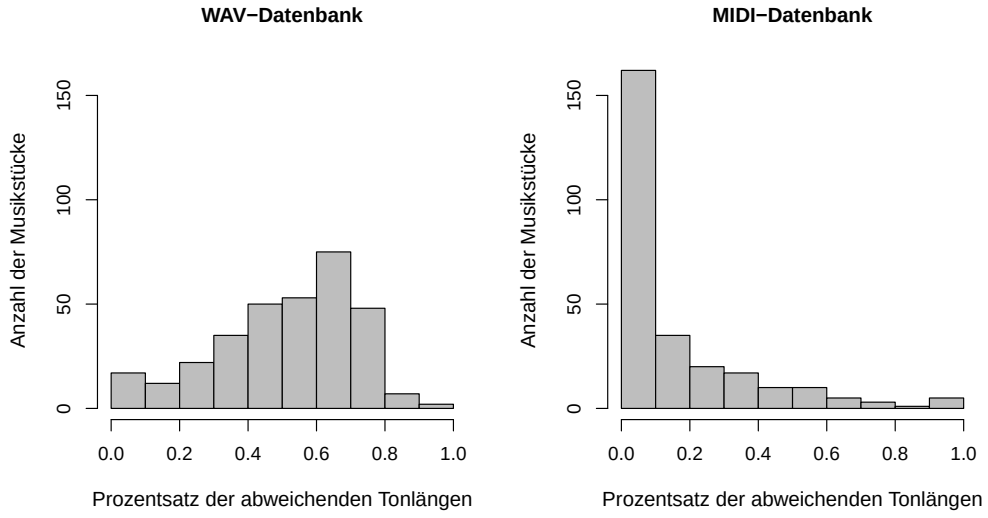

Abbildung 9.3: *Prozentsatz der von dem geschätzten Schema abweichenden Tonlängen je Musikstück. Links ist die Verteilung des Prozentsatzes für die Musikstücke der WAV-Datenbank und rechts der MIDI-Datenbank veranschaulicht.*

Angenommen, das Notenlängen-Schema liegt vor oder wurde geschätzt. Die wichtigste Frage ist nun, wie diese zusätzliche Information bei EZE eingebunden werden kann. Denkbar wäre nach jedem geschätzten Toneinsatz eine a-priori Verteilung über die Zeit zu legen, in der ein neuer Toneinsatz erwartet wird. Fraglich ist allerdings, ob jeder Zeitlinie des Schemas gleiche Wahrscheinlichkeit zugeordnet werden soll oder ob – angesichts der aktuell betrachteten Notenlängen-Verteilung – die besonders häufigen Notenlängen stärker gewichtet werden sollten. Außerdem ist unklar, wie eine solche multimodale Wahrscheinlichkeitsdichte zu konstruieren ist. Nachdem anschließend ein Zeitpunkt für den neuen Toneinsatz anhand des EZE-Algorithmus vorgeschlagen wird, soll die Kompatibilität dieses Vorschlages mit der gelegten a-priori Verteilung überprüft werden. Findet der Einsatz an einer sehr wahrscheinlichen Stelle statt, wird die Schätzung übernommen. Ist dies allerdings nicht der Fall, wären drei Varianten denkbar: den geschätzten Toneinsatz trotzdem übernehmen und das Notenlängen-Schema neu berechnen, den Toneinsatz ignorieren oder den Toneinsatz auf die nächste „wahrscheinliche Stelle“ verschieben.

Zusammenfassend lässt sich sagen, dass die Betrachtung der „nicht zufälligen“ zeitlichen Struktur der Musikstücke ein großes Potenzial für die Verbesserung der EZE darstellt. Es bedarf aber einem relevanter Forschungs- und Optimierungsaufwand, um diese Idee zu verwirklichen[4].

[4]Die hier vorgestellte Idee basiert auf zahlreichen Diskussionen dieses Themas mit Prof. Dr. C. Weihs, der sie hauptsächlich motiviert hat. Weiterhin hat Dr. U. Ligges einen wesentlichen Beitrag zu der durchgeführten Analyse geleistet.

Kapitel 10

Zusammenfassung

Diese Arbeit befasst sich mit der Optimierung der Toneinsatzzeiterkennung. Die Einsatzzeiterkennung ist eine wichtige Komponente für verschiedene Anwendungen wie Musiktranskription oder Wiedergabe der Musik in Hörgeräten. Angesichts der Komplexität der bekannten Algorithmen zur Einsatzzeiterkennung stellt sich ihre Optimierung als ein nicht triviales Problem dar. Als Hauptergebnisse dieser Arbeit lassen sich die Entwicklung einer neuen schnellen Optimierungsstrategie sowie eines innovativen multivariaten Ansatzes für die Einsatzzeiterkennung nennen.

Die Grundidee der verwendeten sequentiellen modellbasierten Optimierungsmethode besteht in der Modellierung des Zusammenhanges zwischen den Einflussparametern und der Zielgröße durch ein so genanntes Surrogatmodell. Die Versuche (bzw. Experimente) werden im Gegensatz zur klassischen Versuchsplanung iterativ durchgeführt, wobei ein neuer Punkt sowohl einen möglichst guten Wert der Surrogatmodell-Vorhersage haben als auch möglichst weit von den bereits evaluierten Punkten entfernt liegen soll (s. Abschnitt 6.2).

Das weit verbreitete Kriging Surrogatmodell ist nur für stetige Einflussparameter konzipiert worden, wobei das betrachtete Anwendungsproblem mehrere kategorielle Parameter aufweist. In Abschnitt 6.2.2 wurden zwei einfache Lösungen vorgeschlagen: naives Kriging, bei dem die kategoriellen Parameter als stetig behandelt werden, und dummy Kriging, das auf einer internen Bildung der Dummy-Variablen für diese Parameter basiert. Dummy Kriging ist statistisch korrekter als naives Kriging, benötigt aber mehr Zeit für die Modellanpassung. Es hat sich ergeben, dass kein signifikanter Unterschied zwischen den beiden Surrogatmodellen bezüglich der Optimierungsgüte nachgewiesen werden konnte. Aus diesem Grund wurde naives Kriging für weitere Experimente bevorzugt. Dieses Ergebnis darf allerdings nicht auf andere Probleme übertragen werden, da die Performance von naivem Kriging weiter erforscht werden sollte.

Des Weiteren wurde in Abschnitt 6.3 die instanzgebundene modellbasierte Optimierung als schnelles Optimierungsverfahren vorgeschlagen. Dieses eignet sich für solche Probleme, bei denen die Zielfunktion in jeder Iteration auf mehreren Instanzen ausgewertet werden soll. Hier stellen die Musikstücke die Probleminstanzen dar. Die Hauptidee liegt dabei in einer modellbasierten Entscheidung, ob die Instanzauswertung in jeder Verfahrensiteration wegen einer nicht aussichtsreichen Güte frühzeitig abgebrochen werden soll. Im Mittel konnte dadurch ca. 85% der

Funktionsauswertungszeit (in den sequentiellen Schritten) gespart werden. Zwar kam das schnelle Verfahren nicht so nah an das Optimum wie die gewöhnliche Optimierung heran, die verwendete Testmethode konnte jedoch keinen signifikanten Unterschied zwischen den beiden Strategien zum globalen Signifikanzniveau von 5% nachweisen.

Was das Anwendungsproblem angeht, wurden verschiedene interessante Fragestellungen diskutiert. Als erstes wurde das etablierte Gütemaß der Einsatzzeiterkennung kritisch betrachtet und seine Nachteile erwähnt. Ein alternatives Gütemaß wurde vorgeschlagen, welches jedoch noch weiter entwickelt werden sollte, um als eigenständiges Gütekriterium zu gelten (s. Abschnitt 3.9). Weiterhin wurden, wie in Abschnitt 5.2 beschrieben, zwei Musikdatenbanken für die Optimierung verwendet: die manuell annotierte WAV-Datenbank (bestehend aus echten Aufnahmen) und die synthetisch erzeugte MIDI-Datenbank. Dabei konnte festgestellt werden, dass auf der MIDI-Datenbank signifikant bessere Güte erreicht werden kann. Da die synthetisch generierten Stücke nicht die Komplexität einer echten Aufnahme wiedergeben können, sind die Toneinsätze anscheinend leichter zu identifizieren.

Außerdem wurde im Rahmen der Forschung ein Konvertierungsprogramm – *RealConverter* – entwickelt (s. Abschnitt 5.1.2), welches ein Notenschema mit Hilfe einer echten Musiktondatenbank in ein Musikstück umwandelt. Die durch *RealConverter* erzeugten Stücke weisen eine bessere Klangqualität im Vergleich zu herkömmlichen Konvertierungsprogrammen auf. Es konnte in einem direkten Vergleich beobachtet werden, dass auf solchen Stücken leicht bessere Zielfunktionswerte erreicht wurden. Ferner wurde das Verbesserungspotential der *online* Einsatzzeiterkennung (d.h. ohne Hinzunahme der zukünftigen Signalinformation) durch die Umskalierung der Signalamplitude identifiziert. Unter Umskalierung ist die Teilung der Amplitude durch den maximalen Amplitudenbetrag zu verstehen. Dabei könnte die maximale Signalamplitude beispielsweise nach wenigen Sekunden der Musikaufnahme berechnet oder geschätzt werden.

Das wesentliche Ergebnis in Bezug auf das Anwendungsproblem ist die Entwicklung der multivariaten Einsatzzeiterkennung (s. Kapitel 4): eine Methode, bei der nicht nur ein, sondern mehrere Merkmale bei der Schätzung der Toneinsatzzeiten berücksichtigt werden. Realisiert wird sie durch die Anwendung von Klassifikationsverfahren. Besonders erfolgreich scheint der Zufallswald für diese Aufgabe zu sein: die Erkennungsgüte konnte deutlich und auch statistisch signifikant (zum globalen Niveau) im Vergleich zu herkömmlichen univariaten Algorithmen verbessert werden. Dabei beträgt die mittlere Verbesserung des F-Wertes 0.026 (entspricht 3.79%) für *online* und 0.037 (entspricht 5.12%) für *offline* Erkennung. Außerdem ermöglichen die optimalen Einstellungen des multivariaten Verfahrens eine deutliche Reduktion der Latenzzeit der Erkennung.

Anschließend wurde in Kapitel 9 eine innovative Idee für eine weitere Verbesserungsmöglichkeit der Einsatzzeiterkennung diskutiert: Berücksichtigung der (größtenteils) strikten zeitlichen Struktur einer Musikaufnahme. Nachdem diese Struktur erkannt wird, könnten beispielsweise die geschätzten Toneinsätze präziser gesetzt bzw. verworfen werden.

