UNIVERSITÄT DORTMUND

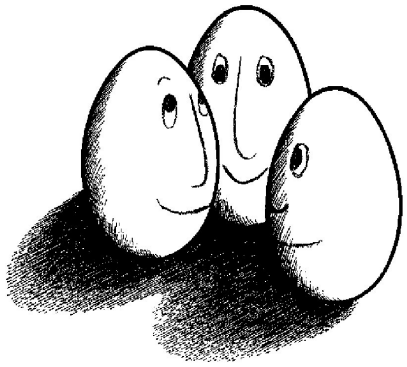
FACHBEREICH INFORMATIK

Diplomarbeit

Datengestützte Segmentierung von Audiodaten

Alexander Daxenberger

15.05.2007



Diplomarbeit am
Fachbereich Informatik
der Universität Dortmund

Betreuer
Prof. Dr. Katharina Morik
Dipl.-Inf. Ingo Mierswa

Inhaltsverzeichnis

Für Felix,
meinen halben und doch ganzen,
späten, jedoch nicht zu späten Bruder

Vorwort

Wenn ich Diplomarbeitstitel wie „Seismische Verfahren zur Kalibrierung von thermoisolaten Injektionsdrosseln in der Fertigung monophoner KFZ-Signalhörner“ lese, dann bin ich froh darüber, an ein nicht ganz so spezielles und anschauliches Diplomarbeitsthema geraten zu sein. Selbstverständlich war aber nicht der Titel bei meiner Wahl ausschlaggebend, sondern mein persönliches Interesse an der Musik, Klängen und der digitalen Klangverarbeitung. Von Anfang an hatte ich bei dem Thema ein Audioabspielprogramm oder auch -gerät vor Augen, welches bei Auswahl einer Audiodatei eine Segmentierung berechnet und mir kurz danach graphisch präsentiert, mit Hilfe derer man eine menschlich nachvollziehbare Struktur der Daten erkennen und zur Wiedergabe entscheidende Stellen direkt anspringen kann. Auch wenn diese Arbeit nicht unmittelbar zur Entwicklung eines solchen Systems führen wird, so hoffe ich trotzdem, dass sie einen guten Überblick über die bisherigen Arbeiten zu dem Thema und das heute Machbare liefert und meine Lösungsansätze und Ausführungen eine weitere kleine Ecke im Raum der Möglichkeiten ausleuchten.

Ich danke Frau Professor Dr. Katharina Morik für die Möglichkeit, diese Diplomarbeit am Lehrstuhl für künstliche Intelligenz geschrieben haben zu können, Herrn Dipl.-Inf. Ingo Mierswa für seine Hinweise, Hilfestellungen und seinen Mülheimer Humor und allen D&D-Seminarteilnehmern für die schmackhaften Frühstücksvorträge.
Ebenso danke ich meinen Eltern und Freunden für die Angebote und Durchführung von unterstützenden Tätigkeiten und Maßnahmen, ihre Gegenwart und Teilnahme.

1. Einführung und Zielsetzung

Die Übertragung und Archivierung von akustischen Inhalten wurde entwickelt, um Schallereignisse orts- und zeitunabhängig hörbar zu machen, und erfolgt inhaltsunabhängig auf der Ebene der physikalischen Vorgänge. In der Analogtechnik werden die Luftschwingungen dabei entweder direkt in elektrische Spannungsschwingungen umgesetzt oder dazu verwendet, ein elektrisches Trägersignal zu modulieren. Die Digitaltechnik, welche heutzutage aufgrund der vielfältigen Weiterverarbeitungsmöglichkeiten durch Computer hauptsächlich zum Einsatz kommt, geht dann noch einen Schritt weiter und codiert diese Spannungsschwingungen zeit- und wertediskret als Folge von Zahlen. Eine semantische Strukturierung dieser Audiodaten ist dabei nicht vorgesehen, was bedeutet, dass uns die akustischen Inhalte kaum (Signalton vor und nach Verkehrsdurchsagen in Radioübertragungen) oder nur recht grob strukturiert (Lieder auf einer CD) erreichen.
Im Zuge der zunehmenden Digitalisierung und automatischen Datenverarbeitung wurde es immer interessanter, Verfahren zu entwickeln, welche in menschenähnlicher Weise in der Lage sind, innerhalb eines unstrukturierten Audiodatenstromes semantische Einheiten zu erkennen und automatisch voneinander zu trennen. Dieser Vorgang der *Segmentierung von Audiodaten* beinhaltet Aufgaben wie z. B. die

- Sprecherwechselerkennung
- Themenwechselerkennung in Rundfunknachrichtensendungen
- Erkennung von Werbeblöcken in Rundfunksendungen
- Einteilung eines Audiodatenstromes in Abschnitte gleichen akustischen Inhalts
- Erkennung der Struktur von Musikstücken (Einleitung, Strophen, Refrains,...),

und Anwendungen, die sich die automatische Segmentierung und die Bewältigung obiger Aufgaben zu Nutze machen, sind

- Sprecher-Clustering als Vorverarbeitungsschritt in der automatischen Spracherkennung (Automatic Speech Recognition, ASR) zur Steigerung der Erkennungsleistung
- Filterung eines Audiodatenstromes auf bestimmte Inhalte zur Ausblendung unerwünschter Teile

- Erstellung von repräsentativen Zusammenfassungen von Musikstücken (music summarization, thumbnailing) für Musikdatenbanken (musical content retrieval)
- Erleichterte weil gezieltere Navigation durch einen Audiodatenstrom.

1.1. Überblick über existierende Lösungen und ihre Methoden

Es existieren bereits recht viele Vorschläge zu Verfahren zur Segmentierung von Audiodaten, die die verschiedensten Methoden verwenden, angefangen mit der gefensterten Suche nach Änderungen in den statistischen Eigenschaften der Merkmalsausprägungsfolge, über die Erkennung von geraden Linien in einer Ähnlichkeits- bzw. Korrelationsmatrix, bis hin zu Clusteringverfahren, Klassifikation mittels Support Vector Machines oder Modellierung als Hidden Markov Modell. Allen gemeinsam ist, dass sie jeweils nur für eine spezielle Anwendung vorgesehen bzw. auf einer Art von Audiodaten evaluiert wurden: Entweder geht es um die Segmentierung von rein musikalischen Audiodaten, rein sprachlichen Audiodaten oder um die Unterscheidung dieser beiden grundlegenden und weiterer Klangklassen (z. B. Geräusche, verschiedene Qualitäten, Überlagerungen der reinen Klassen).
Zur Kategorisierung haben sich drei Begriffe etabliert (Abb. 1.1) [CVR05, KSWW00]: „modellbasiert“, „metrikbasiert“ und „energiebasiert“. Während die „modellbasierten“ Verfahren überwachte Lernmethoden bemühen, bedienen sich die „metrikbasierten“ unüberwachter Lernmethoden. Die „energiebasierten“ oder auch „decoderbasierten“ Verfahren orientieren sich ausschließlich an der Energie des Audiosignals, erkennen also z. B. Stille als Segmentgrenze und sind nur sehr eingeschränkt brauchbar; diese werden deshalb nicht weiter betrachtet. Einige Verfahren beruhen auf der Suche nach sich wiederholenden Sequenzen in der Wertefolge der Merkmalsausprägungen. Obwohl diese Verfahren ebenfalls zu den unüberwachten Lernmethoden gehören, bezeichne ich sie im Folgenden als „sequenzbasiert“, da ihre Vorgehensweise bei der Entdeckung von Segmenten bzw. Definition einer semantischen Einheit sich grundlegend von der der anderen Verfahren unterscheidet.

„Modellbasierte“ Verfahren Im Bereich der **Musik** werden in [AS01] die verschiedenen polyphonen Klangbilder eines Liedes als Zustände eines Hidden Markov Modells modelliert und die wahrscheinlichste Zustandsfolge mittels Viterbi ausgegeben. In [AS02] erweiterten die Autoren dieses Verfahren um die Suche nach wiederkehrenden Teilfolgen durch eine Selbstähnlichkeitsanalyse in der Korrelationsmatrix der Zustandsfolge. Um zunächst die Anzahl K von (Klang-)Zuständen zu schätzen, nimmt das Verfahren aus [PLBR02] an Stellen großer und schneller Änderungen der Merkmalsausprägungen eine Vorsegmentierung vor. Darauf folgen ein Clustering mittels K-Means, das mit den Centroiden initialisierte Training eines Hidden Markov Modells (HMM) und die abschließende Decodierung der wahrscheinlichsten Zustandssequenz. Auch in [ANS+05] wird mit Hilfe eines Hidden Markov Modells

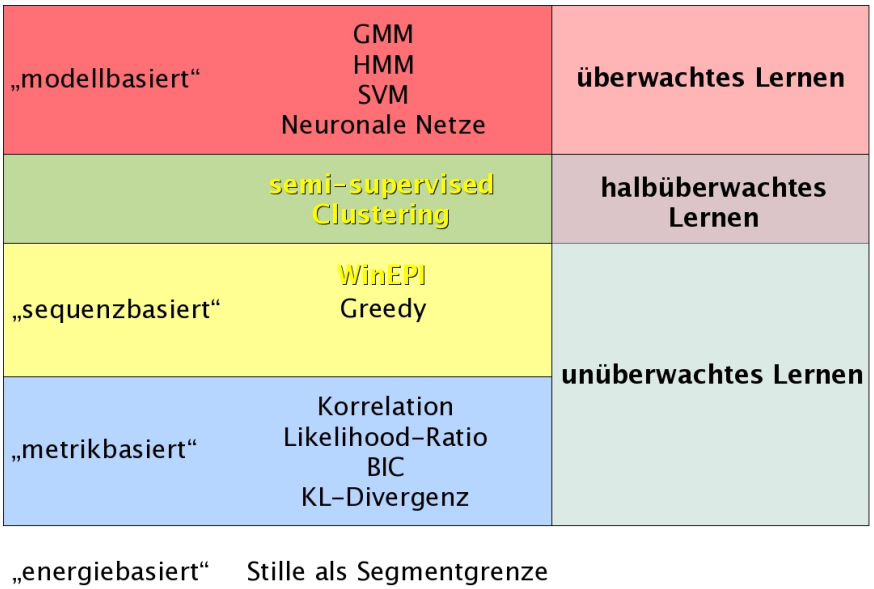


Abbildung 1.1.: Methoden für die Segmentierung von Audiodaten und eine Übersetzung der Begriffe „modellbasiert“, „sequenzbasiert“ und „metrikbasiert“ in die Fachsprache des maschinellen Lernens. Die gelb hervorgehobenen Begriffe sind Methoden, deren Eignung zur Segmentierung von Audiodaten noch nicht untersucht wurde.

und Viterbi die wahrscheinlichste Zustandssequenz berechnet, aus welcher eine Folge von Kurzzeit-Zustandshistogrammen erzeugt wird. Auf diesen Histogrammen werden dann vergleichend zwei Clusteringverfahren (paarweises und Histogramm-Clustering) angewendet, was schließlich zur Segmentbildung führt.
Im Bereich der **Sprache** bzw. **Nachrichtensendungen** werden in [KSWW00] drei Segmentierungsstrategien, energie-, metrik- und modellbasiert, mit einem hybriden Ansatz verglichen, welcher zunächst ein Bottom-up-clustering von kurzen sich nicht überlappenden Abschnitten fester Länge mittels Gish-Distanz und danach die Berechnung eines Gaussian Mixture Modells (GMM) für jeden Cluster vorsieht. Diese GMM werden als Emissionswahrscheinlichkeitsverteilungen eines HMM verwendet, worauf die Berechnung der wahrscheinlichsten Zustandssequenz folgt. Der hybride Ansatz zeigt sich dabei den anderen Verfahren überlegen. Die Autoren von [KS03] führen eine gefensterte und schwellenwertgesteuerte Sprecherwechselerkennung mit anschließendem hierarchischen agglomerativen Clustering der Sprecherwechselsegmente durch. Danach wird für jeden Cluster, also jede Sprachklasse, ein Hidden Markov Modell trainiert, welche dann zu einem größeren Hidden Markov Modell zur Sprachklassifikation kombiniert werden, um abschließend wieder per Viterbi die wahrscheinlichste Zustandsfolge zu ermitteln. In [AMB04] wird in einem dreistufigen,

schwellenwert- und heuristikfreien Prozess Sprache von Nicht-Sprache getrennt, eine Sprecherwechselerkennung und abschließend ein Sprechersegmentclustering durchgeführt, welcher Multi Layer Perceptrons, die Log Likelihood Ratio und Hidden Markov Modelle verwendet.
Zur Unterscheidung von grundlegenden **Klangklassen** werden in [Bia03] nach der Transformation des Audiosignals in eine Folge von Zero-Crossing-Intervallen auf dieser Darstellung basierende Merkmale gefenstert extrahiert und jeder Merkmalsausprägungsvektor mit Hilfe eines gaußschen Klassifizierers einer Klasse Stille, Sprache, Geräusche oder Applaus zugeordnet. Danach wird mittels dynamischer Programmierung eine Segmentierung berechnet, welche die Homogenität der Klassenkennzeichnungen innerhalb eines Segmentes maximiert. Noch besser funktioniert die Klassifikation der Merkmalsausprägungsvektoren in [LZL03], wo in einem binärbaumartigen Klassifikationsprozeß mittels Support Vector Machines (SVM) kurze Abschnitte fester Länge einer Merkmalsausprägungsfolge als Stille, Musik, Hintergrundgeräusche, Sprache oder Sprache mit Hintergrundgeräuschen erkannt werden und die Audiodatenfolge dementsprechend segmentiert wird. Eine Evaluation der Segmentgrenzen findet dort jedoch nicht statt. In [MN05] werden in einem Audiodatenstrom mit Hilfe eines Systemmoduls, welches metrik-, energie- und modellbasierte Methoden kombiniert, Stellen akustischer Änderungen als Segmentgrenzen erkannt, das Vorhandensein von bestimmten Inhalten angezeigt (Sprache, Nicht-Sprache, Sprechergeschlecht, Hintergrundgeräusche) und sprachliche Segmente gruppiert. Aufgrund der geringen Latenzzeit ist das Verfahren, welches ausgiebigen Gebrauch von Multi-Layer-Perceptrons (MLP) macht, zur strombasierten Audiodatenverarbeitung geeignet.
„Modellbasierte“ Verfahren haben immer den Nachteil des Trainingsaufwandes und der festen Anzahl von Klassen, wobei dies jedoch im Falle der Unterscheidung von grundlegenden Klangklassen weniger von Bedeutung ist. Hier sind die Klassen meist im Vorhinein bekannt und so generell zu beschreiben, dass die Modelle nur einmal gelernt werden müssen. Handelt es sich aber um Sprachaudiodaten mit einer unbekannten Anzahl von Sprechern oder Lieder, deren Klangbilder sehr verschieden sein und in einem komplexen Zusammenhang mit ihrer wahren Struktur stehen können, dann wiegen diese Nachteile deutlich schwerer.

„Metrikbasierte“ Verfahren Im Bereich der **Musik** wird in [JLC03] eine Segmentierung bestimmt, indem Merkmale, welche komplementäre musikalische Eigenschaften darstellen, auf signifikante Änderungen untersucht und diese heuristisch priorisiert werden. In [PE04] findet sich die Vorstellung eines Verfahrens zur skalenspezifischen Merkmalsgewichtung und eine gefensterte lokale Variante zur direkten Segmentierung, welche in dieser Arbeit implementiert und evaluiert wurde. Die Autoren von [Log00b] führen zum Auffinden von Schlüsselphrasen eines Liedes ein Bottom-up-clustering kurzer sich nicht überlappender Ab-

schnitte fester Länge mittels symmetrisierter Kullback-Leibler-Divergenz (Abschnitt A.4) durch, liefern jedoch keine Segmentierung.
Für Audiodaten aus dem Bereich der **Sprache** bzw. **Nachrichtensendungen** vergleicht [CVR05] die Komplexität dreier Implementierungen eines verbreiteten Algorithmus zur gefensterten Segmentierung basierend auf der Anwendung des Bayesian Information Criterion (BIC) und kommt zu dem Ergebnis, dass die lokalen Algorithmen nur minimal schlechtere Ergebnisse liefern, als ein globaler Algorithmus. Das Verfahren in [DKW99] hat sich bei kurzen Segmenten als besser erwiesen, als die „klassische“ BIC-Vorgehensweise. Dabei werden zunächst mittels der Generalized Likelihood Ratio (GLR) Distanzen zwischen benachbarten, in einer gewissen Schrittweite über die Merkmalsausprägungsfolge wandernden Fenstern berechnet und dann die Spitzen dieser Differenzfunktion als vorläufige Segmentgrenzen erkannt. In einem zweiten Durchgang werden diese Segmentgrenzen per BIC-Verfahren validiert oder verworfen. [HH04] versucht das Problem der kurzen Segmente anzugehen, indem in kleinen Fenstern das BIC-Maß durch eine neue Distanzmetrik („T^2-mean“) ersetzt wird. Darauf folgend wird zusätzlich ein Bottom-up-Clusteringverfahren zur relativen Klassifikation und durch paarweisen Vergleich adjazenter Segmente eine Fehlalarmkompensation durchgeführt. Ebenfalls auf das BIC stützt sich [CW04], wo zunächst mit großem Fenster vorsegmentiert und dann eine verfeinerte Suche mittels einer Divide-and-Conquer-Strategie innerhalb dieser Segmente gestartet wird.
Zur Unterscheidung von grundlegenden **Klangklassen** werden in [ZK99] die Segmentgrenzen und Segmentklassifikationen ausschließlich heuristisch und regelbasiert bestimmt. [WH06] versucht mittels einer hierarchischen binären Segmentierungsprozedur, die Beschreibungslänge eines gaußschen Modells der Merkmalsausprägungen in Abhängigkeit von der Anzahl und der Position der Segmentgrenzen global zu minimieren. Abschließend werden die Segmente als Sprache, Musik, Sprache mit Musik, Sprache mit Hintergrundgeräuschen oder Hintergrundgeräusche schwellenwertbasiert klassifiziert.
Insgesamt haben die „metrikbasierten“ Verfahren den Nachteil, dass sie explizit oder implizit Schwellenwerte, welche angepaßt werden müssen, oder Heuristiken verwenden. Hinzu gesellt sich das Problem der zuverlässigen Entdeckung sehr kurzer Segmente, innerhalb derer aufgrund der wenigen Daten keine genaue Berechnung von statistischen Kenngrößen möglich ist.

„Sequenzbasierte“ Verfahren Im Bereich der **Musik** berechnet das Verfahren in [FC03] eine Segmentierung durch Selbstähnlichkeitsanalyse mittels Kernel-Funktionskorrelation in der Ähnlichkeitsmatrix der Merkmalsausprägungsfolge und anschließenden Clustering der erkannten Segmente. Ebenfalls per Selbstähnlichkeitsanalyse geht [Got03] vor, indem nach Zeilenabschnitten hoher Ähnlichkeitswerte in der Ähnlichkeitsmatrix einer Chroma-Merkmalsausprägungsfolge gesucht wird und diese Abschnitte dann zu Segmenten zusam-

mengefügt werden. In [CV03] werden sich wiederholende Sequenzen mit Hilfe dynamischer Programmierung erkannt. Danach wird durch die Anwendung heuristischer Regeln auf die Liedstruktur geschlossen. [DH] beinhaltet die Vorstellung zweier greedy-Strategien zur Suche nach sich wiederholenden Strukturen, zum einen nach monophoner Transkription, zum anderen nach Extraktion von Chroma-Merkmalen.
Die „sequenzbasierten“ Verfahren suchen nach ähnlichen Passagen innerhalb von Liedern, welche für die Musik ein wichtiges Strukturierungskriterium darstellen, sind jedoch auf den Bereich der musikalischen Audiodaten beschränkt und bieten kaum Spielraum zur Anpassung an andere Segmentierungsaufgaben. Außerdem findet keine, im Falle von [Got03] keine genreübergreifende Evaluation statt.

1.2. Schwierigkeiten

Erste Probleme treten schon bei der Definition der semantischen Einheit, also der Frage danach, in welcher Weise die zu erkennenden Segmente Einheiten bilden, auf. In [AMB04] wird die Aufgabe der Segmentierung von Audiodaten wie folgt beschrieben:

> *„Audio segmentation, in general, is the task of segmenting a continuous audio stream in terms of acoustically homogenous regions, where the rule of homogeneity depends on the task.“*

Für viele Anwendungen bringt diese Aussage es auf den Punkt, und doch ist es nur ein Teil der Wahrheit, denn in manchen Fällen ist die akustische Homogenität kein ausreichendes Entscheidungskriterium zur Erkennung von Segmenten. In der Musik, insbesondere in der klassischen, definieren sich zusammengehörige Teile manchmal fast ausschließlich über wiederholt auftretende Sequenzen von Schallereignissen, was in ihrer Natur liegt, denn Musik kann man auch als *strukturierte Erzeugung von Klängen* beschreiben (vgl. [AS01, CV03]). Um beispielsweise das Thema eines reinen Klavierstücks als Segment zu erkennen, reicht es nicht, nach unterschiedlichen Klangbildern zu segmentieren, sondern es müssen zeitliche Strukturen beachtet werden. Dies wird in [AS01] als „Anti-Clustering-Situation“ beschrieben. (Der Wahrnehmung von Wiederholungen bei dem Vorgang des „Musikhörens“ wird entscheidende Bedeutung zugeschrieben [DH02].)
Eng verbunden mit der Definition der semantischen Einheit ist die Wahl der Merkmale, welche die akustischen Eigenschaften des Audiomaterials repräsentieren sollen. Doch je nach Art der Anwendung, also ob man z. B. einzelne Sprecher voneinander, Sprache von anderen akustischen Phänomenen, wie Musik oder Geräuschen, oder einfach nur laute von leisen Abschnitten unterscheiden möchte, eignen sich manche Merkmale besser als andere zur Differenzierung der Segmente.
Hat man für einen bestimmten Merkmalsatz ein Modell zur Unterscheidung von Segmenten bzw. Erkennung von Segmentgrenzen gelernt, so ist es unwahrscheinlich, dass dieses Modell

für andere Segmentierungsaufgaben erfolgreich eingesetzt werden kann; z. B. eignet sich ein Modell zur Sprecherwechselerkennung kaum dazu, Werbeblöcke vom Rest einer Radioübertragung zu trennen und umgekehrt, denn diese beiden Aufgaben bedürfen verschiedener Differenzierungskriterien. Die Nicht-Übertragbarkeit von Modellen stellt also Anforderungen an die Adaptivität eines Verfahrens.
Aber auch in Bezug auf die Granularität bzw. die Skalierung der Segmentierung sollte das Verfahren anpassungsfähig sein, denn je nachdem, unter welcher Skala man Audiodaten betrachtet, was auch wieder anwendungsabhängig ist, lassen sich sowohl relativ kurze als auch lange Segmente sinnvoll erklären. Beispiele hierfür sind Segmente auf der Phonem-, Wort- oder Sprecherebene von Sprach-Audiodaten.
Ein weiteres Problem stellt die Beurteilung der Güte eines Segmentierungsverfahrens dar, da, außer im Falle von künstlich erzeugten Testaudiodatensätzen, meistens keine optimalen Segmentgrenzen definierbar sind. In [TC99] begegnete man dem dadurch, dass nur solche Segmentgrenzen zur Evaluierung herangezogen wurden, über deren Existenz sich eine Mehrheit von menschlichen Segmentierern einig waren, und in [DKW99] wurden die berechneten Segmentgrenzen subjektiv bewertet.

1.3. Zielsetzung

Ein Ziel dieser Arbeit ist es, einen Überblick über die existierenden Ansätze zur Lösung des Problems der Segmentierung von Audiodaten und die dabei verwendeten Methoden zu liefern.
Da die mir bekannten Verfahren den Nachteil haben, dass sie in ihrer Vorgehensweise und/oder der Wahl der Audiomerkmale auf eine bestimmte Audiodatenart spezialisiert sind, möchte ich Verfahren erarbeiten und implementieren, welche überwachte und halbüberwachte Methoden aus dem Gebiet des maschinellen Lernens nutzen, darunter eine Methode des *Constrained clustering*, welche für die Segmentierung von Audiodaten noch nicht eingesetzt wurde, um ihre Fähigkeit zur Anpassung an verschiedene Segmentierungsaufgaben sicherzustellen.
Die Evaluation der realisierten Verfahren auf Audiodaten verschiedener Arten, auch im Hinblick auf die Frage, ob sich der Benutzer- und Lernaufwand im Vergleich zur erzielten Segmentierungsgüte lohnt, soll den Abschluß dieser Arbeit bilden. Dies beinhaltet die Erstellung eines Datensatzes und eine Gegenüberstellung der Verfahren.