Anhang A

Abbildungen

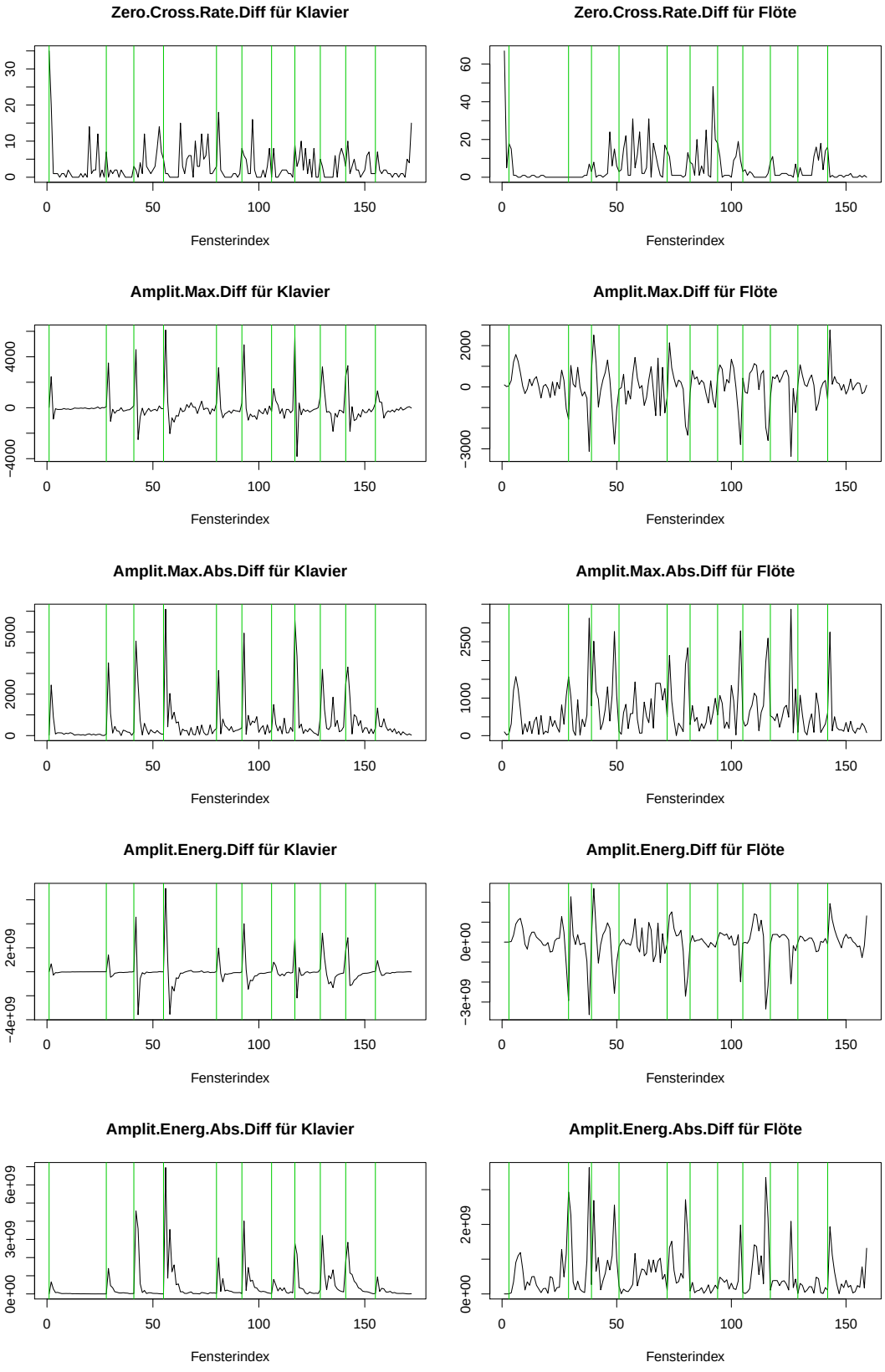

Abbildung A.1: *Gruppe A: auf der Amplitudenenergie basierende Merkmale. Die vertikalen Linien markieren die wahren Toneinsatzzeiten.*

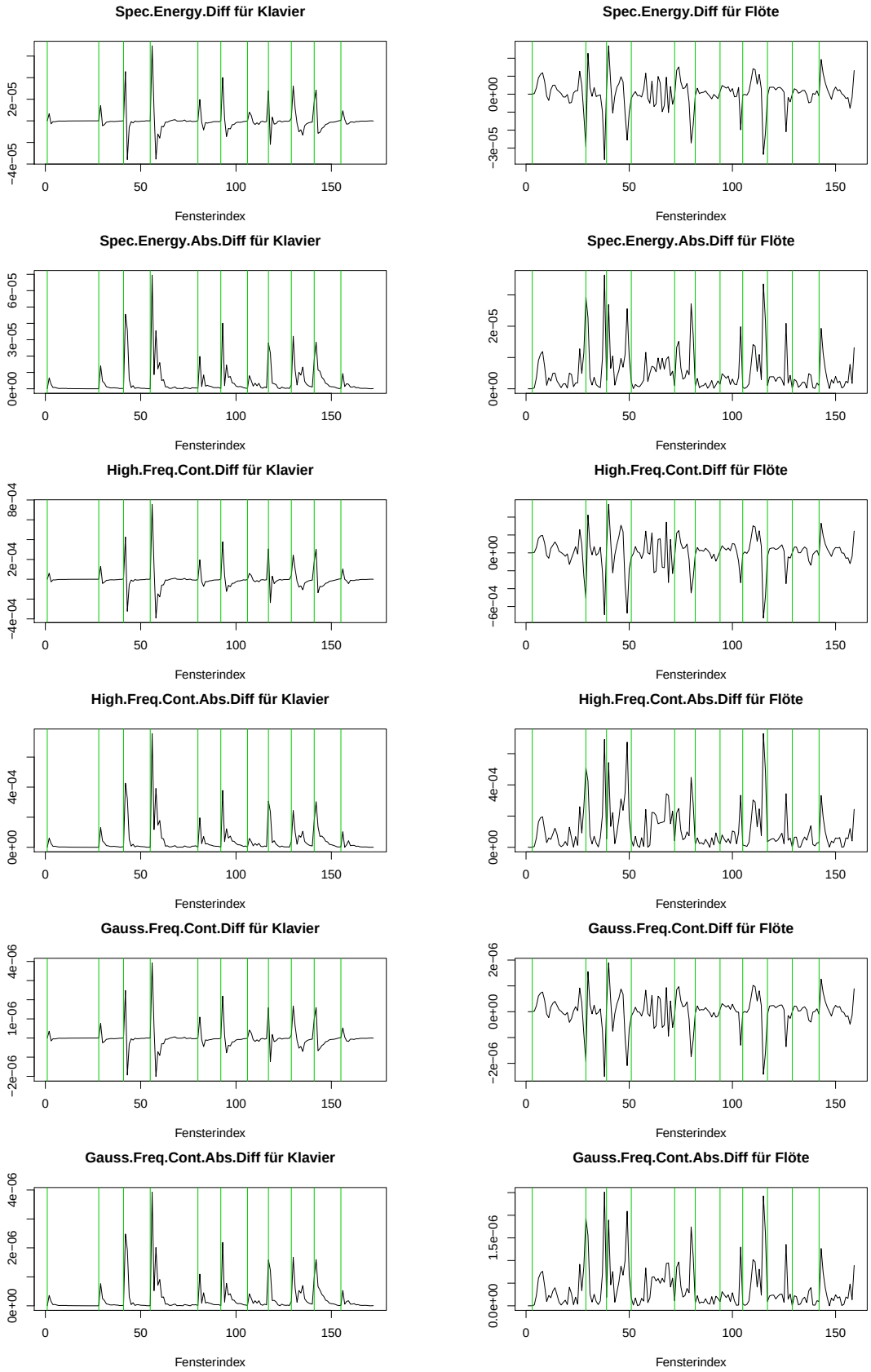

Abbildung A.2: *Gruppe B: auf der Spektralenergie basierende Merkmale. Die vertikalen Linien markieren die wahren Toneinsatzzeiten.*

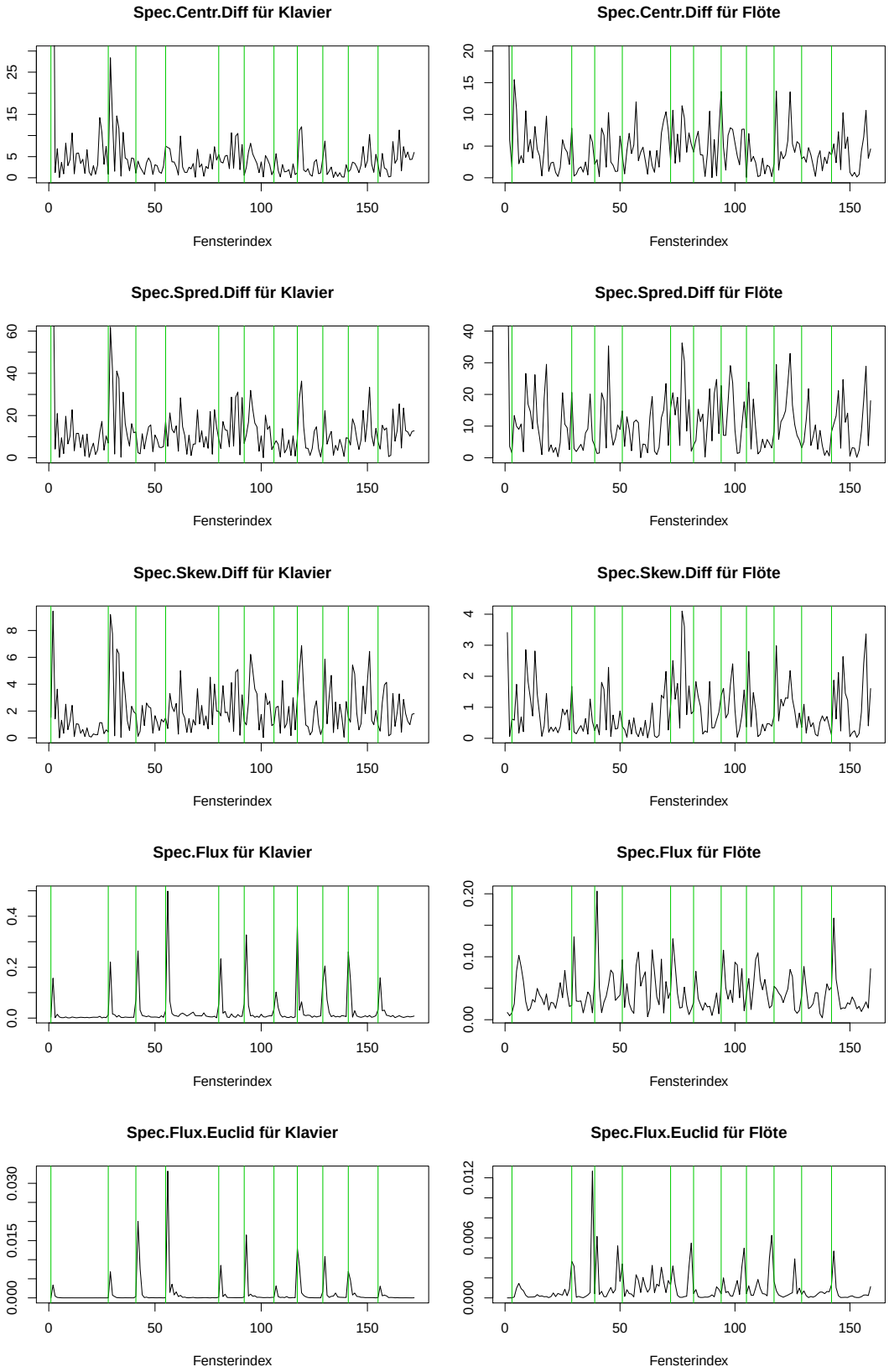

Abbildung A.3: *Gruppe C: auf dem Absolutwert der Fourier Koeffizienten basierende Merkmale. Die vertikalen Linien markieren die wahren Toneinsatzzeiten.*