Kapitelübersicht Kapitel 2 ist den Grundlagen für die Verarbeitung von akustischen Signalen im Hinblick auf deren Strukturierung gewidmet. Es sollen Fragen danach, was **Audiodaten** sind, welche Arten es zu betrachten gilt und nach welchen Kriterien sie strukturiert werden können, was akustische **Merkmale** sind und in welcher Hinsicht sie das akustische

Signal beschreiben, beantwortet werden. Anschließend werden **ausgewählte Beispiele** bereits existierender Vorgehensweisen zur Lösung des Problems der Segmentierung dargestellt. In Kapitel 3 stelle ich die von mir **implementierten Segmentierungsverfahren** vor. Als Vergleichsbasis dient ein relativ einfaches, unüberwachtes, „metrikbasiertes“ Verfahren, welches auf Distanzberechnungen zwischen benachbarten Fenstern beruht. Das zweite verwendet Support Vector Machines zur Klassifikation von Merkmalausprägungsvektoren, und das dritte macht von einem halbüberwachten Constrained-clustering-Verfahren Gebrauch. Als Nachverarbeitungsschritt wird bei den beiden letzten Verfahren eine **Ausreißerbehandlung** durchgeführt.
Über die Funktionen und die Bedienung des von mir für die Durchführung der Experimente entwickelten Programms ***Segfried*** gebe ich in Kapitel 4 einen kurzen Überbick.
Kapitel 5 enthält die Beschreibung der durchgeführten **Experimente** und die ausführlichen **Ergebnisse**, wobei die Vorgehensweise bei der Evaluation näher erläutert wird. Abschließend folgt die **Auswertung**, welche einen Vergleich der realisierten Verfahren sowohl untereinander als auch mit bereits existierenden Lösungen anderer Arbeiten beinhaltet.
Das letzte Kapitel 6 fasst die geleisteten Arbeiten zusammen und soll einen Ausblick auf weitere Anstrengungen bieten, die zu unternehmen sich vielleicht lohnen.

2. Segmentierung von Audiodaten

Die Audiodaten, auf denen wir arbeiten, um in ihnen Strukturen zu entdecken, kann man als unterste Stufe der digitalen Repräsentation von akustischen Phänomenen, also Schwingungen von Luftteilchen, betrachten. Technisch gesehen sind Audiodaten die direkte und diskretisierte Beschreibung der Bewegung von schallsensitiven oder -produktiven Elementen, wie z. B. Mikrophon- oder Lautsprechermembranen. Dabei ist zu normalerweise äquidistanten Zeitpunkten ein zur Abweichung der Position des Elementes von seiner Ruheposition proportionaler Wert gegeben, den man als *Auslenkung* oder *Elongation* bezeichnet.
Mathematisch betrachtet sind Audiodaten Zeitfolgen, also Abbildungen $t : \mathbb{N} \to \mathbb{R} \times \mathbb{R}$, wobei eine Dimension des Bildvektors den Zeitpunkt und die andere die Auslenkung angibt. Um jedoch sowohl Audiodaten als auch ihre Transformationen einheitlich beschreiben zu können, wurde in [MM05] der Begriff der „Wertefolge“ als Generalisierung der Zeitfolge eingeführt:

Definition 2.1. (Wertefolge) Eine Wertefolge ist eine Abbildung

$$\begin{aligned} x : \mathbb{N}_0 &\to \mathbb{R} \times \mathbb{C}^k \\ i &\mapsto (x_i^d, (v_0, \ldots, v_{k-1})_i = x_i^v) = x_i \end{aligned}$$

mit $k \in \mathbb{N}$, und wir schreiben $(x_i)_{i\in\{0,\ldots,n-1\}}$ für eine Wertefolge der Länge n und $(x_i)_{i\in\{l,\ldots,m\}}$ mit $0 \leq l \leq m < n$ für eine Teilfolge dieser Wertefolge. Der zu jedem Wertevektor x_i^v gegebene Zeitpunkt x_i^d wird als *displacement* bezeichnet. Für $k = 1$ heißt die Wertefolge *univariat*, für $k > 1$ *multivariat*.

In Tab. 2.1 sind die wichtigsten akustischen Phänomene und Beispiele für Elemente, welche ihnen eine für uns interessante Struktur geben, aufgeführt. Um nun einen Audiodatenstrom eines solchen Phänomens erfolgreich zu segmentieren, also die strukturgebenden Elemente voneinander abzugrenzen bzw. den Anfangs- und Endzeitpunkt jedes Elementes anzugeben, ist es von Nutzen, sich Gedanken darüber zu machen, in welcher Weise sich die verschiedenen Elemente voneinander unterscheiden bzw. was charakteristisch für ein bestimmtes Element ist. Das Ziel ist es, geeignete Kriterien zu finden, anhand derer dies möglich ist, was uns zum Thema der Audiomerkmale führt.

Musik	Einleitung, Strophe, Refrain, Solo, Zwischenteil, Schluß, Satz
Rundfunkübertragungen	Musik, Werbung, Themen, Szenen, Sendungen, Moderation, Sprecher
Telefongespräche	Sprecher
Konferenzmitschnitte	Sprecher, Vorträge, Beifall, Zurufe
Hörspiele	Sprecher, Szenen, Musik
Hörbücher	Sprecher, Kapitel

Tabelle 2.1.: Akustische Phänomene und ihre strukturgebenden Elemente

2.1. Merkmale

Merkmale sind gewisse Eigenschaften eines Datensatzes, welche durch eine Transformation der Daten offenbar werden. Diese Transformation dient dazu, die Daten in eine für die weitere Verarbeitung geeignete Darstellung zu überführen, die diese Eigenschaften repräsentiert, z. B. um daraus ein Modell zu lernen. Die Antwort auf die Frage, welche Eigenschaften eines Datensatzes von Bedeutung sind, ist natürlich anwendungsabhängig und kann durch Überlegung, aber auch durch automatische Verfahren wie z. B. in [MM05] gefunden werden. In unserem Fall bilden Audiodaten in oben beschriebener Repräsentation den Datensatz. Diese Darstellung ist jedoch für unsere Aufgabe der Segmentierung aus mindestens zwei Gründen ungeeignet. Zum einen beurteilen wir Menschen akustische Phänomene nicht auf der Ebene der zeitlichen Schwingungen der Luftteilchen, sondern orientieren uns an Klangbildern, deren Entwicklung und der abstrakteren Eigenschaft der Lautstärke. Der Grund dafür ist, dass das akustische Signal durch ein Organ im Innenohr, der *Cochlea* oder *Hörschnecke*, vom Zeitbereich in den Frequenzbereich transformiert [WIK07] wird, bevor es unser Gehirn erreicht, was vom Prinzip her einer Fouriertransformation entspricht (siehe Abschnitt A.3). Ein weiterer Grund für die Transformation des Audiodatenmaterials ist das Datenvolumen, welches für die weitere Verarbeitung zu groß und unhandlich ist.
Der erste Schritt auf dem Weg zur Lösung unseres Problems ist also die Extraktion von abstrahierenden Merkmalen.

2.1.1. Audiomerkmalsextraktion

Allgemein ausgedrückt ist eine Audiomerkmalsextraktion eine wie auch immer geartete Transformation einer Audiodatenfolge. Sie dient der Datenreduktion und der kompakten Darstellung wesentlicher Eigenschaften des Audiomaterials, welche bei der akustischen Wahrnehmung des Menschen eine Rolle spielen.

Definition 2.2. (Audiomerkmalsextraktion) Eine Audiomerkmalsextraktion ist eine Abbildung

$$\begin{aligned} \mathcal{F} : X' &\rightarrow \mathbb{C}^k \\ x = (x_i)_{i\in\{l,\ldots,m\}} &\mapsto (f_0, \ldots, f_{k-1}) = f \end{aligned}$$

mit $X' \subset X$ und $X = \{(x_i)_{i\in\{l,\ldots,m\}} | 0 \leq l \leq m < n\}$. Der Vektor f wird auch als Merkmalsausprägungsvektor bezeichnet.

Für die Aufgabe der zeitlichen Segmentierung ist es nun notwendig, dass bei der Transformation der Daten der Bezug zur Zeitdimension erhalten bleibt. Die Originalwertefolge wird dazu gleichförmig in viele kurze, im Normalfall benachbarte oder sich überlappende Teilfolgen oder „Fenster“ zerlegt. Jedes Fenster wird dann einer beliebigen Transformation unterzogen und das Ergebnis wieder einem Zeitpunkt zugeordnet, in unserem Fall dem Mittelpunkt des Zeitabschnitts des entsprechenden Fensters:

Definition 2.3. (Gefensterte Audiomerkmalsextraktion) Eine gefensterte Audiomerkmalsextraktion ist eine Abbildung

$$\begin{aligned} W : X^* &\rightarrow X^* \\ x = (x_i)_{i\in\{0,\ldots,n-1\}} &\mapsto \hat{x} = (\hat{x}_j)_{j\in\{0,\ldots,|\frac{n-w}{s}|\}} \end{aligned}$$

mit

$$\begin{aligned} X^* &= \left\{x = (x_i)_{i\in\{0,\ldots,a\}} | a \in \mathbb{N}_0\right\} \\ \hat{x}_j &= \left(\hat{x}_j^d, \mathcal{F}\left((x_i)_{i\in\{j*s,\ldots,(j*s)+w-1\}}\right)\right) \quad \hat{x}_j^d = \tfrac{1}{2}\left(x_{j*s}^d + x_{(j*s)+w-1}^d\right), \end{aligned}$$

w als Fenstergröße und s als Schrittweite. $\hat{x}$ wird als Merkmalsausprägungsfolge bezeichnet.

Für jedes Fenster werden also die Ausprägungen bestimmter Merkmale berechnet.

2.1.2. Grundlegende Audiomerkmale

In den folgenden Abschnitten möchte ich die Audiomerkmale, welche in dieser Arbeit verwendet werden, vorstellen und beispielhaft einige weitere Gebräuchliche nennen und erläutern. Zunächst gehe ich dabei auf Merkmale ein, die relativ direkt aus den Audiodaten gewonnen werden können bzw. grundlegende Eigenschaften des Audiomaterials darstellen, und man kann bezüglich ihrer Berechnung zwischen Merkmalen im Zeitbereich, Frequenzbereich und Phasenraum unterscheiden [TC02, LZL03, Mie03, Got03].

Audiomerkmale im Zeitbereich

Die Merkmale im Zeitbereich werden direkt aus der Zeitfolgendarstellung gewonnen bzw. repräsentieren zeitliche Abläufe.

Lautstärke Die Energie eines akustischen Signals bestimmt seine Lautstärke, welche ein für die Unterscheidung von Segmenten ein sehr grundlegendes Merkmal ist. Als Maß für die Lautstärke LD wird oft die Quadratwurzel aus der Signalenergie, also das quadratischen Mittel oder *Root Mean Square (RMS)*, verwendet [TC99]. Für eine Teilfolge $(x_i)_{i\in\{l,\ldots,m\}}$ einer univariaten Wertefolge gilt:

$$LD\left((x_i)_{i\in\{l,\ldots,m\}}\right) = \sqrt{\frac{1}{m-l+1}\sum_{i=l}^{m} x_i^{v\,2}}$$

Zero Crossing Rate Die *Zero Crossing Rate* ZCR ist ein Maß für die Anzahl von Vorzeichenwechseln im Audiosignal. Dies liefert eine grobe Charakterisierung des Spektrums bezüglich des Rauschanteils und hat sich besonders bei der Unterscheidung von Sprach- und Musiksignalen bewährt (vgl. [TC02, LZL03]):

$$ZCR\left((x_i)_{i\in\{l,\ldots,m\}}\right) = \frac{1}{2*(m-l)}\sum_{i=0}^{m-1}\left|sgn(x_{i+1}^v) - sgn(x_i^v)\right|$$

Flux Der *Flux* ist zunächst einmal ein generisches Merkmal, welches die zeitliche Veränderung einer Größe darstellt, also eine Art Ableitung nach der Zeit. Der spektrale Flux SF („Spectral/Spectrum Flux“) zeigt das Ausmaß der Veränderung im Spektrum benachbarter Fenster (vgl. [TC02, LZL03]):

$$SF\left((X_i^t)_{i\in\{0,\ldots,n-1\}}, (X_i^{t+1})_{i\in\{0,\ldots,n-1\}}\right) = \sum_{i=0}^{n-1}\left(X_i^{v^t} - X_i^{v^{t+1}}\right)^2 ,$$

wobei die beiden univariaten Wertefolgen $(X_i^t)_{i\in\{0,\ldots,n-1\}}$ und $(X_i^{t+1})_{i\in\{0,\ldots,n-1\}}$ die Amplitudenspektren darstellen.