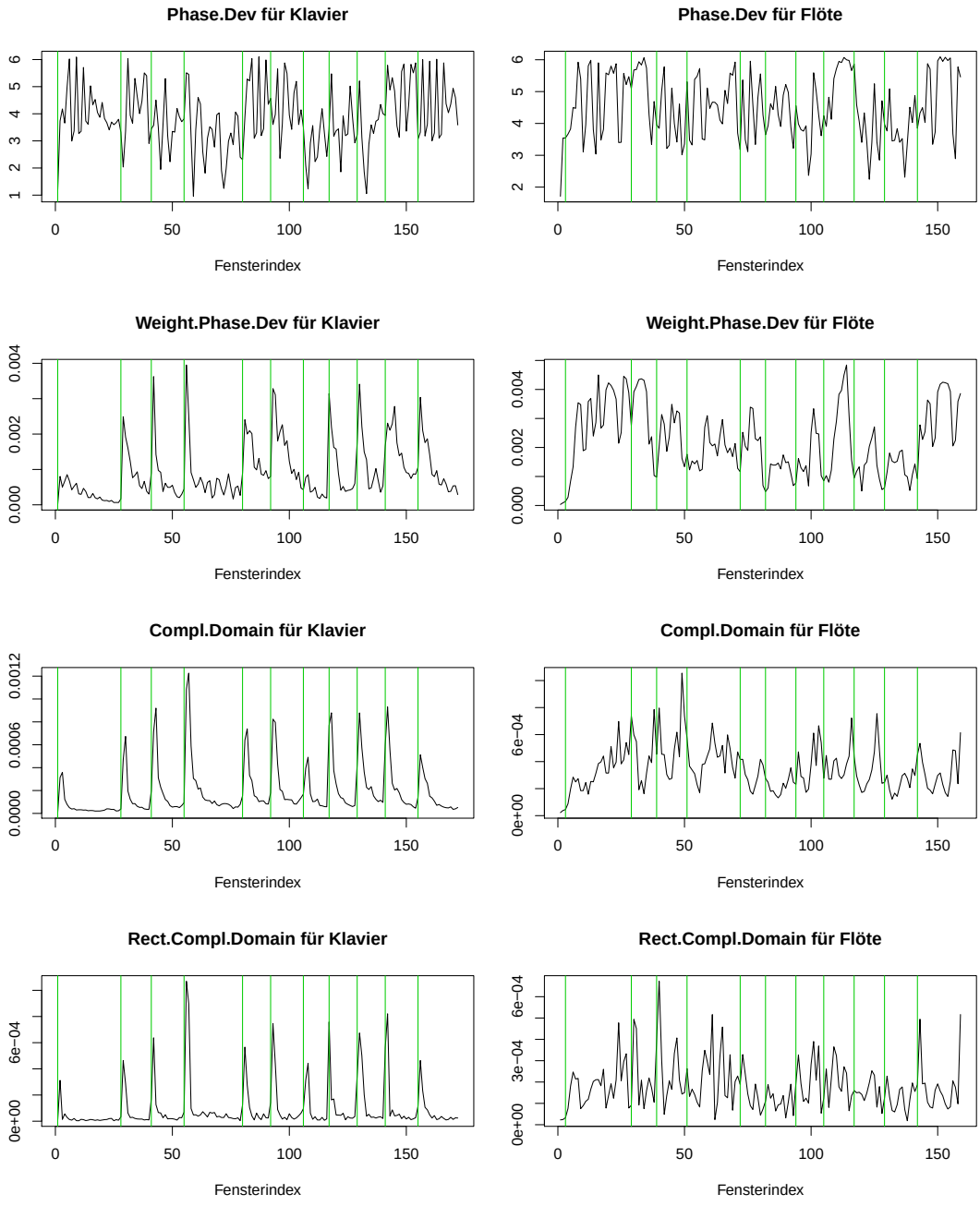

Abbildung A.4: *Gruppe D: auf dem Absolutwert und der Phase von Fourier Koeffizienten basierende Merkmale. Die vertikalen Linien markieren die wahren Toneinsatzzeiten.*

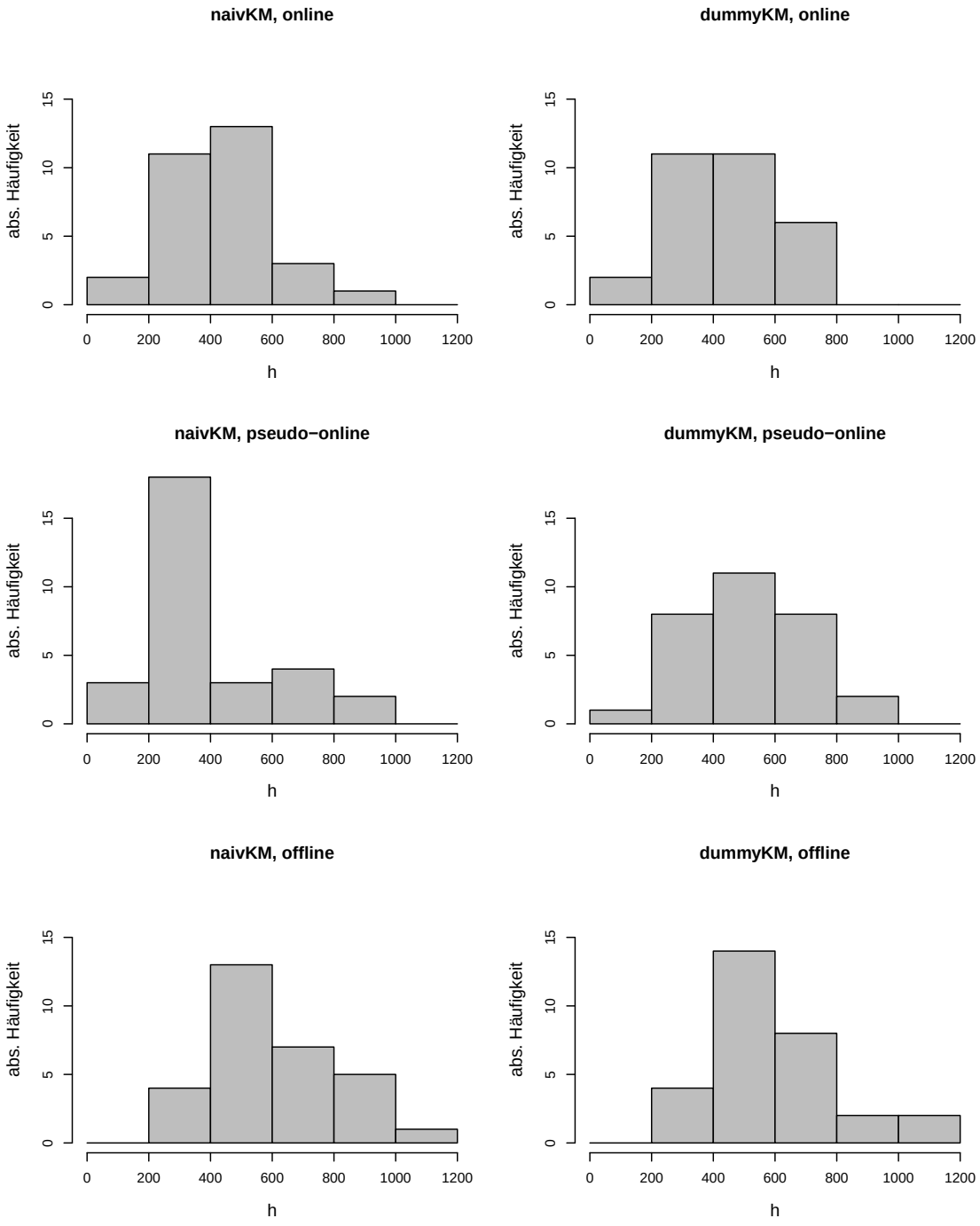

Abbildung A.5: *Verteilung der optimalen Einstellungen für den Parameter h (Sprungweite) in Samples. Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

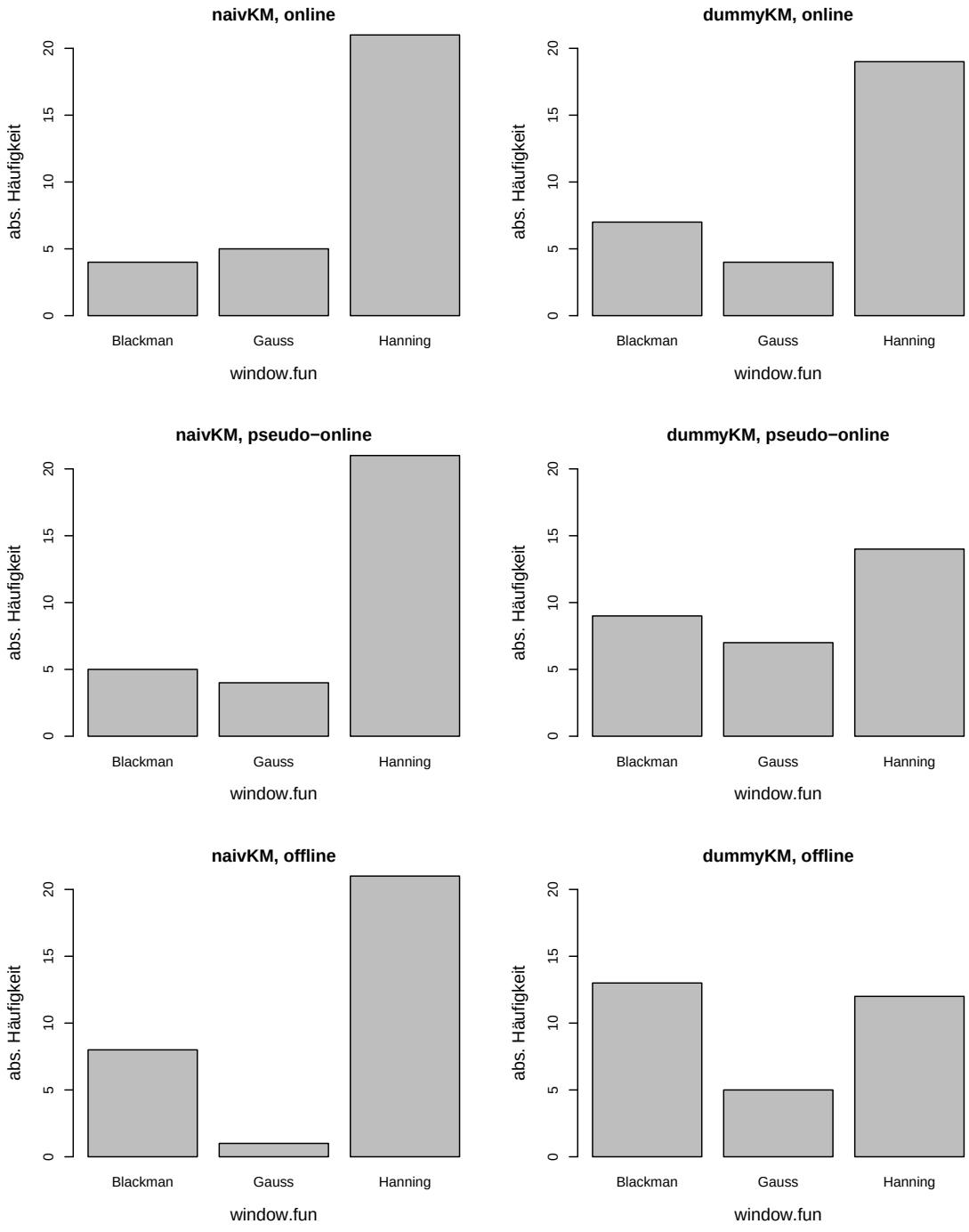

Abbildung A.6: *Verteilung der optimalen Einstellungen für den Parameter window.fun (Fensterfunktion). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

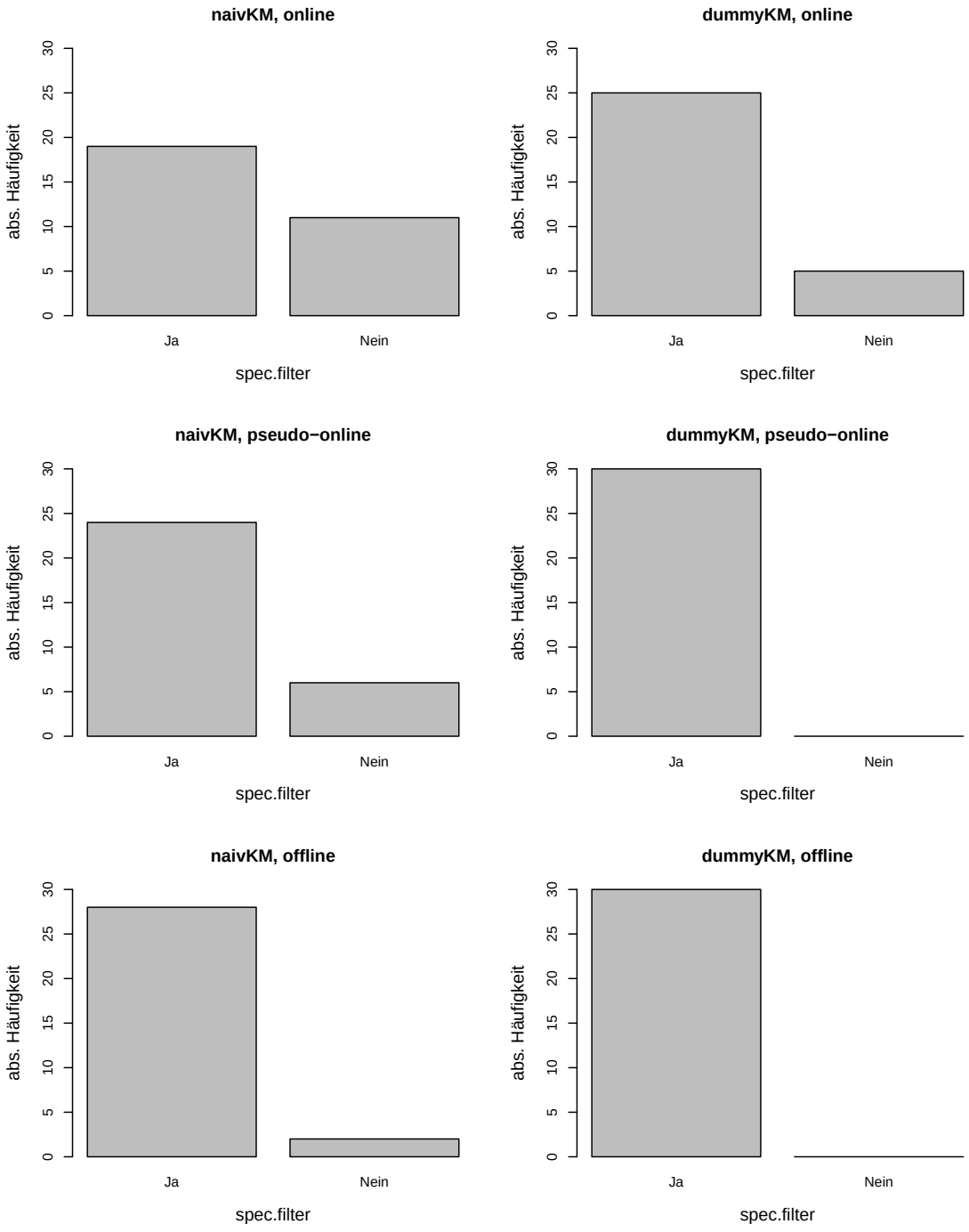

Abbildung A.7: *Verteilung der optimalen Einstellungen für den Parameter spec.filter (spektrale Filterung). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

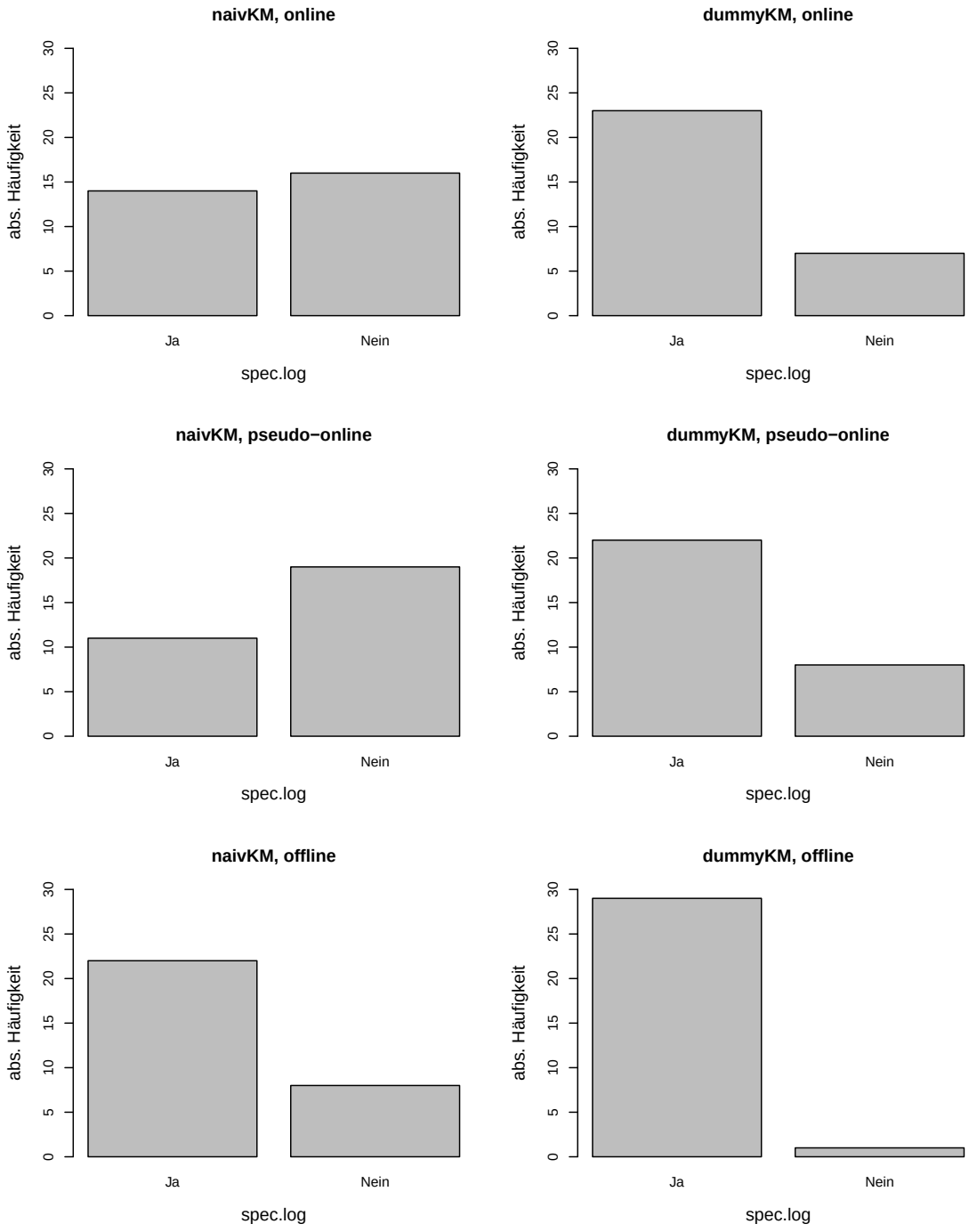

Abbildung A.8: *Verteilung der optimalen Einstellungen für den Parameter spec.log (Logarithmierung des Spektrums). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

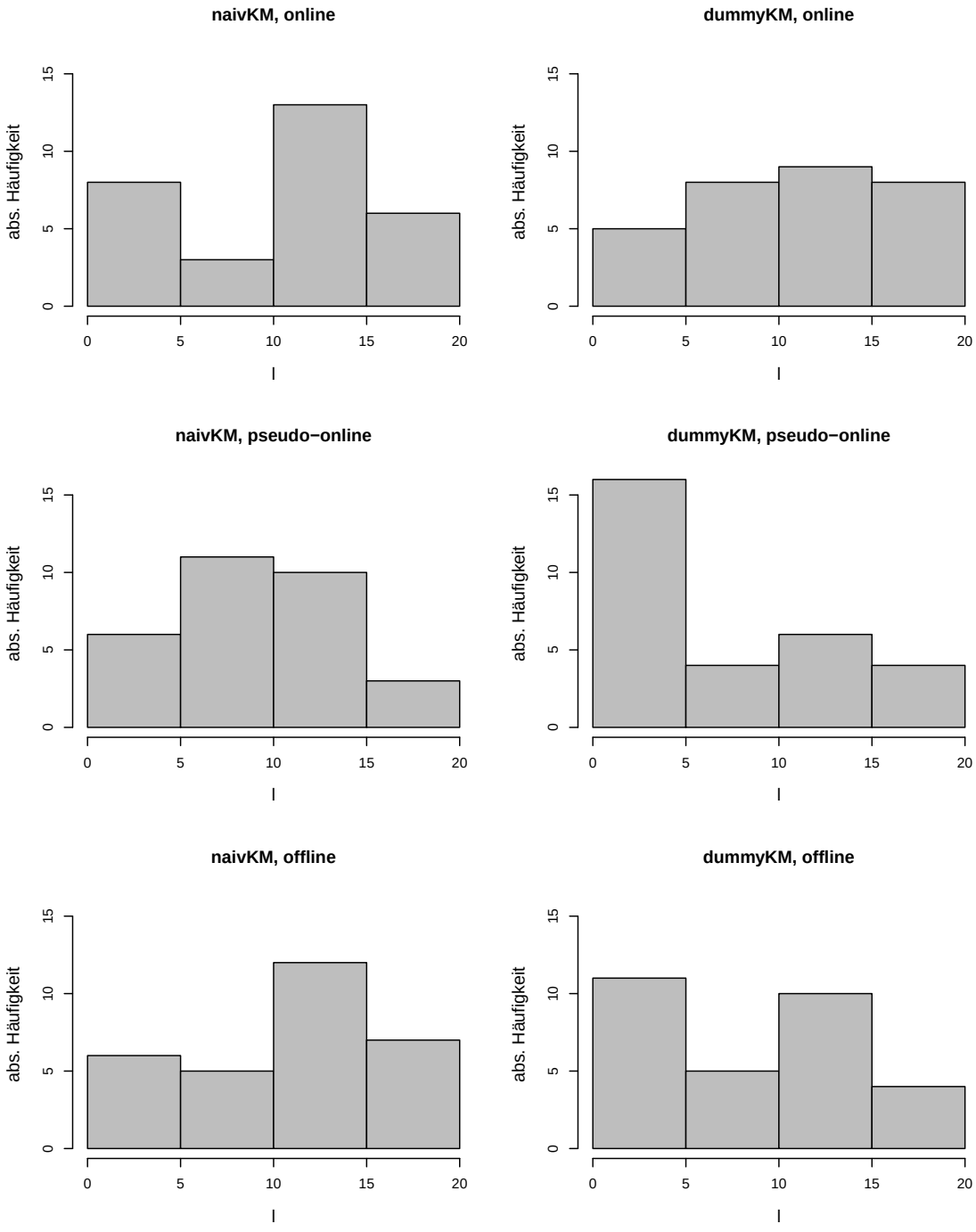

Abbildung A.9: *Verteilung der optimalen Einstellungen für den Parameter ℓ (Komprimierungsparameter der Logarithmierung). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