N-Gramme Im Allgemeinen ist ein *N-Gramm* ein N-Tupel benachbarter Objekte. In dieser Arbeit handelt es sich bei diesen Objekten um zeitlich benachbarte Merkmalsausprägungsvektoren der Dimension k, welche zu einem Vektor der Dimension $N \cdot k$ zusammengefasst werden. Hat man z. B. eine Folge von Lautstärke-Ausprägungen $(ld_1, ld_2, ld_3, ld_4, ld_5, \ldots)$, dann besteht eine Folge von 3-Gramm-Ausprägungen aus

$$\left(\begin{pmatrix} ld_1 \\ ld_2 \\ ld_3 \end{pmatrix}, \begin{pmatrix} ld_2 \\ ld_3 \\ ld_4 \end{pmatrix}, \begin{pmatrix} ld_3 \\ ld_4 \\ ld_5 \end{pmatrix}, \ldots\right) .$$

Allgemein aufgeschrieben:

$$NGRAM\left((x_i)_{i\in\{l,\ldots,l+N-1\}}\right) = \begin{pmatrix} x_l^v \\ x_{l+1}^v \\ \vdots \\ x_{l+N-1}^v \end{pmatrix}$$

Die N-Gramm-Merkmale entsprechen, bis auf die zeitliche Verschiebung der Folgenelemente, den ursprünglichen Merkmalen und sollen die zeitliche Dynamik der Audiodaten abschnittsweise modellieren.

Audiomerkmale im Frequenzbereich

Alle Merkmale in diesem Abschnitt basieren auf einer vorherigen Transformation des Zeitsignals, also der einzelnen Fenster der Audiodatenfolge, in den Frequenzbereich mittels einer *Diskreten Fourier Transformation (DFT)*, welcher im Normalfall noch eine Anwendung einer Fensterfunktion, wie der Hamming- oder Hannfunktion, vorausgeht, um Kanteneffekte zu vermeiden. Das Ergebnis dieser gefensterten Transformation wird auch als „Kurzzeitspektrum“ bezeichnet.
Die Gruppe der klangorientierten (*sound*) Merkmale versucht auf verschiedene Weisen, das Spektrum „möglichst gut“ auf relativ wenige Werte abzubilden, und dies so, dass relevante Änderungen im Spektrum sich auch im Bild niederschlagen bzw. sehr unterschiedliche Spektren auf möglichst unterschiedliche Bilder abgebildet werden.

Mel Frequency Cepstral Coefficients Die *Mel Frequency Cepstral Coefficients (MFCC)* kommen aus dem Bereich der automatischen Spracherkennung, haben sich aber auch bei der Anwendung auf musikalische Audiodaten als nützlich erwiesen [Log00a]. Sie dienen der kompakten Repräsentation des Spektrums unter Berücksichtigung der menschlichen Wahrnehmung. Die Berechnung der MFCC aus einer univariaten Teilfolge $(x_i)_{i\in\{l,\ldots,m\}}$ geschieht folgendermaßen:

1. **DFT mit Hannfunktion**:

$$X_k^v = \sum_{j=l}^{m} x_j^v e^{-2\pi ijk/N} \frac{1}{2}\left[1 - cos\left(\frac{2\pi j}{N}\right)\right]$$

   für $k = 0,\ldots,N-1$ mit $N = m - l + 1$. Ergebnis ist das Spektrum (Abschnitt A.3) $(X_i)_{i\in\{0,\ldots,N-1\}}$, welches bei reellwertigen Funktionen symmetrisch ist. Für die weitere Verarbeitung werden nur die Beträge der ersten $\frac{N}{2}$ komplexen Koeffizienten benötigt.

2. **Bandpassfilterung mit Mel-Filterbank**:

$$\hat{X}_k^v = ln\left(\sum_{j=0}^{\frac{N}{2}-1} X_j^v H^k(j)\right)$$

   für $k = 0,\ldots,N_{mf}-1$. $H^k(j)$ ist das Spektrum der Impulsantwort des k-ten Mel-Filter, mit welchen die Abbildung auf die Mel-Frequenzskala geschieht („warping“). N_{mf} ist die Anzahl der verwendeten Filter, heutzutage häufig $N_{mf} = 40$.

Anschaulich handelt es sich bei den Mel-Filtern (im einfachsten Fall) um Dreiecksfilter, die im unteren Frequenzbereich schmalbandiger sind und näher beieinander liegen und mit zunehmender Mittenfrequenz breitbandiger werden und weiter voneinander entfernt liegen [FGZ01]. Die Motivation dazu ist, dass Menschen den unteren Frequenzbereich differenzierter wahrnehmen, als den oberen. Darüber hinaus wird die Filterausgabe noch logarithmiert, was dem annähernd logarithmischen menschlichen Lautstärkeempfinden Rechnung trägt.

3. **Cosinustransformation**: Der letzte Schritt ist die Berechnung der *Cepstral Coefficients*[1] $MFCC_k$:

$$MFCC_k = \sum_{j=0}^{N_{mf}-1} \hat{X}_j^v \; cos\left(k\left(j-\frac{1}{2}\right)\frac{\pi}{N_{mf}}\right)$$

für $k = 0, \ldots, N_{mfcc} - 1$. N_{mfcc} ist die Anzahl der berechneten Koeffizienten, häufig ist $N_{mfcc} = 13$. Dieser Schritt dient der Dekorrelation.

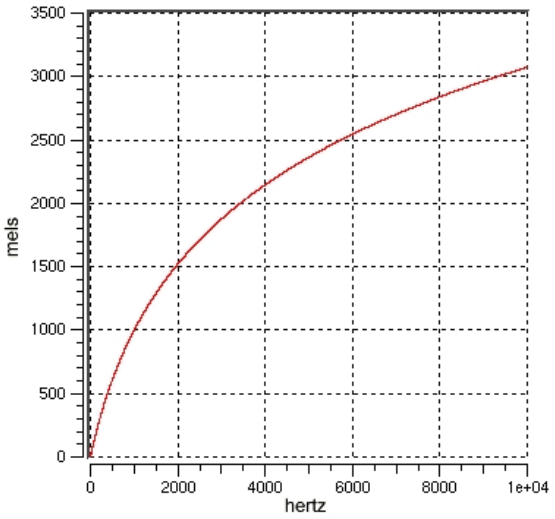


Abbildung 2.1.: $Mel(f) = 1127,01 \cdot ln\left(1+\frac{f}{700}\right)$, der Zusammenhang zwischen Hertz- und Melfrequenzskala

Spectral Centroid Hierbei wird der Schwerpunkt des Spektrums berechnet. Dieses Merkmal wird auch als „(Klang-)Helligkeit“ bezeichnet (vgl. [LZL03, TC02]):

$$SC\left((X_i)_{i\in\{l,\ldots,m\}}\right) = \frac{\sum_{i=l}^{m}(X_i^v)^2 X_i^d}{\sum_{i=l}^{m}(X_i^v)^2}$$

Bandbreite Die Bandbreite ist ein Maß für den Frequenzumfang eines Signals und wird mit Hilfe des spektralen Schwerpunktes berechnet [LZL03]:

$$BW\left((X_i)_{i\in\{l,\ldots,m\}}\right) = \sqrt{\frac{\sum_{i=l}^{m}(X_i^v)^2\left(X_i^d - SC\left((X_i)_{i\in\{l,\ldots,m\}}\right)\right)^2}{\sum_{i=l}^{m}(X_i^v)^2}}$$

[1] „cepstrum“ ist ein Kunstwort und wird für die Bezeichnung des Spektrums eines Spektrums verwendet

Spectral Rolloff Der *Spectral Rolloff* ist definiert als $SR\left((X_i)_{i\in\{l,\ldots,m\}}\right) = X^d_{r^*}$, wobei r^* das kleinste $l \leq r < m$ ist, für das gilt:

$$\sum_{i=l}^{r} X_i^v \geq 0.85 \sum_{i=l}^{m} X_i^v$$

Das heißt, gesucht ist die Frequenz, unterhalb derer bei einer Summierung der Koeffizienten des Spektrums 85% der Summe aller Beträge liegen [TC02].

Linear Prediction Cepstral Coefficients Die *Linear Prediction Cepstral Coefficients (LPCC)* verwenden die Methode der linearen Prädiktion und sind ebenfalls ein Weg, das Spektrum durch die Angabe weniger Koeffizienten zu beschreiben.

Die Gruppe der tonhöhenorientierten (*pitch*) Merkmale versucht, das Spektrum aus musikalischer Sicht, d. h. Töne oder Tonstrukturen zu beschreiben:

Fundamentalfrequenz Harmonische Klänge, wie sie von Melodieinstrumenten erzeugt werden, bestehen im Wesentlichen aus einem Grundton mit der Fundamentalfrequenz f_0 und einer Reihe von sogenannten Obertönen, deren Frequenzen ganzzahlige Vielfache von f_0 sind. Die Fundamentalfrequenz ist das, was wir Menschen im Verlauf eines Liedes als Melodie wahrnehmen, und wäre deshalb ein gutes Merkmal zur Strukturierung von musikalischen Audiodaten. Leider ist es, vor allem bei polyphoner Musik, schwierig, f_0 aus den Kurzzeitspektren sicher zu bestimmen, weil sie, bedingt durch die Klangcharakteristik des Instrumentes oder durch Überlagerungen, nicht unbedingt zu den dominanten Frequenzen gehören muß [Got04].

Chroma Das Chroma-Merkmal ist ein 12-dimensionaler Vektor, dessen Komponenten jeweils den Anteil einer der zwölf Tonklassen (A...G#), aufsummiert über mehrere Oktaven, an einem Spektrum repräsentieren. Dazu wird das Spektrum mittels Bandpässen, deren Mittenfrequenzen jeweils eine Oktave, d. h. eine Frequenzverdopplung, auseinander liegen, gefiltert und über das Ergebnis integriert. Dieses Merkmal hat sich als dazu geeignet erwiesen, Akkorde zu identifizieren [Got03].

Audiomerkmale im Phasenraum

Eine weitere Möglichkeit der Darstellung von Audiodaten bietet die Transformation in den Phasenraum. Dabei werden die Audiodaten als Beobachtungen eines dynamischen Systems mit unbekannten Zustandsvariablen aufgefasst und eine sogenannte „Zustandsraumrekonstruktion“ durchgeführt [Mie03].

Definition 2.4. (Transformation in den Phasenraum) Die Transformation einer univariaten Wertefolge $(x_i)_{i\in\{0,\ldots,n-1\}}$ in den Phasenraum geschieht durch Bildung der Vektoren

$$p_i = (x_i, x_{i+d}, x_{i+2d}, \ldots, x_{i+(m-1)d}) \ ,$$

wobei d die Verzögerung und m die Dimension des Phasenraumes darstellen.

$$P_{d,m} = \{p_i | i = 0, \ldots, n-1-(m-1)d\}$$

ist die Transformierte von $(x_i)_{i\in\{0,\ldots,n-1\}}$ im Phasenraum.

Zirkularität Es hat sich gezeigt, dass der Plot der Transformierten eines klassischen Musikstücks, dessen Klangbild mehr harmonische Anteile enthält, im Phasenraum mit $d = 1, m = 2$ rundere, elliptischere Kurven beschreibt, als im Falle eines typischen Popliedes, in dem die Punkte einem etwas chaotischeren Weg folgen. Der Grad der „Rundheit“ ist definiert als der Durchschnitt aller Winkel, die die Differenzvektoren zwischen einem Element p_i und den beiden benachbarten Elementen p_{i-1}, p_{i+1} einschließen:

$$\alpha = arccos \frac{\langle s_{iv}, s_{in} \rangle}{|s_{iv}| \cdot |s_{in}|}$$

mit $s_{iv} = p_{i-1} - p_i$ und $s_{in} = p_{i+1} - p_i$.

Als zusätzliches Merkmal kann die Varianz dieser Winkel dienen.

Vektorlänge Dieses Merkmal ist die durschnittliche Länge der Differenzvektoren zwischen Elementen p_i und p_{i+1}, und auch hier kann die Varianz der Längen mit aufgenommen werden.

2.1.3. Audiomerkmale höherer Ebenen

Die folgenden Merkmale liegen auf einem höheren Abstraktionsniveau, als die im vorhergehenden Abschnitt aufgeführten grundlegenden Merkmale.