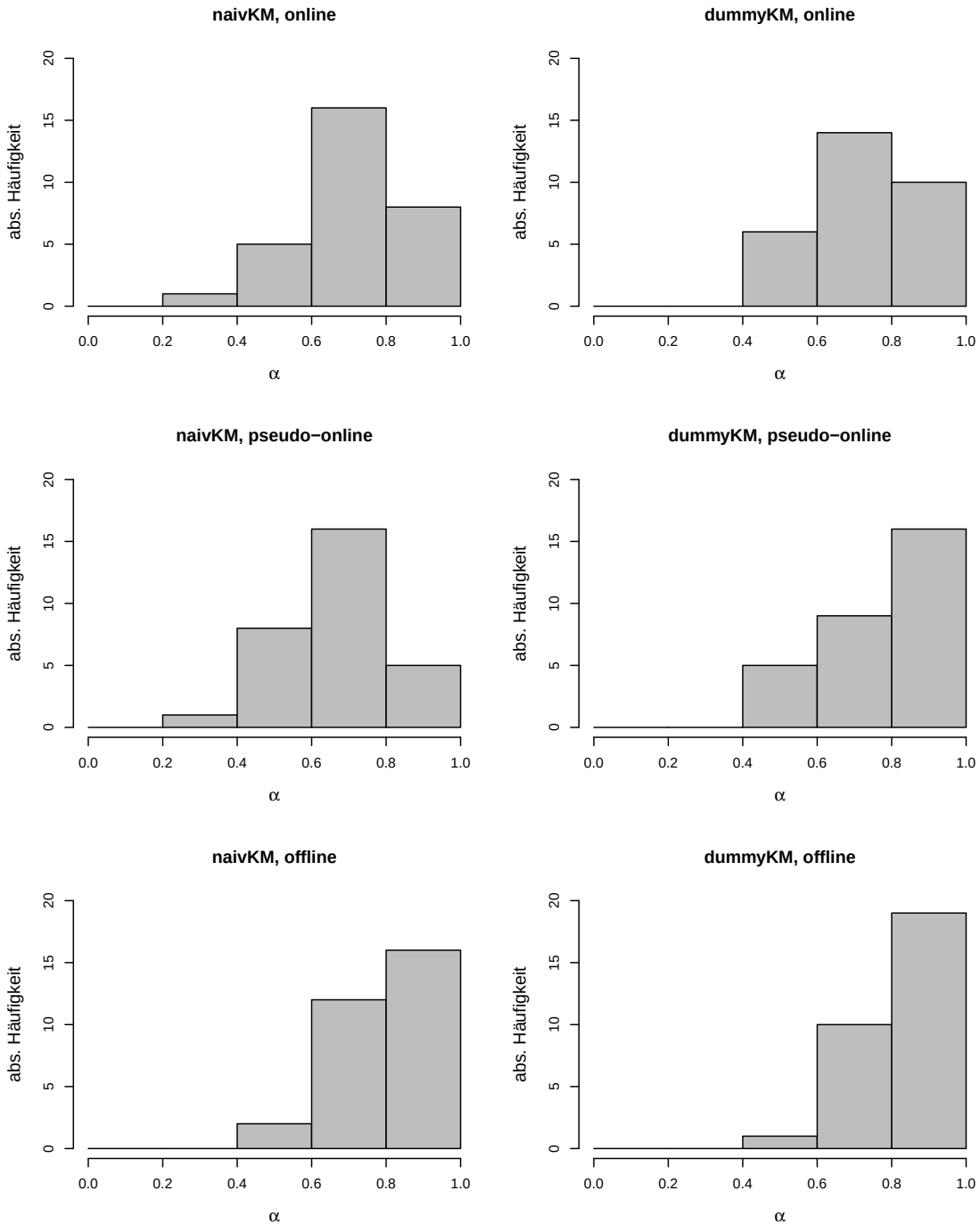

Abbildung A.10: *Verteilung der optimalen Einstellungen für den Parameter α (Parameter der exponentiellen Glättung). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

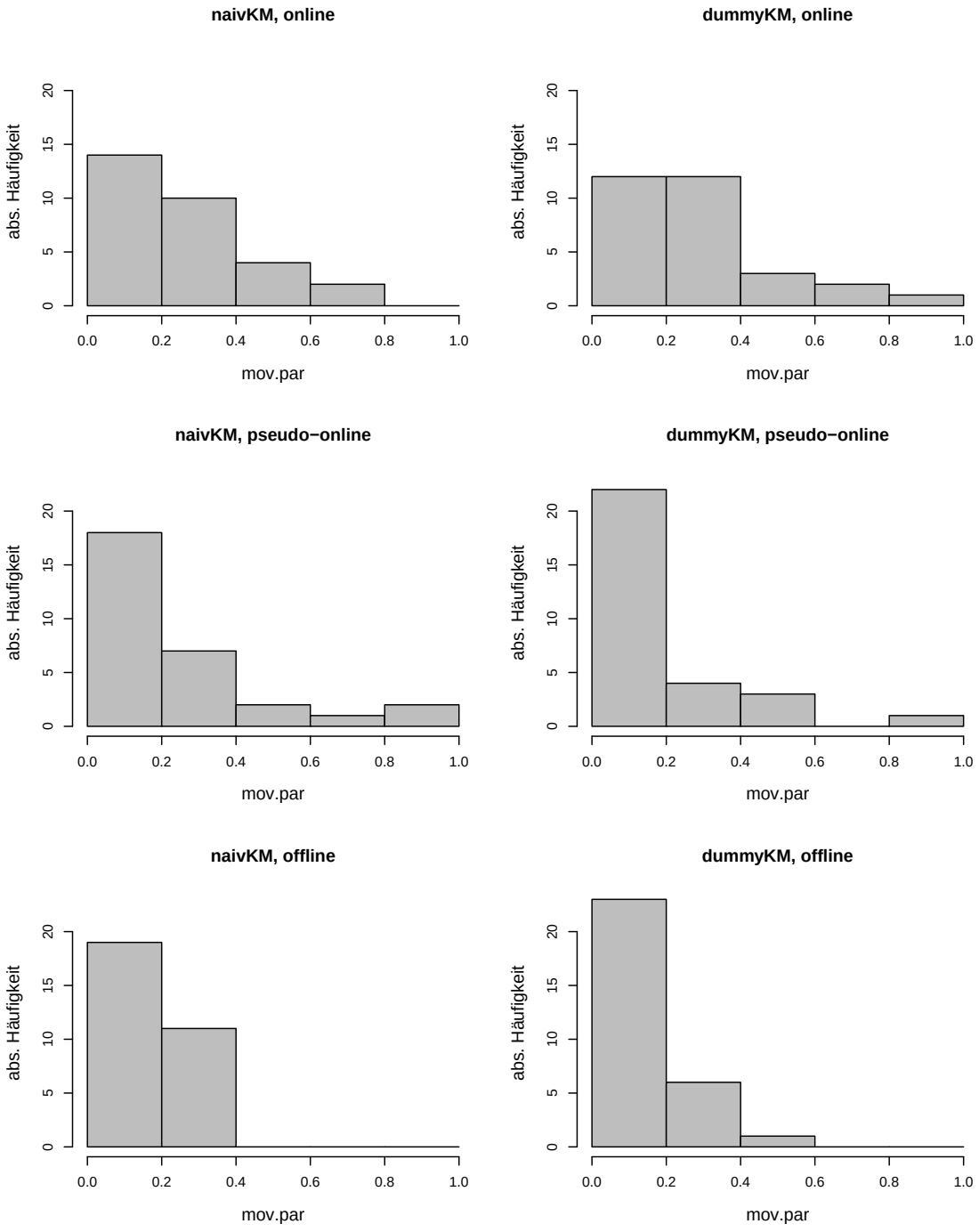

Abbildung A.11: *Verteilung der optimalen Einstellungen für den Parameter mov.par (Hilfsparameter der Schwellenwertfunktion). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

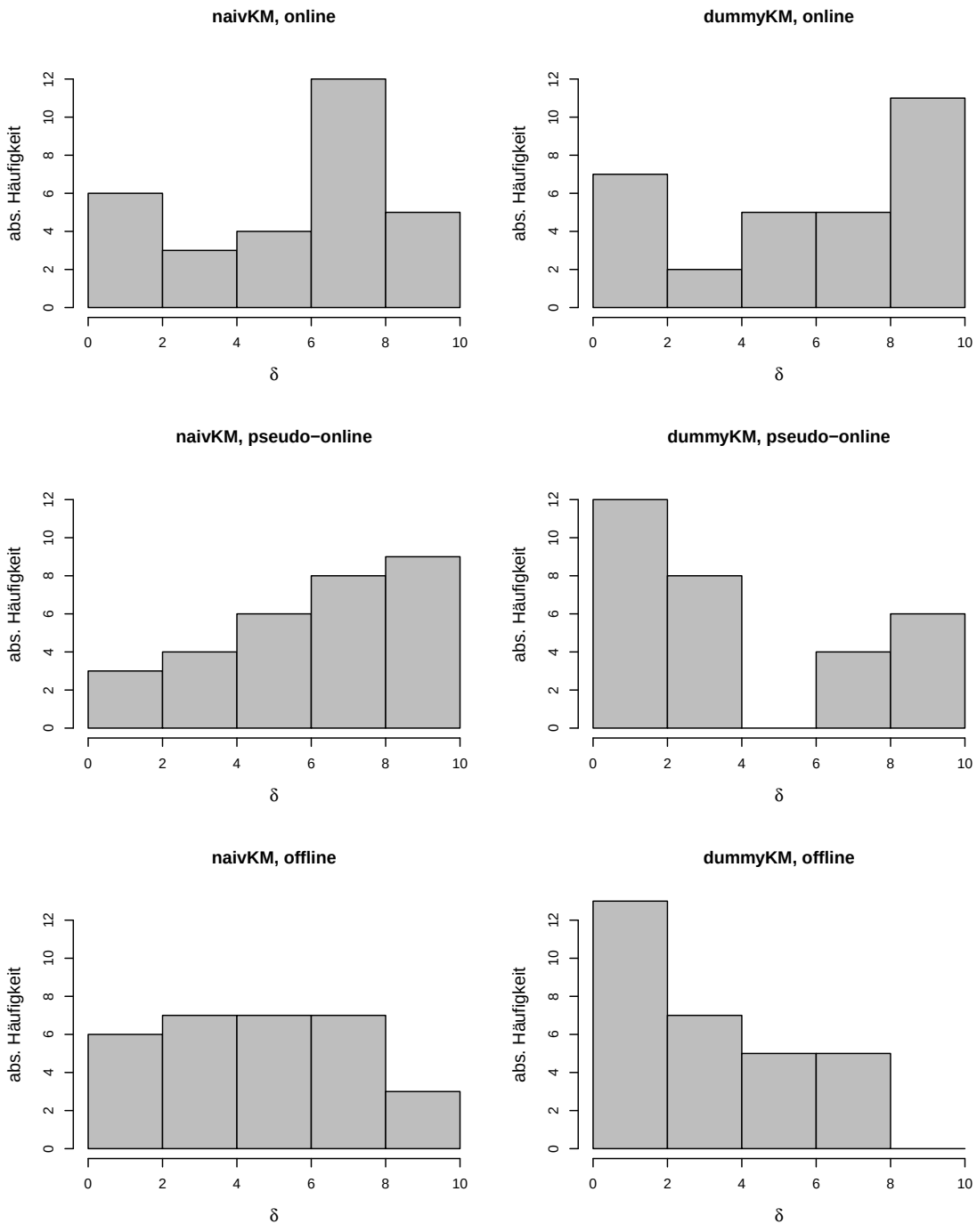

Abbildung A.12: *Verteilung der optimalen Einstellungen für den Parameter δ (additiver Parameter der Schwellenwertfunktion). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