Statistische Modelle Hat man durch eine gefensterte Merkmalsextraktion als Ergebnis wieder eine Wertefolge erhalten, dann kann man durch eine weitere Fensterung eine Folge von statistischen Modellen auf dieser Wertefolge berechnen. Im einfachen Fall bedeutet dies z. B. die Berechnung von einzelnen Mittelwerten, im komplexeren Fall die Schätzung von *Gaussian Mixture Models (GMM)* für jedes Fenster.

Transkription Transkription kann als Umkehrung des Musizierens angesehen werden. Dabei wird versucht, aus musikalischen Audiodaten z. B. durch die Analyse der Kurzzeitspektren eine Partitur zu erstellen, d. h. ein Audiosignal in eine Folge von Noten und Pausen, strukturiert in Takten, zu transformieren. Die Genauigkeit der bisher bekannten Verfahren ist, gerade bei komplexen polyphonen Kompositionen, aber leider nicht sehr hoch.

Rhythmische Merkmale Diese Merkmale beschreiben z. B. das Tempo eines Musikstücks, die Taktart oder Stärke des Hauptschlages. Dabei ist eine gemeine Vorgehensweise

die Untersuchung von verschiedenen Frequenzbändern auf Periodizitäten z. B. mittels Autokorrelation [TC02].

Musterregeln Mit Hilfe einer speziellen Grammatik, der *Unification-based Temporal Grammar (UTG)*, werden in [MU04] zeitliche Muster in Datenfolgen beschrieben. Auf verschiedenen Hierarchiestufen werden dabei die zeitlichen Konzepte Dauer, Synchronizität und Reihenfolge repräsentiert. Die Beschreibung der Ereignisse und Sequenzen in einer Folge von Merkmalsausprägungen wiederum können zur Identifikation sich wiederholender Abschnitte und zur Segmentierung verwendet werden.

2.2. Segmentierung

Allgemein ausgedrückt ist das Ziel der Segmentierung, einer zunächst unstrukturierten Folge von Audiodaten Struktur zu verleihen, indem Start- bzw. Endzeitpunkte von Bereichen festgelegt werden, die in einer vorher definierten Weise eine Einheit bilden. Manchmal ist es aber auch wünschenswert oder erforderlich, nicht nur die Zeitpunkte der Segmentgrenzen zu kennen, sondern die entstandenen Segmente zusätzlich einer Klasse zuzuordnen. Dies zeigt, dass die Segmentierung auch mit dem Problem der Klassifikation verbunden ist, z. B. in Form eines zweistufigen Prozesses (zunächst Segmentbildung, dann Klassifizierung der Segmente). Sie kann aber auch als reine Klassifikationsaufgabe betrachtet werden.

Definition 2.5. (Segmentierung als reine Klassifikationsaufgabe) Sei ein Audiosignal durch eine Merkmalsausprägungsfolge $(e_i)_{i\in\{0,\dots,n-1\}}$ charakterisiert, deren Werteveketoren e_i^v die Beispiele darstellen, und Klassen $C = \{c_1, c_2, \dots, c_m\}$ mit $c_i \in \mathbb{N}$ gegeben. Klassifiziere die Beispiele e_i^v und erzeuge die Folge $(\hat{e}_i)_{i\in\{0,\dots,n-1\}}$ mit $\hat{e}_i^d = e_i^d, \hat{e}_i^v = (c_j)_i, c_j \in C$. Teilfolgen maximaler Länge von aufeinanderfolgenden $\hat{e}_i^v$ der selben Klasse c_j bilden dann die Segmente der Klasse c_j. Die Folge $(\hat{e}_i)_{i\in\{0,\dots,n-1\}}$ wird als Markierungsfolge bezeichnet.

Das weiter unten beschriebene Verfahren der SVM Segmentierung und in ähnlicher Weise auch die Constrained clustering Segmentierung funktionieren nach diesem Prinzip.
Die Ausgabe der in dieser Arbeit realisierten Verfahren ist eine Datenstruktur, welche eine geordnete Anzahl von Segmenten enthält. Diese Datenstruktur heißt ebenfalls *Segmentierung* und möchte ich kurz definieren:

Definition 2.6. (Segmentierung einer Wertefolge) Eine Segmentierung S ist eine aufsteigend sortierte Liste von adjazenten Segmenten s_i, welche jeweils einen Startzeitpunkt $start_i$, einen Endzeitpunkt end_i und eine Markierung $label_i$ mit $start_i, end_i \in \mathbb{R}$ und $label_i \in \mathbb{N} \cup \{-1\}$ besitzen. $start_i, end_i$ geben Zeitpunkte in der *displacement*-Dimension der Wertefolge an, und es gilt $end_i = start_{i+1}$.

Sind die Markierungen der Segmente nicht von Belang, dann kann eine Segmentierung auch einfach als Liste von Segmentgrenzen b_i angesehen werden.

2.2.1. Ausgewählte Beispiele und ihre Vor- und Nachteile

Nachfolgend möchte ich einige bereits existierende Verfahrensweisen zur Segmentierung von Audiodaten aus den weiter oben erwähnten Kategorien „modellbasiert“, „sequenzbasiert“ und „metrikbasiert“ skizzieren und jeweils Vor- und Nachteile nennen.

Ein „modellbasiertes“ Beispiel

In [AS01] wird ein Hidden Markov Modell (HMM) auf der Folge der Merkmalsausprägungen trainiert, um danach die wahrscheinlichste Zustandsfolge zu berechnen, wobei jeder Zustand des HMM während der Trainingsphase einem bestimmten polyphonen Klangbild, einer Klang-„Textur“, zugeordnet wird. Dies folgt aus der Überlegung, dass jede Textur, erzeugt von dem Zusammenklang bestimmter Instrumente und/oder Stimmen, eine einzigartige spektrale Hülle, beschrieben durch eine glättende Funktion, welche die Spitzen des Spektrums einschließt und dabei die Täler möglichst gut nachbildet, besitzt. Die spektrale Hülle einer Textur wird erkennbar durch die Überlagerung, also Addition mit anschließender Normierung, mehrerer aufeinander folgender Kurzzeitspektren des Audiosignals eines Klangbildes. Hier wird sie durch die Berechnung der MFCC für jedes Extraktionsfenster geschätzt. Mittels dieses Verfahrens werden Audiodaten also im Hinblick auf die in ihnen enthaltenen Klangbilder segmentiert.

1. Gefensterte Extraktion von MFCC
2. Training eines ergodischen HMM auf der Folge der Merkmalsausprägungen mit dem Baum-Welsh-Algorithmus (EM)
3. Decodierung mit dem Viterbi-Algorithmus und Ausgabe der wahrscheinlichsten Zustandssequenz

\+ **Vorteile** von „modellbasierten“ Verfahren

Möglichkeit zur Anpassung Durch die Modellbildung in der Lernphase können sich die Verfahren an unterschiedliche Audiodatenquellen anpassen

Hohe Genauigkeit Durch die Anpassung an die zu unterscheidenden Klassen können die Segmente und deren Grenzen mit hoher Genauigkeit bestimmt werden

Segmentierung und Klassifikation Die entstehenden Segmente können sofort einer Klasse zugeordnet werden

– **Nachteile** von „modellbasierten“ Verfahren

U.U. höherer Berechnungsaufwand Ist für jeden neuen Datensatz eine Lernphase notwendig, wie im oben beschriebenen Beispiel, dann ist der Berechnungsaufwand i. Allg. höher, als bei anderen Verfahren

Keine Beachtung zeitlicher Strukturen Es werden lediglich Klangbilder erkannt, aber keine wiederholt auftretenden Sequenzen, die in der Musik eine wichtige Rolle spielen

Eventuell Ausreißerbehandlung notwendig Je nach Verfahren sollten Fehlklassifikationen zur Verbesserung des Segmentierungsergebnisses erkannt und korrigiert werden

Feste Anzahl von Klassen Ist die Anzahl der Klassen nicht im Vorhinein bekannt, können wahre Segmente unerkannt bleiben (zu wenige Klassen) oder in mehrere Segmente unterteilt werden (zu viele Klassen)

Zwei „sequenzbasierte“ Beispiele

Ganz nach dem Motto „Repetition generates structure“ versuchen die Autoren von [DH], in einer Folge von Noten, welche vorher durch Transkription aus einer musikalischen Audiodatei extrahiert wurde, sich wiederholende Sequenzen zu finden. Die Suche folgt einer greedy-Strategie, wobei eine Matrix S mit Einträgen $s_{i,j} =$ *Länge einer ähnlichen Melodei beginnend an Note* i *und* j erzeugt wird. Auf der Basis dieser Matrix werden dann ähnliche Sequenzen gruppiert und diese Gruppen mit einem Namen versehen. Die Zeitintervalle der Sequenzen einiger Gruppen, welche mit einem einfachen Suchverfahren ausgewählt werden, bilden die dem Namen der Gruppe entsprechend markierten Segmente.

1. Transkription
2. Berechnung der Matrix S
3. Gruppierung ähnlicher Sequenzen und Segmentbildung

In [Got03] werden sich wiederholende Abschnitte in musikalischen Audiodaten auf graphisch recht anschauliche Weise gefunden. Nach einer gefensterten Extraktion von Chromamerkmalen wird mit Hilfe eines Distanzmaßes eine Ähnlichkeitsmatrix aller Wertefolgenelemente berechnet. In Abb. 2.2 sehen wir eine solche Matrix in Zeit/Versatz-Darstellung, wobei helle Punkte hohe, dunkle Punkte geringe Ähnlichkeit anzeigen. Die Art der Darstellung ist nun so gewählt, dass ähnliche Sequenzen von Folgenelementen als helle, waagerechte Linien sichtbar werden. Die Suche nach diesen Linien und das Zusammenfügen der Ergebnisse führt dann zur Definition von Segmenten. Da das Ziel jener Arbeit das Auffinden des Re-

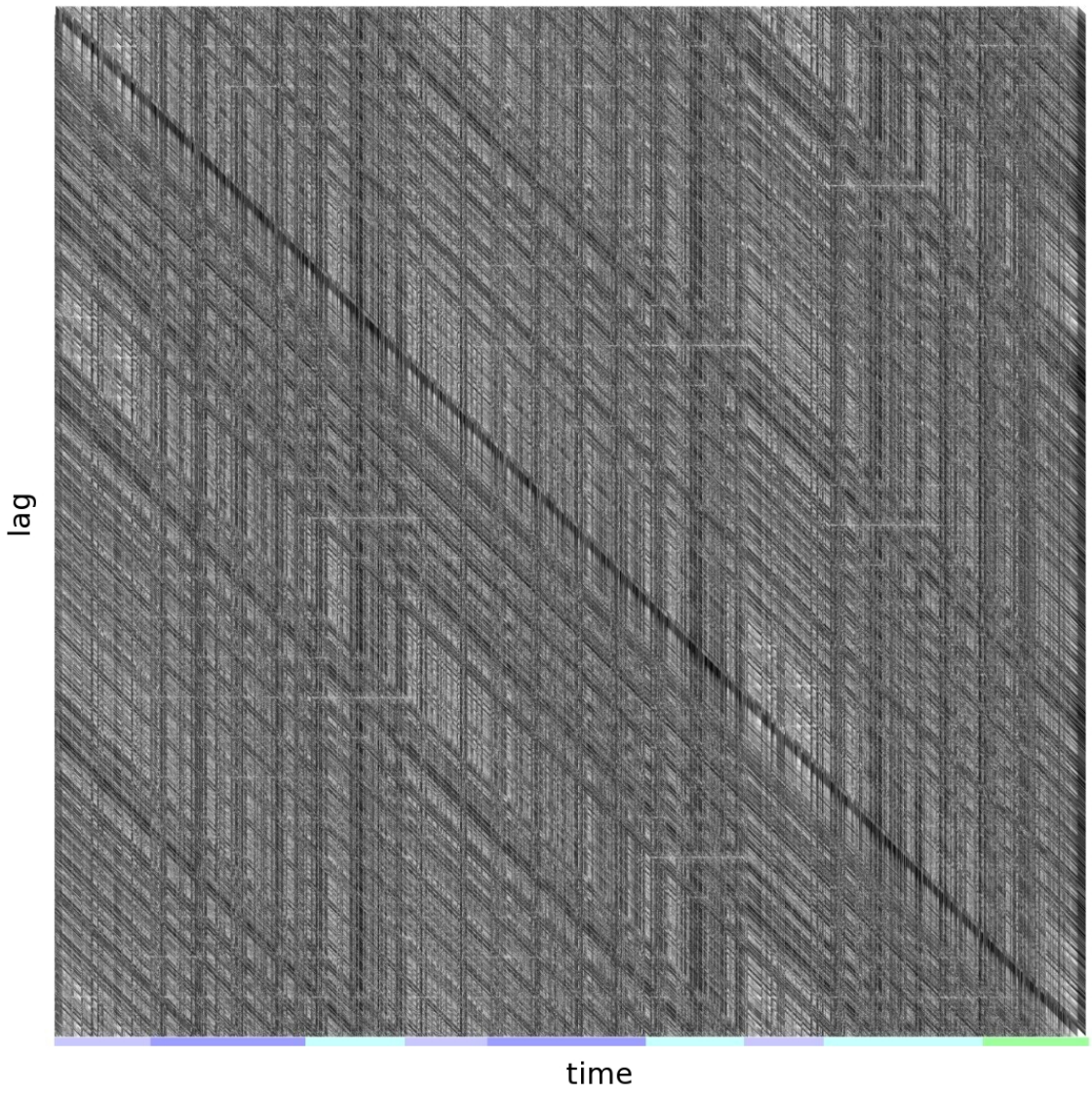


Abbildung 2.2.: Ähnlichkeitsmatrix aller Elemente einer Wertefolge von Chromamerkmalsausprägungen für „Struggling Man“ von Jimmy Cliff

frains eines Liedes ist, werden abschließend noch die Segmente bestimmt, welche diesen wahrscheinlich enthalten.

1. Gefensterte Extraktion von Chromamerkmalen
2. Berechnung einer Ähnlichkeitsmatrix
3. Segmentbestimmung
   a) Berechnung der durchschnittlichen Zeilensummen
   b) Auswahl von Zeilen mit den höchsten Werten
   c) Suche nach Abschnitten hoher Ähnlichkeit
   d) Zweiter verfeinerter Durchgang und Integration in die vorherigen Ergebnisse
4. Heuristische Auswahl des Refrains

Mittels dieser Verfahren werden Audiodaten also im Hinblick auf sich wiederholende musikalische, zeitliche Strukturen segmentiert.

+ **Vorteile** von „sequenzbasierten“ Verfahren

Relative Klassifikation Ähnliche Sequenzen können der selben Klasse zugeordnet werden, wenn auch nicht bekannt ist, welche Klassen überhaupt existieren

Kein Trainingsaufwand Es müssen keine Modelle gelernt werden

– **Nachteile** von „sequenzbasierten“ Verfahren

Mangelnde automatische Adaptivität greedy-Strategien, Heuristiken und die Verwendung von Schwellenwerten machen die Verfahren unflexibel bzw. nur manuell anpassbar

Ausrichtung auf musikalische Audiodaten Zeitliche Strukturen sind bei nicht-musikalischen Audiodaten von untergeordneter Bedeutung

Probleme bei Variationen und Improvisationen Schon kleine Variationen, wie sie in der Musik häufig vorkommen, erschweren das Auffinden von gleichen musikalischen Teilen deutlich

Ungenaue Segmentgrenzen Einige Verfahren haben aufgrund der ähnlichkeitsbasierten Suche Probleme mit der genauen Positionierung der Segmentgrenzen [CV03]

Ein „metrikbasiertes“ Beispiel

Die Evaluation dreier verschiedener Implementierungen eines Algorithmus, welcher dynamische Fensterung, also ein Fenster sich verändernder Breite, und das *Bayesian Information Criterion (BIC)* (Abschnitt A.4) benutzt, wird in [CVR05] vorgestellt. Dieser Algorithmus berechnet für ein über die Folge der Merkmalsausprägungen wanderndes Fenster die sogenannten ΔBIC-Werte, welche die Entscheidung ermöglichen, ob die im Fenster enthaltene Teilfolge durch ein oder durch zwei gaußsche Modelle besser beschrieben wird, wobei im Falle der zwei Modelle das Fenster in einen linken und einen rechten Abschnitt aufgeteilt wird und diese getrennt modelliert werden.
Für ein Fenster der Größe N wird für alle Trennungspunkte $i = 1, \ldots, N-1$ berechnet:

$$\Delta BIC_i = \frac{N}{2} \log|\Sigma| - \frac{i}{2} \log|\Sigma_1| - \frac{N-i}{2} \log|\Sigma_2| - \lambda P_{N_d(\mu,\Sigma)} \ ,$$

wobei $\Sigma, \Sigma_1, \Sigma_2$ die Maximum-Likelihood-Schätzungen der Kovarianzmatrizen für die gesamte Teilfolge, den linken und den rechten Abschnitt sind. $P_{N_d(\mu,\Sigma)}$ ist der von der Verwendung des BIC herrührende Strafterm, welcher die Anzahl der freien Parameter im Modell für die gesamte Teilfolge miteinbezieht, und λ ein Gewicht zur Einstellung der Sensitivität des Verfahrens. Ist $\Delta BIC_i > 0$ und ein lokales Maximum, so wird dieser Punkt i_max genauer untersucht und bei Bestätigung als Segmentgrenze ausgegeben, denn in diesem Fall ist die Modellierung durch zwei Modelle plausibler (je besser ein Modell auf die beobachteten Daten passt, desto kleiner ist der Betrag der Kovarianzmatrix), woraus geschlossen wird, dass ein Abschnitt mit neuem akustischen Inhalt beginnt.

1. Gefensterte Extraktion von MFCC
2. Dynamische Fensterung
   a) Für jeden Trennungspunkt i innerhalb des Fensters Berechnung der ΔBIC_i-Werte
   b) Für jedes lokale Maximum mit $\Delta BIC_i > 0$ nähere Überprüfung und gegebenenfalls Ausgabe als Segmentgrenze

+ Vorteile von „metrikbasierten“ Verfahren

Effizient Die Verfahren sind recht effizient und relativ einfach zu implementieren

Kein Trainingsaufwand Es müssen keine Modelle gelernt werden

– **Nachteile** von „metrikbasierten“ Verfahren

Keine Beachtung zeitlicher Strukturen Es werden lediglich Differenzen/Distanzen zwischen Kenngrößen/lokalen Modellen berechnet, keine wiederholt auftretenden Sequenzen gefunden

Wahl des Distanzmaßes Das Segmentierungsergebnis ist von der Wahl des Distanzmaßes abhängig

Anpassung von Parametern Bei vielen Verfahren ist eine Anpassung von Parametern notwendig

Keine Klassifikation Es werden lediglich die Übergänge zwischen Segmenten erkannt, aber keine Informationen bezüglich Klassen oder der Beziehung der Segmente untereinander geliefert, was eventuell einen Nachverarbeitungsschritt notwendig macht

Probleme mit kurzen Segmenten Innerhalb von kurzen Segmenten ist keine zuverlässige Bestimmung von statistischen Kenngrößen möglich, auf welche sich viele Verfahren stützen

3. Realisierte Segmentierungsverfahren

Bevor nun in diesem Kapitel die realisierten Segmentierungsverfahren vorgestellt werden, möchte ich kurz auf das Konzept der *Operatorbäume* eingehen, da sowohl die Merkmalsextraktion, welche allen Verfahren zu Grunde liegt, als auch die automatische Segmentierung auf diese Weise implementiert wurden.

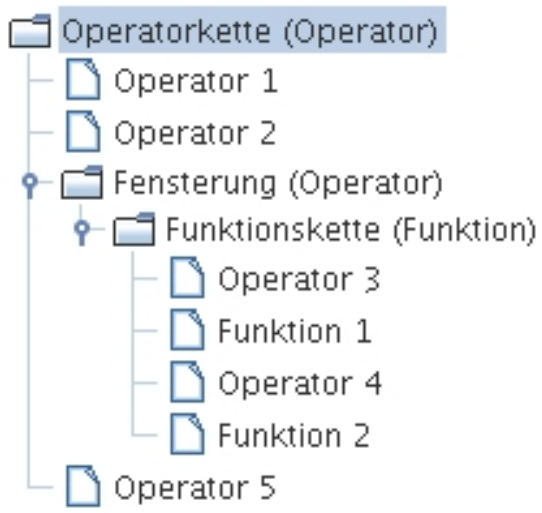


Abbildung 3.1.: Ein abstrakter Operatorbaum

Operatorbäume sind ein allgemeines Konzept zur Datenverarbeitung, welches ähnlich zu den in [MM05] definierten Methodenbäumen, oder *Method trees*, ist und schon in YALE [MWK+06] zur Erstellung von Lernexperimenten verwendet wird. Auch in *Segfried* kommen Operatorbäume zur Anwendung, einer Software, die in dieser Arbeit zur Durchführung der Experimente eingesetzt wurde (Kapitel 4). Dabei werden auf eine Wertefolge, der Struktur eines Baumes folgend, eine Reihe von Einzeloperationen angewendet. Diese Einzeloperationen, die die Knoten eines Operatorbaumes darstellen, unterscheidet man in Operatoren und Funktionen:

Operatoren sind Knoten, die als Eingabe eine Wertefolge $(x_i)_{i\in\{0,\dots,n-1\}}$ erwarten, eine Transformation durchführen und als Ausgabe die Ergebniswertefolge $(\hat{x}_i)_{i\in\{0,\dots,m-1\}}$ zurückgeben.

Funktionen sind Knoten, die als Eingabe eine Wertefolge $(x_i)_{i\in\{0,\dots,n-1\}}$ erwarten, in Abhängigkeit von dieser einen Funktionswert $\mathcal{F}\left((x_i)_{i\in\{0,\dots,n-1\}}\right) \in \mathbb{C}^k$ berechnen und als Ausgabe den Ergebnisvektor zurückgeben.

Im Unterschied zu Methodenbäumen ist bei den Operatorbäumen eine Operatorkette, ein Operator, dessen Kinder eine Kette von Operatoren bilden, und keine *allgemeine Fensterung* die Wurzel. Eine allgemeine Fensterung kann aber durch Verwendung einer *Funktionskette*, einer Funktion, deren Kinder Operatoren und Funktionen sein können und welche die

Resultate der Funktionen kombiniert als Vektor zurückliefert, realisiert werden (Abb. 3.1). Ausgehend von dem Wurzelknoten, der also immer ein Operator ist, und diesem einfachen Grundkonzept, ist es anschaulich und recht intuitiv möglich, Operatorbäume zu erstellen, welche die unterschiedlichsten Transformationen, wie z. B. die Extraktion von Audiomerkmalen, auf Wertefolgen durchführen.

3.1. Ein einfaches, „metrikbasiertes“ Verfahren

Wie in [PE04] beschrieben, wird bei diesem Verfahren ein Fenster über die Folge der Merkmalsausprägungen geschoben und jeweils mittels einer Kriteriumsfunktion, die auch als *Fishers Kriterium* bezeichnet wird, ein Funktionswert berechnet. Dazu teilt man das Fenster bzw. die Teilfolge in zwei Hälften und bestimmt für jede Dimension und Hälfte die Mittelwerte und Varianzen. Fishers Kriterium berechnet sich nun für eine Dimension als Differenz der Mittelwerte normiert auf die Summe der Varianzen und liefert ein Maß für die Unterscheidbarkeit der Kenngrößen.