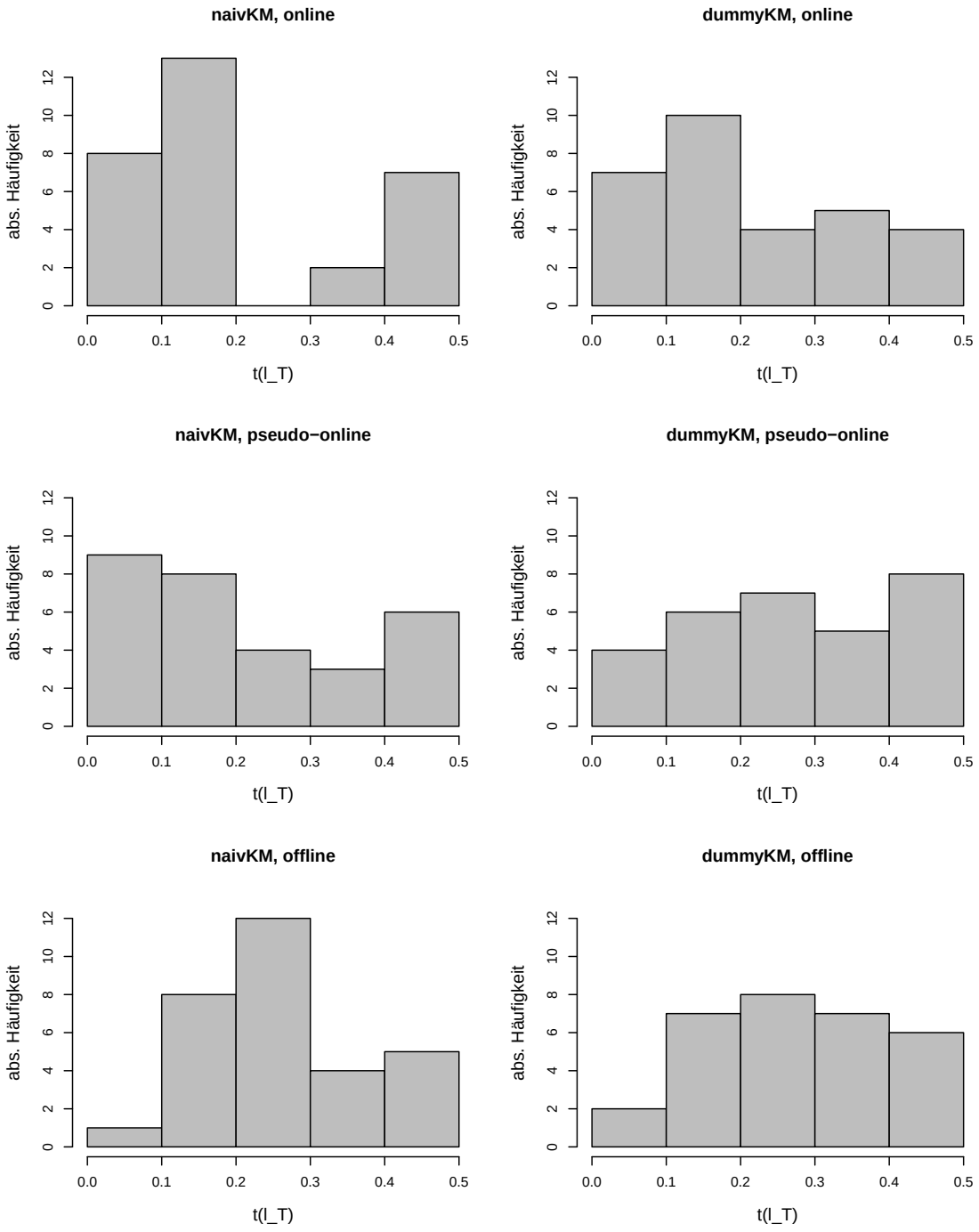

Abbildung A.13: *Verteilung der optimalen Einstellungen für den Parameter $t(l_T)$ (Zeit in die Vergangenheit für die Bestimmung der Schwellenwertfunktion). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

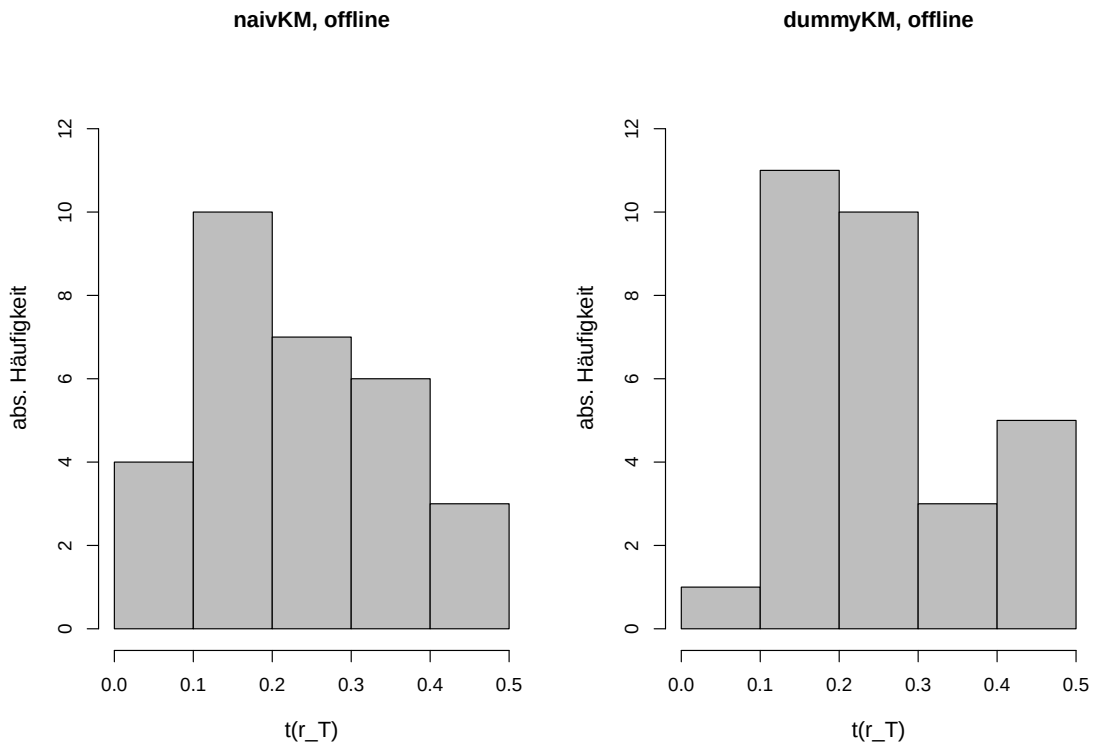

Abbildung A.14: *Verteilung der optimalen Einstellungen für den Parameter $t(r_T)$ (Zeit in die Zukunft für die Bestimmung der Schwellenwertfunktion). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

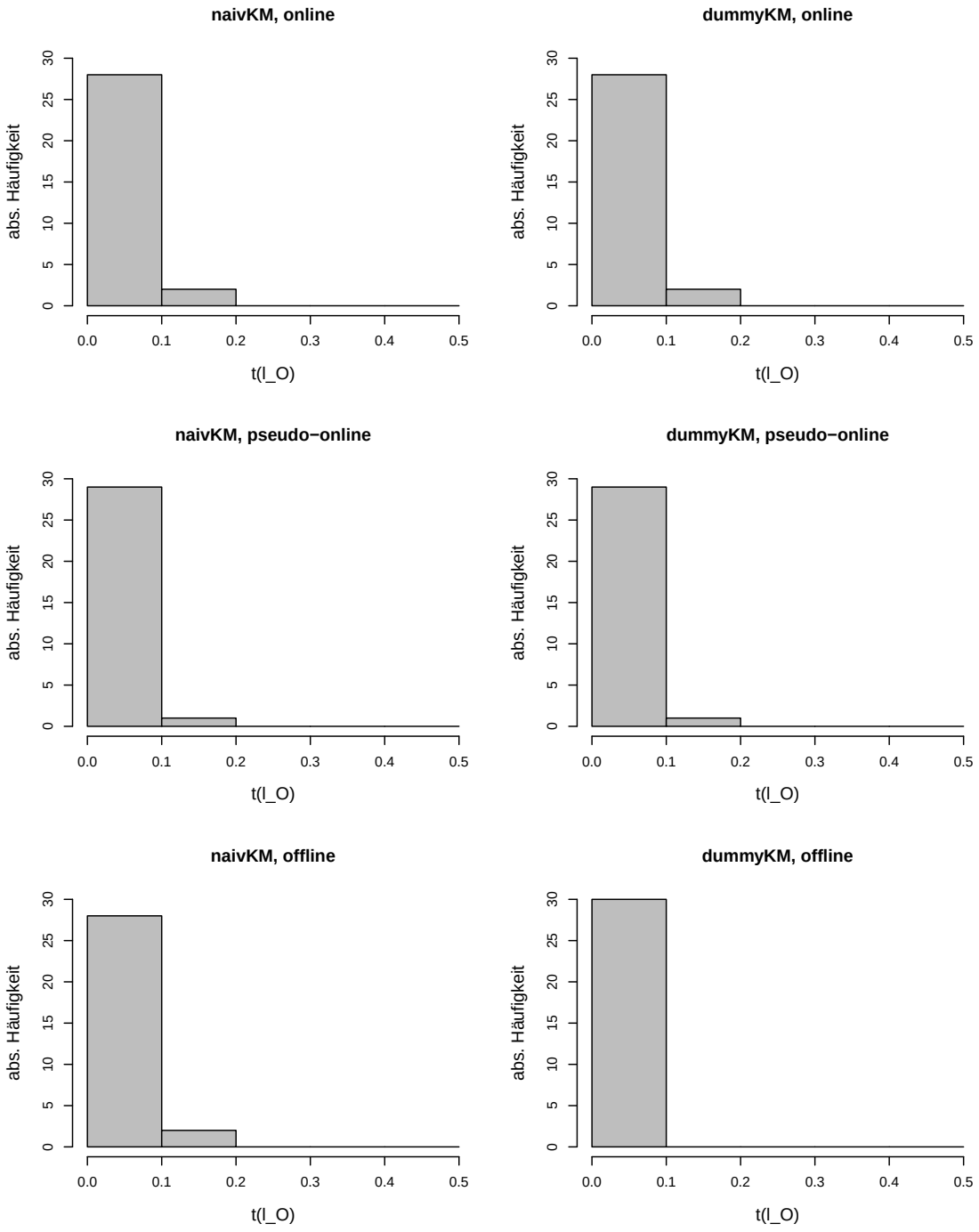

Abbildung A.15: *Verteilung der optimalen Einstellungen für den Parameter $t(l_O)$ (Zeit in die Vergangenheit für die Lokalisierung der Toneinsätze). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

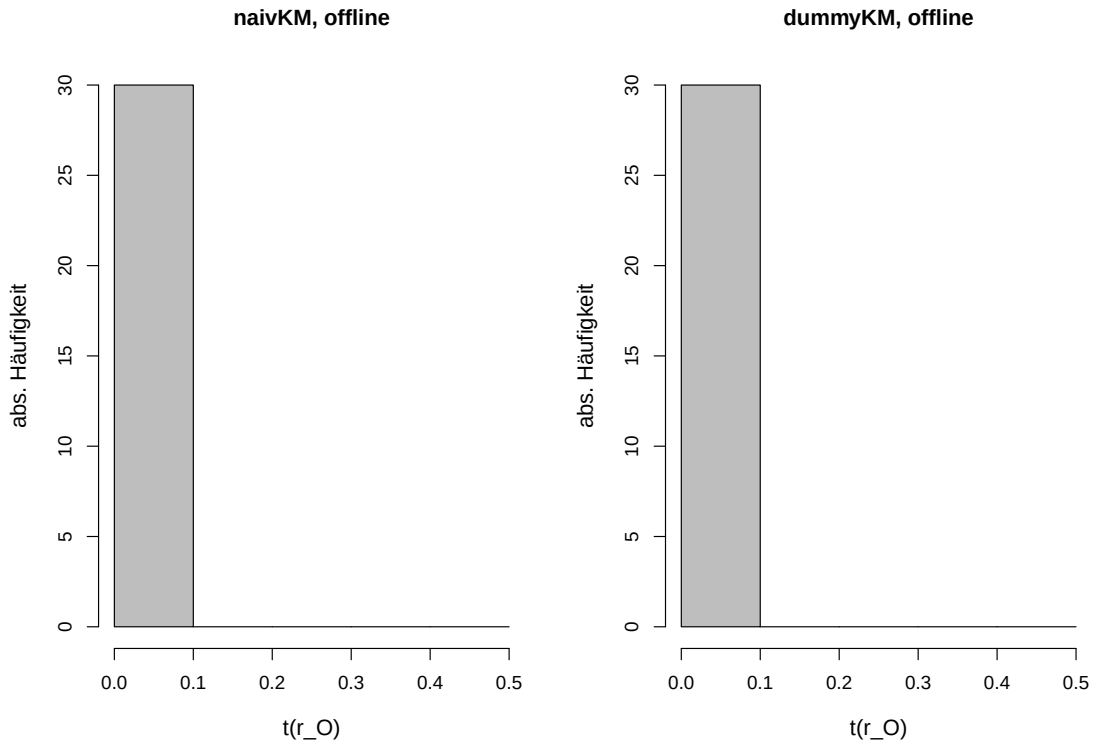

Abbildung A.16: *Verteilung der optimalen Einstellungen für den Parameter $t(r_O)$ (Zeit in die Zukunft für die Lokalisierung der Toneinsätze). Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