Abbildung 3.2.: Der Operatorbaum für die „metrikbasierte“ Segmentierung

Definition 3.1. (Fishers Kriterium) Für eine Teilfolge $(x_j)_{j\in\{l,\ldots,m\}}$ mit $x_j = (x_j^d, (v_0,\ldots,v_{k-1})_j)$ und ein Fenster der Größe $w = m-l+1$ berechnet sich Fishers Kriterium für Dimension i zu

$$fc_i = \frac{(\bar{v}_{i,1} - \bar{v}_{i,2})^2}{\sigma_{i,1}^2 + \sigma_{i,2}^2}$$

mit

$$\bar{v}_{i,1} = \frac{2}{w}\sum_{j=l}^{l+\frac{w}{2}-1}(v_i)_j \qquad \bar{v}_{i,2} = \frac{2}{w}\sum_{j=l+\frac{w}{2}}^{m}(v_i)_j$$

$$\sigma_{i,1}^2 = \frac{2}{w}\sum_{j=l}^{l+\frac{w}{2}-1}((v_i)_j - \bar{v}_{i,1})^2 \quad \sigma_{i,2}^2 = \frac{2}{w}\sum_{j=l+\frac{w}{2}}^{m}((v_i)_j - \bar{v}_{i,2})^2 \ .$$

Dieser Wert ist umso größer, je unterscheidbarer die Kenngrößen der beiden Fensterhälften sind. Die Summe dieser Werte über alle Dimensionen ist dann der Funktionswert für das gesamte Fenster:

$$FC\left((x_j)_{j\in\{l,\ldots,m\}}\right) = fc_0 + fc_1 + \ldots + fc_{k-1}$$

Das Ergebnis dieser gefensterten Berechnung ist eine Wertefolge, deren lokale Maxima Stellen größter lokaler Unterscheidbarkeit der Kenngrößen der Teilfenster links und rechts davon

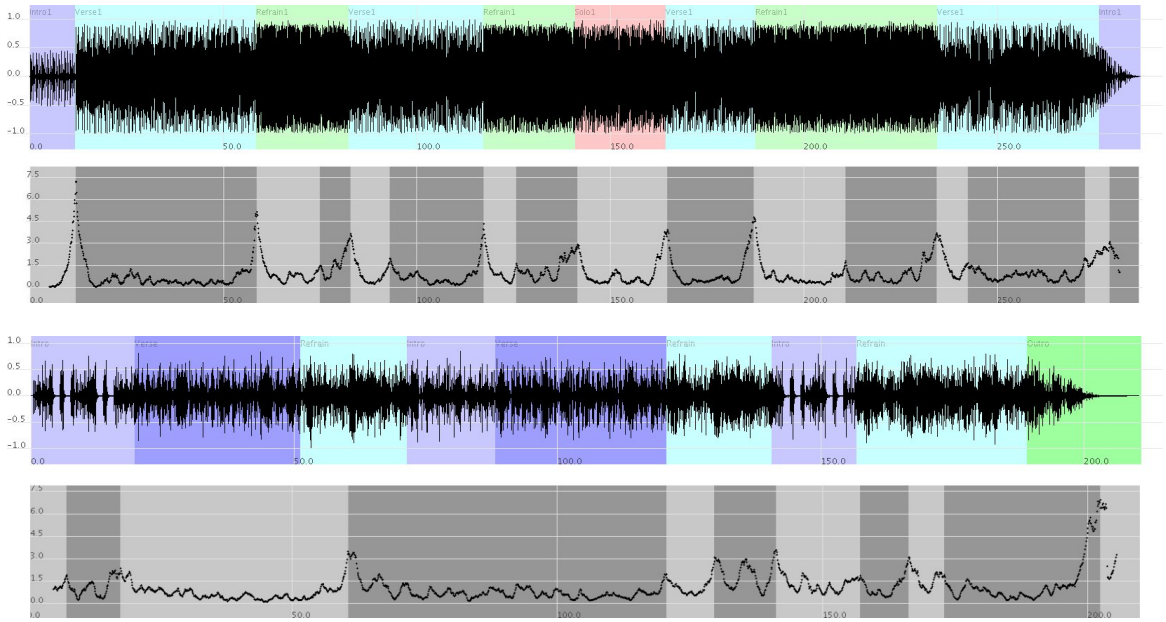


Abbildung 3.3.: Die „metrikbasierte“ Segmentierung der Lieder „Believe“ von King's X (oben) und „Struggling Man“ von Jimmy Cliff (unten) mit der Fenstergröße $w = 10s$. Zu sehen sind jeweils das Audiomaterial samt manueller Segmentierung und die Funktion FC mit dem Segmentierungsergebnis. Als Merkmale dienten MFCC.

anzeigen. Die Fenstergröße w ist dabei ein kritischer Parameter und bestimmt, unter welcher Skala das Audiomaterial betrachtet wird: Ein kleines Fenster offenbart feinere Strukturen kürzerer Dauer, ein großes Fenster gröbere Strukturen von längerer Dauer.
Zur Segmentierung werden nun die Spitzen der Wertefolge, also Maximalstellen mit abfallenden Flanken, bestimmt und als Grenzen erkannt. Die heuristische Suche nach den Funktionsspitzen basiert auf dem Divide-and-Conquer-Verfahren aus [Mie04] und wird durch die maximale Anzahl der zu findenden Spitzen, den Mindestabstand zwischen zwei Spitzen und Parametern, die die Bestimmung der Breite einer Spitze beeinflussen, gesteuert. Dabei wird für eine univariate Teilfolge $(x_i)_{i\in\{l,\dots,m\}}$ zunächst der Index des Wertemaximums i_{max} als Spitze und die Indizes des linken und rechten Spitzenendes i_{lend}, i_{rend} bestimmt. Danach wird diese Prozedur auf den Teilfolgen $(x_i)_{i\in\{l,\dots,i_{lend}\}}$ und $(x_i)_{i\in\{i_{rend},\dots,m\}}$ wiederholt.
Die Berechnung der Spitzenenden geschieht von i_{max} ausgehend unter Beachtung des Mindestabstands zwischen zwei Spitzen und durch Vergleich der Werte mit einem Schwellenwert, der für aufeinanderfolgende Werte nur begrenzt oft überschritten werden darf. Die Bestimmung des linken Endes einer Spitze geschieht nach folgendem Algorithmus, die des rechten Endes in analoger Weise:

INPUT: value series $(x_i)_{i\in\{l,\dots,m\}}$, index of peak i_{max}, minimum peak distance d, tolerance t, sloppy value s, global value series average a_{glob}
OUTPUT: index of left peak end i_{lend}
INITIALIZATION: current index $j = i_{max}$, value sum $sum = 0$, value count $vc = 0$, sloppy count $sc = 0$
$GetLeftEndOfPeak((x_i)_{i\in\{l,\dots,m\}}, i_{max}, d, t, s, a_{glob})$
(1) WHILE $j > l$ AND $sc \leq s$
(2) $lastval = x_j^v$
(3) $sum = sum + x_j^v$
(4) $j = j - 1$
(5) $vc = vc + 1$
(6) IF $(vc < d)$ OR $(x_j^v \leq lastval)$ OR $(x_j^v < a_{glob})$ OR $(x_j^v < t\frac{sum}{vc})$
(7) $sc = 0$
(8) ELSE
(9) $sc = sc + 1$
(10) RETURN j

Eine Klassifikation der Segmente erfolgt bei diesem Verfahren nicht.

1. Gefensterte Audiomerkmalsextraktion $\rightarrow \hat{x} = (\hat{x}_j)_{j\in\{0,\dots,n-1\}}$
2. Gefensterte Berechnung der Funktion FC mittels Fishers Kriterium
3. Heuristische Bestimmung der Segmentgrenzen an Stellen lokaler Maxima

3.2. SVM Segmentierung

Das Verfahren der SVM Segmentierung fußt, wie jenes in [LZL03], auf der Klassifikation der Wertevektoren einer Merkmalsausprägungsfolge. Dabei wird zunächst mittels der durch manuelle partielle Segmentierung markierten Teilfolgen eine Beispielmenge erzeugt, wobei die Merkmalsausprägungsvektoren die Markierung bzw. das *label* der Teilfolge erhalten, zu welcher sie gehören, und anschließend eine Support Vector Machine (SVM) (siehe Abschnitt 3.2.1) darauf trainiert. Bei dieser Arbeit wurde der LibSVMLearner-Operator der

Abbildung 3.4.: Der Operatorbaum für die SVM Segmentierung

Lernumgebung YALE [MWK+06] mit einer radialen Basisfunktion (rbf) als Kernel und den Parametern $C = 2, \gamma = \frac{1}{k}$ verwendet, wobei k die Anzahl der Attribute bzw. die Dimension der Merkmalsausprägungsvektoren ist.
Nachdem das Modell gelernt ist, wird eine Beispielmenge aus allen Wertevektoren der Merkmalsausprägungsfolge generiert, welche dann klassifiziert bzw. markiert werden. Aus den klassifizierten Beispielen wird dann wiederum eine Folge der Markierungen gewonnen, die denselben Zeitbezug wie die Merkmalsausprägungsfolge hat. Da diese Markierungsfolge sich aufgrund von Fehlklassifikationen jedoch noch nicht direkt zur Segmentbildung eignet, wird als Nachverarbeitungsschritt eine auf dynamischer Programmierung basierende Ausreißerbehandlung durchgeführt (Abschnitt 3.2.2). Nachdem die Markierungsfolge auf diese Weise homogenisiert wurde, werden die Zeitabschnitte maximaler Teilfolgen mit konstanter Markierung schlussendlich als Segmente aufgefasst.

1. Manuelle Teilsegmentierung
2. Gefensterte Audiomerkmalsextraktion $\rightarrow \hat{x} = (\hat{x}_j)_{j \in \{0,\ldots,n-1\}}$
3. Generierung einer Trainingsmenge aus markierten Teilfolgen $(\hat{x}_j)_{j \in \{l,\ldots,m\}}$ und Training einer SVM
4. Generierung einer Testmenge aus gesamt $\hat{x}$ und Klassifikation der Merkmalsausprägungsvektoren
5. Erzeugung einer Markierungsfolge $l = (l_j)_{j \in \{0,\ldots,n-1\}}$
6. Ausreißerbehandlung mittels dynamischer Programmierung
7. Zeitabschnitte maximaler Teilfolgen mit konstanter Markierung bilden die Segmente

3.2.1. Support Vector Machines

Support Vector Machines (SVM) sind überwachte Lernmethoden zur linearen Klassifikation, die für eine Menge von markierten Trainingsbeispielen $E = \{(e_1, c_1), (e_2, c_2), \ldots, (e_n, c_n)\}$ mit Beispielen $e_i \in \mathbb{R}^k$ und Klassenmarkierungen $c_i \in \{-1, 1\}$ versuchen, eine Hyperebene zu berechnen, welche die Beispiele beider Klassen im $\mathbb{R}^k$ optimal trennt. Das Optimalitätskriterium ist dabei nicht nur die Anzahl (bzw. das Gewicht) der Fehlklassifikationen, welche minimiert werden soll, sondern zusätzlich noch der Abstand der Trainingsbeispiele beider Seiten zur trennenden Hyperebene, welcher maximiert werden soll [CV95, Bur98].
Hat man im Idealfall von linear separierbaren Beispielen eine trennende Hyperebene S mit $\langle w, e \rangle + b = 0$ und Normalenvektor w und sind d_+, d_- jeweils der senkrechte Abstand des nächsten positiven bzw. negativen Beispiels zu S, dann möchte man den Rand (*margin*) $d_+ + d_-$ von S maximieren, und es soll $d_+ = d_-$ gelten. Dieses Problem ist äquivalent

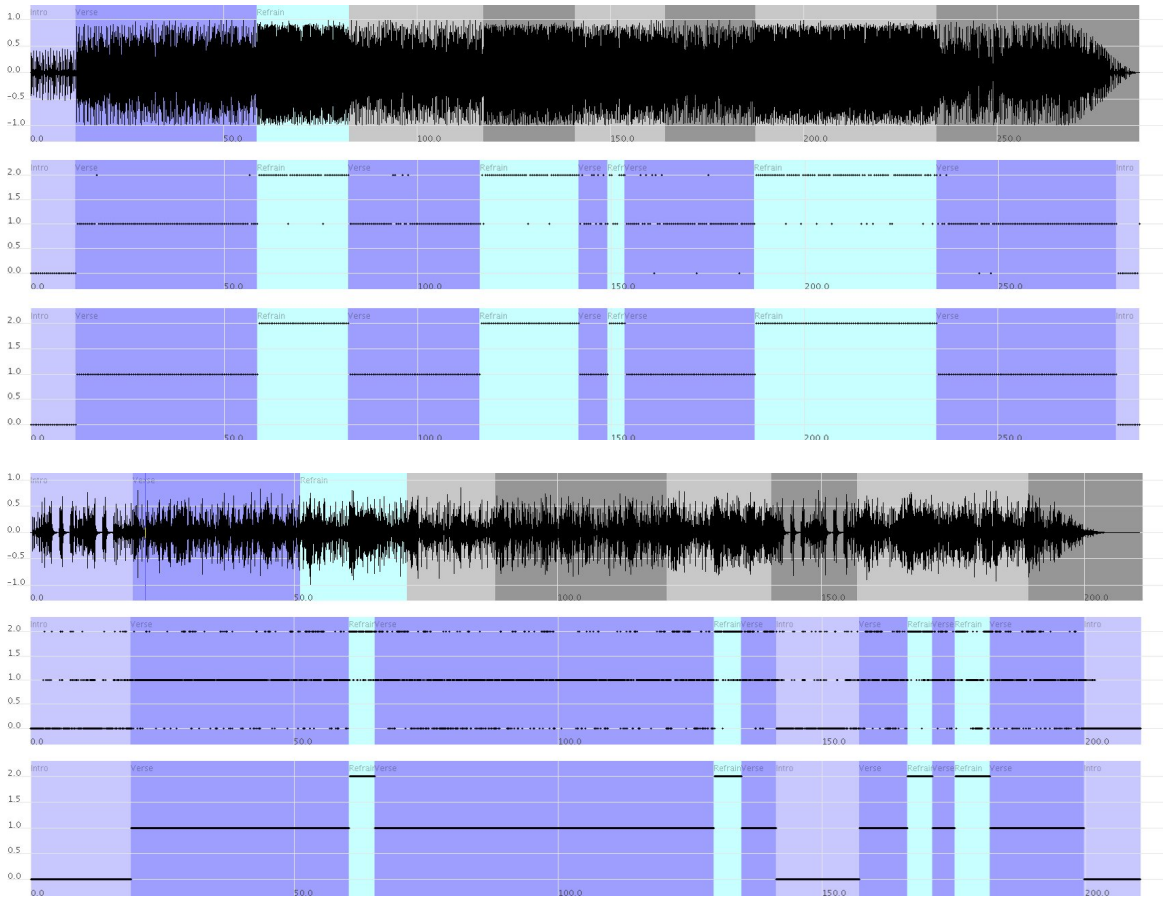


Abbildung 3.5.: Die SVM Segmentierung der Lieder „Believe“ von King’s X (oben) und „Struggling Man“ von Jimmy Cliff (unten). Zu sehen sind jeweils das Audiomaterial samt manueller Teilsegmentierung (farbig) sowie die Markierungsfolge vor und nach Anwendung der Ausreißerbehandlung zusammen mit dem Segmentierungsergebnis. Die Parameter der Ausreißerbehandlung waren $minlength = 4s, maxlength = 30s$. Als Merkmale dienten MFCC.