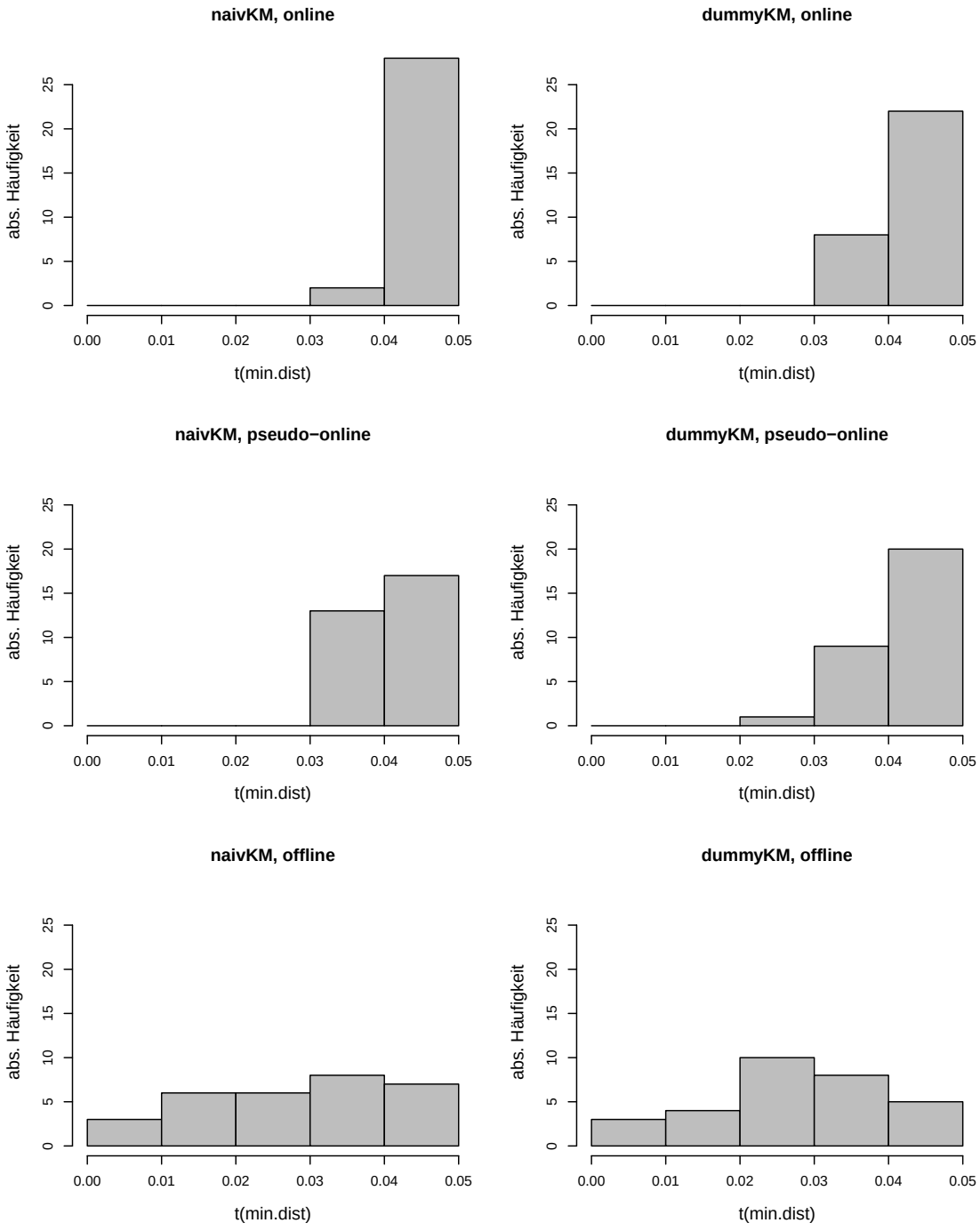

Abbildung A.17: *Verteilung der optimalen Einstellungen für den Parameter $t(min.dist)$. Links: Optimierungsläufe mit naivem Kriging als Surrogatmodell, rechts: mit dummy Kriging. Optimierungsmethode: MBO, Datensatz: WAV.*

Anhang B

Tabellen

Tabelle B.1: p-Werte des Wilcoxon-Rangsummentests für die jeweiligen Stichproben. *OF* steht für Optimierungsfrage und *AF* für die Anwendungsfrage (mit entsprechender Nummer). Jeder p-Wert kommt in der Tabelle zwei mal vor und jede Stichprobe definiert eine eigene Hypothesengruppe.

Stichprobe	*OF1*	*AF1*	*AF2*	*AF3*	*OF2*	*AF4*
MBO_naivKM_wav_online	0.4442	0.8430	$5.633 \cdot 10^{-7}$	$1.133 \cdot 10^{-15}$	0.1870	$1.252 \cdot 10^{-4}$
MBO_naivKM_wav_ps.online	0.7082	0.3659	$5.633 \cdot 10^{-7}$ 0.3980	$2.050 \cdot 10^{-14}$	0.1970	
MBO_naivKM_wav_offline	0.5619	0.9941	0.3980	$9.536 \cdot 10^{-10}$	0.3738	$2913 \cdot 10^{-6}$
MBO_naivKM_midi_online	0.8315		0.0023	$1.133 \cdot 10^{-15}$	0.0370	
MBO_naivKM_midi_ps.online	0.0041		0.0023 0.8087	$2.050 \cdot 10^{-14}$	0.0800	
MBO_naivKM_midi_offline	0.2601		0.8087	$9.536 \cdot 10^{-10}$	0.0415	
MBO_dummyKM_wav_online	0.4442					
MBO_dummyKM_wav_ps.online	0.7082					
MBO_dummyKM_wav_offline	0.5619					
MBO_dummyKM_midi_online	0.8135					
MBO_dummyKM_midi_ps.online	0.0041					
MBO_dummyKM_midi_offline	0.2601					
FMBO_naivKM_wav_online					0.1870	
FMBO_naivKM_wav_ps.online					0.1970	
FMBO_naivKM_wav_offline					0.3738	
FMBO_naivKM_midi_online					0.0370	
FMBO_naivKM_midi_ps.online					0.0800	
FMBO_naivKM_midi_offline					0.0415	
MBO_mulit.randForest_naivKM_wav_online						$1.252 \cdot 10^{-4}$
MBO_mulit.randForest_naivKM_wav_offline						$2.913 \cdot 10^{-6}$

Literaturverzeichnis

M. Abramowitz und I. A. Stegun. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publications, New York, 1965.

T. Bartz-Beielstein, C. Lasarczyk und M. Preuß. Sequential parameter optimization. In B. McKay et al., Hrsg., *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, Vol. 1, S. 773–780, Piscataway NJ, 2005. IEEE Press.

N. Bauer, J. Schiffner und C. Weihs. Einsatzzeiterkennung bei polyphonen Musikzeitreihen. Technischer Bericht 22/10, TU Dortmund, 2010.

N. Bauer, J. Schiffner und C. Weihs. Einfluss der Musikinstrumente auf die Güte der Einsatzzeiterkennung. Technischer Bericht 10/12, TU Dortmund, 2012.

N. Bauer, J. Schiffner und C. Weihs. Comparison of classical and sequential design of experiments in note onset detection. In B. Lausen, D. V. den Poel und A. Ultsch, Hrsg., *Data Analysis, Machine Learning and Knowledge Discovery*, S. 501–509. Springer International Publishing, 2013a.

N. Bauer, J. Schiffner und C. Weihs. Comparison of parameter optimization techniques for a music tone onset detection algorithm. In *Proceedings of the 4th Meeting on Statistics and Data Mining (MSDM)*, S. 28–34, 2013b. URL `http://www.tasa-online.com/`.

N. Bauer, K. Friedrichs, D. Kirchhoff, J. Schiffner und C. Weihs. Tone onset detection using an auditory model. In M. Spiliopoulou, L. Schmidt-Thieme und R. Janning, Hrsg., *Data Analysis, Machine Learning and Knowledge Discovery*, S. 315–324. Springer International Publishing, 2014.

N. Bauer, K. Friedrichs, B. Bischl und C. Weihs. Fast model based optimization of tone onset detection by instance sampling. In A. Wilhelm und H. A. Adalbert, Hrsg., *Analysis of Large and Complex Data*. Springer International Publishing, 2015.

J. P. Bello, L. Daudet, S. A. Abdallah, C. Duxbury, M. E. Davies und M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5-2): 1035–1047, 2005.

E. Benetos und S. Dixon. Polyphonic music transcription using note onset and offset detection. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, S. 37–40. IEEE, 2011.

Y. Bengio und Y. Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105, 2004.

B. Bischl, O. Mersmann, H. Trautmann und C. Weihs. Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20 (2):249–275, 2012.

B. Bischl, S. Wessing, N. Bauer, K. Friedrichs und C. Weihs. MOI-MBO: Multiobjective infill for parallel model-based optimization. In *Learning and Intelligent Optimization Conference (LION)*, S. 173–186, 2014.

B. Bischl, J. Bossek, D. Horn und M. Lang. *mlrMBO: Model-Based Optimization for mlr*, 2015a. URL `https://github.com/berndbischl/mlrMBO`. R package version 1.0.

B. Bischl, M. Lang, O. Mersmann, J. Rahnenführer und C. Weihs. BatchJobs and BatchExperiments: Abstraction mechanisms for using R in batch environments. *Journal of Statistical Software*, 64(11):1–25, 2015b.

B. Bischl, M. Lang, J. Richter, J. Bossek, L. Judt, T. Kuehn, E. Studerus und L. Kotthoff. *mlr: Machine Learning in R*, 2015c. URL `http://CRAN.R-project.org/package=mlr`. R package version 2.4.

S. Böck, F. Krebs und M. Schedl. Evaluating the online capabilities of onset detection methods. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR'12)*, S. 49–54, 2012.

D. Butler. *The Musician's Guide to Perception and Cognition*. Schirmer Books, New York, 1992.

T. Butz. *Fouriertransformation für Fußgänger*. Vieweg+Teubner-Verlag, Wiesbaden, Germany, 6 Ausgabe, 2009.

R. Carnell. *lhs: Latin Hypercube Samples*, 2012. URL `http://CRAN.R-project.org/package=lhs`. R package version 0.10.

C.-C. Chang und C.-J. Lin. Training ν-support vector regression: Theory and algorithms. *Neural Computation*, 14(8):1959–1977, 2002.

C. Chevalier und D. Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In G. Nicosia und P. Pardalos, Hrsg., *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, S. 59–69. Springer, 2013.

N. Collins. Using a pitch detector for onset detection. In *MIREX Online Proceedings (ISMIR'05)*, 2005. URL `http://dblp.uni-trier.de/db/conf/ismir/ismir2005.html#Collins05`.

J. Cooley und J. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

H. Dette, A. Pepelyshev und A. Zhigljavsky. Optimal designs in regression with correlated errors. Technischer Bericht 1/15, TU Dortmund, 2015.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

S. Dixon. Onset detection revisited. In *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx'09)*, S. 133–137, 2006.

C. Duxbury, M. Sandler und M. Davies. A hybrid approach to musical note onset detection. In *Proceedings of the 5th International Conference on Digital Audio Effects (DAFx'05)*, S. 33–38, 2002.

B. Efron und R. Tibshirani. Improvements on cross-validation: the .632+ bootstrap method. *Journal of the American Statistical Association*, (92):548–560, 1997.

F. Eyben, S. Böck, B. Schuller und A. Graves. Universal onset detection with bidirectional long short-term memory neural networks. In J. S. Downie und R. C. Veltkamp, Hrsg., *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR'10)*, S. 589–594. International Society for Music Information Retrieval, 2010.

R.-E. Fan, P.-H. Chen und C.-J. Lin. Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.*, 6:1889–1918, 2005.

W. J. Fu, R. J. Carroll und S. Wang. Estimating misclassification error with small samples via bootstrap cross-validation. *Bioinformatics*, 21(9):1979–1986, 2005.

D. Ginsbourger. *Multiples métamodèles pour l'approximation et l'optimization de functions numériques multivariables*. Doktorarbeit, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009.

D. Ginsbourger, R. Le Riche und L. Carraro. Kriging is well-suited to parallelize optimization. In Y. Tenne und C.-K. Goh, Hrsg., *Computational Intelligence in Expensive Optimization Problems*, Vol. 2 of *Adaptation, Learning, and Optimization*, S. 131–162. Springer, 2010.

D. Ginsbourger, V. Picheny, O. Roustant, C. Chevalier und T. Wagner. *DiceOptim: Kriging-based optimization for computer experiments*, 2013. URL `http://CRAN.R-project.org/package=DiceOptim`. R package version 1.4.

M. Goto, H. Hashiguchi, T. Nishimura und R. Oka. Rwc music database: Music genre database and musical instrument sound database. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR'03)*, S. 229–230. International Society for Music Information Retrieval, 2003.

F. Gouyon, F. Pachet und O. Delerue. Classifying percussive sounds: a matter of zero-crossing rate? In A. de Souza Britto Jr., F. Gouyon und S. Dixon, Hrsg., *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR'13)*, S. 523–528, 2013.

U. Groemping. *DoE.base: Full factorials, orthogonal arrays and base utilities for DoE packages*, 2014. URL `http://CRAN.R-project.org/package=DoE.base`. R package version 0.26-3.

J. Hammond und P. White. Signals and systems. In D. Havelock und M. Kuwano, Sonoko ans Vorländer, Hrsg., *Handbook of Signal Processing in Acoustics*, Vol. 1, S. 3–16. Springer, 2008.

F. J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *IEEE*, 66 (1):51–83, 1978.

J. A. Hartigan und M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 28:100–108, 1979.

T. Hastie, R. Tibshirani und J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, 2001.

S. Herbrandt, C. Weihs, U. Ligges, M. Ferreira, C. Rautert, D. Biermann und W. Tillmann. Optimization of a simulation for inhomogeneous mineral subsoil machining. In *Proceedings of the European Conference on Data Analysis*. Springer International Publishing, 2015.

S. Hess, T. Wagner und B. Bischl. Progress: Progressive reinforcement-learning-based surrogate selection. In G. Nicosia und P. Pardalos, Hrsg., *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, S. 110–124. Springer, 2013.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):pp. 65–70, 1979.

A. Holzapfel, Y. Stylianou, A. Gedik und B. Bozkurt. Three dimensions of pitched instrument onset detection. *IEEE Transactions on Audio, Speech, and Language Ppocessing*, 18 (6):1517–1527, 2010.

D. Horn, T. Wagner, C. Biermann, C. Weihs und B. Bischl. Model-based multi-objective optimization: Taxonomy, multi-point proposal, toolbox and benchmark. In A. Gaspar-Cunha, C. Henggeler Antunes und C. C. Coello, Hrsg., *Evolutionary Multi-Criterion Optimization (EMO)*, Vol. 9018 of *Lecture Notes in Computer Science*, S. 64–78. Springer, 2015.

X. Huang, A. Acero und H. Hon. *Spoken Language Processing - A Guide to Theory, Algorithm and System Development*. Prentice Hall, New Jersey, second Ausgabe, 2001.

F. Hutter, H. H. Hoos und K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. C. Coello, Hrsg., *Learning and Intelligent Optimization*, Vol. 6683 of *Lecture Notes in Computer Science*, S. 507–523. Springer, 2011.

F. Hutter, H. H. Hoos und K. Leyton-Brown. Parallel algorithm configuration. In Y. Hamadi und M. Schoenauer, Hrsg., *Learning and Intelligent Optimization Conference (LION)*, Lecture Notes in Computer Science, S. 55–70. Springer, 2012.

J. Janusevskis, R. Le Riche, D. Ginsbourger und R. Girdziusas. Expected improvements for the asynchronous parallel global optimization of expensive functions: Potentials and challenges. In Y. Hamadi und M. Schoenauer, Hrsg., *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, S. 413–418. Springer, 2012.

D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

D. R. Jones, M. Schonlau und W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

A. Klapuri. Introduction to music transcription. In A. Klapuri und M. Davy, Hrsg., *Signal Processing Methods for Music Transcription*, S. 3–20. Springer, New York, 2006.

J. Knowles. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10 (1):55–66, 2006.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, S. 1137–1143, San Francisco, 1995. Morgan Kaufmann Publishers Inc.

D. G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Metallurgical and Mining Society of South Africa*, 52(6):119–139, 1951.

W. J. Krzanowski und D. J. Hand. Assessing error rate estimators: the leave-one-out method reconsidered. *Australian Journal of Statictics*, 39(1):35–46, 1997.

A. Lacoste und D. Eck. A supervised classification algorithm for note onset detection. *EURASIP Journal on Applied Signal Processing*, 2007(1):1–13, 2007.

A. Liaw und M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

U. Ligges. *Transkription monophoner Gesangszeitreihen*. Doktorarbeit, TU Dortmund, 2006.

U. Ligges, S. Krey, O. Mersmann und S. Schnackenberg. *tuneR: Analysis of music*, 2014. URL `http://r-forge.r-project.org/projects/tuner/`.

J. London. Building a representative corpus of classical music. *Music Perception: An International Journal*, 31(1):68–90, 2013.

P. Masri. *Computer modeling of Sound for Transformation and Synthesis of Musical Signal*. Doktorarbeit, University of Bristol, 1996.

B. Matérn. *Spatial variation*. Lecture notes in statistics. Springer-Verlag, 1986.

Matlab. *version R2013a*. Natick, Massachusetts, 2013.

M. D. McKay, R. J. Beckman und W. J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2): 55–61, 1979.

R. Meddis. Auditory-nerve first-spike latency and auditory absolute threshold: A computer model. *Journal of the Acoustical Society of America*, 119(1):406–417, 2006.

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel und F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2015. R package version 1.6-7.

A. M. Molinaro, R. Simon und R. M. Pfeiffer. Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 2005.

D. Moore, P. Fuchs, A. Rees, A. Palmer und C. Plack. *Oxford Handbook of Auditory Science: Hearing*. Oxford Handbook of Auditory Science. OUP Oxford, 2010.

C. J. Morgan. *Theoretical and practical aspects of variography: in particular, estimation and modelling of semi-variograms over areas of limited and clustered or widely spaced data in a two-dimensional South African gold mining context*. Doktorarbeit, University of the Witwatersrand in Johannesburg, 2011.

B. Osgood. The fourier transform and its applications. 2007. URL `http://see.stanford.edu/materials/lsoftaee261/book-fall-07.pdf`.

G. Peeters und X. Rodet. A large set of audio feature for sound description (similarity and classification) in the cuidado project. Technischer bericht, Ircam, 2004.

V. Picheny, T. Wagner und D. Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3):607–626, 2013.

R. Polfreman. Comparing onset detection & perceptual attack time. In A. de Souza Britto Jr., F. Gouyon und S. Dixon, Hrsg., *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR'13)*, S. 523–528, 2013.

F. Pukelsheim. *Optimal Design of Experiments*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2006.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL `http://www.R-project.org/`.

C. E. Rasmussen und C. K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

A. Roebel. Onset detection in polyphonic signals by means of transient peak classification. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)*, London, Great Britain, Septembre 2005.

C. Rosão, R. Ribeiro und D. Martins De Matos. Influence of peak selection methods on onset detection. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR'12)*, S. 517–522, 2012.

J. Sacks, W. Welch, T. Mitchell und H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4:409–423, 1989.

F. Salfner, M. Lenk und M. Malek. A survey of online failure prediction methods. *ACM Computing Surveys*, 42, 2010. URL `http://doi.acm.org/10.1145/1670679.1670680`.

M. J. Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. Doktorarbeit, University of Michigan, 2002.

W. A. Schloss. On the automatic transcription of percussive music - from acoustic signal to high-level analysis. Master's thesis, Stanford University, 1985.

M. Schonlau. *Computer Experiments and Global Optimization*. Doktorarbeit, University of Waterloo, 1997.

J. S. Sekhon und W. R. Mebane, Jr. Genetic optimization using derivatives: Theory and application to nonlinear models. *Political Analysis*, 7:189–213, 1998.

S. Siegel und N. Castellan. *Nonparametric statistics for the behavioral sciences*. McGraw–Hill, Inc., second Ausgabe, 1988.

J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley, 2003.

M. L. Stein. A kernel approximation to the kriging predictor of a spatial process. *Annals of the Institute of Statistical Mathematics*, 43:61–75, 1991.

J. Stoer und F. Jarre. *Optimierung*. Springer, 2004.

D. Stowell und M. Plumbley. Adaptive whitening for improved real-time audio onset detection. In *Proceedings of the International Computer Music Conference (ICMC'07)*, S. 312–319, 2007.

M. J. van der Laan, E. C. Polley und A. E. Hubbard. Super learner. *Statistical Applications of Genetics and Molecular Biology*, 6(25), 2008.

C. Van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

E. Vazquez und J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and Inference*, 140(11): 3088–3095, 2010.

T. Voigt, R. Fried, M. Backes und W. Rhode. Threshold optimization for classification in imbalanced data in a problem of gamma-ray astronomy. *Advances in Data Analysis and Classification*, 8(2):195.216, 2014.

J. Vos und R. Rasch. The perceptual onset of musical tones. *Perception & Psychophysics*, 29(4): 323–335, 1981.

T. Wagner, M. Emmerich, A. H. Deutz und W. Ponweiser. On expected-improvement criteria for model-based multi-objective optimization. Lecture Notes in Computer Science, S. 718–727. Springer, 2010.

C. Weihs. Canonical discriminant analysis: comparison of resampling methods and convex-hull approximation. In O. Opitz, B. Lausen und R. Klar, Hrsg., *Information and Classification*, S. 225–238. Springer, Heidelberg, 1993.

C. Weihs und J. Jessenberger. *Statistische Methoden zur Qualitätssicherung und -optimierung in der Industrie*. Wiley-VCH, 1999.

S. M. Weiss und C. A. Kulikowski. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufmann Publishers Inc., San Francisco, 1991.