zu der Aufgabenstellung, zwei parallele Hyperebenen $\langle w, e\rangle + b = 1$ und $\langle w, e\rangle + b = -1$ mit Abstand $2/||w||_2$ so zwischen den Beispielen zu platzieren, dass die eine mindestens ein positives, die andere mindestens ein negatives Beispiel berührt, der Raum zwischen ihnen frei von Beispielen ist und der Abstand maximiert wird [Bur98]. Die Beispiele, die auf diesen Hyperebenen zu liegen kommen und deren Position im Raum bestimmen, nennt man *Stützvektoren*, und es gilt

$$\frac{1}{2}||w||_2^2$$

unter den Nebenbedingungen

$$c_i(\langle w, e_i\rangle + b) \geq 1, \quad 1 \leq i \leq n$$

zu minimieren.
Sind die Beispiele nicht linear separierbar, dann kann man die Nebenbedingungen mit Hilfe von sogenannten *Schlupfvariablen* $\xi_i \geq 0$ aufweichen. Ein positiver Wert der Schlupfvariablen ξ_i zeigt dann eine Verletzung der Nebenbedingung für das Beispiel i, also eine Fehlklassifikation, an. Das Optimierungsproblem stellt sich dann folgendermaßen dar: Minimiere

$$\frac{1}{2}||w||_2^2 + C\sum_{i=1}^{n}\xi_i$$

unter den Nebenbedingungen

$$c_i(\langle w, e_i\rangle + b) \geq 1 - \xi_i, \quad 1 \leq i \leq n\ .$$

Der zu minimierende Wert wird also mit dem Gewicht der Fehlklassifikationen, welches mit dem positiven konstanten Faktor C eingestellt werden kann, bestraft.
Mit Hilfe der *Lagrange-Multiplikatoren* und der *Karush-Kuhn-Tucker-Bedingungen* kann dieses konvexe Optimierungsproblem in seine **duale Form** überführt werden: Maximiere für Variablen $\alpha_i,\ 1 \leq i \leq n$,

$$\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j c_i c_j \langle e_i, e_j\rangle$$

unter den Nebenbedingungen

$$0 \leq \alpha_i \leq C, \quad \textstyle\sum_{i=1}^{n}\alpha_i c_i = 0\ .$$

Diese enthält keine Schlupfvariablen mehr und hat vor allem den Vorteil, dass der Test der Nebenbedingungen deutlich effizienter durchgeführt werden kann. Die Ergebnisse des nun in dem zu maximierenden Ausdruck vorkommenden Skalarprodukts $\langle e_i, e_j\rangle$ zweier Beispiele kann während der Optimierung zwischengespeichert und wiederverwendet werden, da die Beispiele ja konstant sind, was den Aufwand der Berechnung klein hält. Die Herleitung

dieser Form offenbart zudem, dass der Normalenvektor w auch als Linearkombination der Trainingsbeispiele geschrieben werden kann:

$$w = \sum_{i=1}^{n} \alpha_i c_i e_i$$

Durch eine Abbildung der Beispiele in einen höherdimensionalen Raum mit Hilfe einer Abbildung ϕ und entsprechende Ersetzung des Skalarprodukts $\langle e_i, e_j \rangle$ durch $\langle \phi(e_i), \phi(e_j) \rangle$, können im Ursprungsraum auch nichtlineare Klassifikationsgrenzen realisiert werden. Die Kosten der Berechnung von $\langle \phi(e_i), \phi(e_j) \rangle$ lassen sich durch Verwendung einer positiv definiten kernel-Funktion $k(e_i, e_j) = \langle \phi(e_i), \phi(e_j) \rangle$ dabei stark reduzieren. In dieser Arbeit wurde eine radiale Basisfunktion mit $k(e_i, e_j) = \exp(-\gamma ||e_i - e_j||_2^2)$ verwendet.
Zur Klassifikation eines Beispiels e ergibt sich die Klassifikationsfunktion dann zu

$$f(e) = \operatorname{sgn}(\langle w, e \rangle + b) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i c_i k(e_i, e) \ + b \right) \ .$$

Zur Unterscheidung von mehr als zwei Klassen mit Hilfe von SVM gibt es verschiedene Ansätze. Hier kam die sogenannte „1-gegen-1“-Strategie zum Einsatz. Dabei wird zunächst für jedes mögliche Klassenpaar (c_i, c_j) aus M Klassen eine SVM trainiert. Ein Beispiel wird dann von jeder der insgesamt $M(M-1)/2$ SVM klassifiziert und am Ende der Klasse zugeordnet, für welche die meisten SVM „gestimmt“ haben.

3.2.2. Ausreißerbehandlung

Die durch Klassifikation der Merkmalsausprägungsvektoren erhaltene Markierungsfolge $l = (l_j)_{j \in \{0,\ldots,n-1\}}$ weist aufgrund von Fehlklassifikationen mehr oder weniger starke Inhomogenitäten auf, weshalb sie als Grundlage zur Berechnung einer vernünftigen Segmentierung ungeeignet ist. Zur Verbesserung des Segmentierungsergebnisses wird deshalb durch die Anwendung der Idee aus [Bia03] die Markierungsfolge bezüglich eines Homogenitätsmaßes Hom mit Hilfe dynamischer Programmierung global optimiert. Das Verfahren berechnet zunächst eine optimale Segmentierung und verwendet die folgenden Datenfelder:

G[i]: Wert einer optimalen Lösung für die Teilfolge $(l_j)_{j \in \{0,\ldots,i-1\}}$

z[i]: Länge des letzten Segmentes der optimalen Lösung für i

q[i]: Anzahl der Segmente der optimalen Lösung für i

In [Bia03] basiert die Berechnung der Homogenität auf der Entropie. Aus Effizienzgründen wird hier jedoch ein einfacheres Maß verwendet, welches sich als ebenso brauchbar erwiesen hat. Seien $L_1, \ldots, L_p$ die in l vorkommenden verschiedenen Markierungen und $\#L_1, \ldots, \#L_p$ jeweils die Anzahl der Vorkommen in $(l_j)_{j \in \{k,\ldots,m\}}$, dann ist

$$Hom((l_j)_{j \in \{k,\ldots,m\}}) = 2 \max\{\#L_1, \ldots, \#L_p\} - (m - k + 1) \ .$$

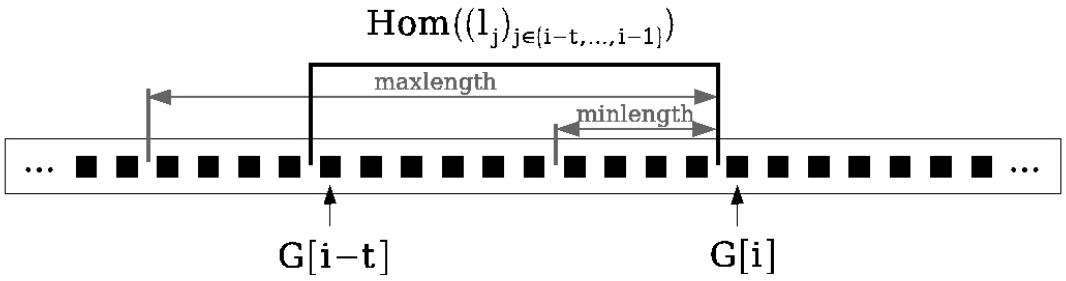


Abbildung 3.6.: Das Datenfeld G und die Berechnung des Wertes einer optimalen Lösung $G[i]$ für die Markierungsteilfolge $(l_j)_{j\in\{0,\ldots,i-1\}}$

Bei einem Durchlauf von $i = 1, \ldots, n-1$ werden nun folgende Berechnungen durchgeführt:

$$G[i] = \max_{t=minlength\ldots maxlength} \{G[i-t] + Hom((l_j)_{j\in\{i-t,\ldots,i-1\}})\}$$
$$z[i] = \operatorname*{argmax}_{t=minlength\ldots maxlength} \{G[i-t] + Hom((l_j)_{j\in\{i-t,\ldots,i-1\}})\}$$
$$q[i] = q[i - z[i]] + 1$$

Die homogenisierte Markierungsfolge erhält man abschliessend dadurch, dass in l für jedes berechnete Segment die darin am häufigsten vorkommende Markierung bestimmt und alle Folgenglieder auf diese Markierung gesetzt werden. Mit Hilfe der Parameter $minlength, maxlength$ kann der Rechenaufwand begrenzt werden. Während $minlength$ die minimale Länge der Segmente definitiv festlegt, beschränkt $maxlength$ lediglich die Suchweite, jedoch nicht die maximale Länge der Segmente.

3.3. Constrained clustering Segmentierung

Mit diesem Verfahren wird erstmalig versucht, Strukturen in Audiodaten mittels eines halbüberwachten Clustering-Verfahrens zu entdecken. Dabei wird die Menge der Merkmalsausprägungsvektoren unter Verwendung einer Constrained clustering Methode partitioniert, wobei die Vektoren einer Partition sich in einer vorher definierten Weise möglichst ähnlich sein sollen. Die Ähnlichkeitsbeziehungen werden durch die Angabe von Nebenbedingungen oder *constraints* zum Ausdruck gebracht. Es können

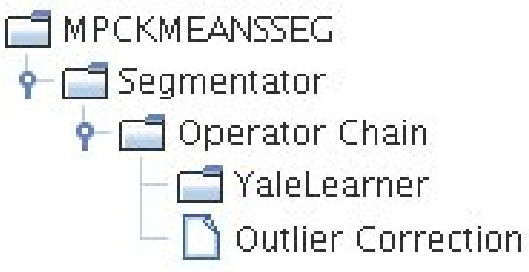


Abbildung 3.7.: Der Operatorbaum für die Constrained clustering Segmentierung

sogenannte Must-Link-constraints und Cannot-Link-constraints spezifiziert werden, erstere für Vektorpaare, welche zur selben Partition gehören sollen, letztere für Paare, welche zu verschiedenen Partitionen gehören sollen. Für die Segmentierung von Audiodaten stellt die Verwendung von halbüberwachten Clustering-Verfahren eine Neuerung dar.
Zunächst wird aus allen Merkmalsausprägungsvektoren eine Beispielmenge generiert, die durch die vorherige manuelle partielle Segmentierung teilweise markierte Beispiele enthält. Mittels dieser Beispiele wird die Menge der Nebenbedingungen erzeugt: Aus einigen Beispielpaaren mit derselben Markierung werden Must-Link-constraints, aus anderen Beispielpaaren mit unterschiedlichen Markierungen Cannot-Link-constraints gebildet. Dies geschieht nach einer *Random-Walk-Strategie*, damit die Must-Link-constraints, stellt man sich die Beispiele als Knoten und die constraints als Kanten vor, je Markierung genau eine Zusammenhangskomponente im Graphen bilden, was für die Initialisierung des clustering-Algorithmus von Bedeutung ist; wenn also (x_i, x_j) der zuletzt erzeugte constraint ist, dann wird im nächsten Schritt ein constraint (x_j, x_k) mit zufälliger Wahl von x_k und $k \neq i$ erzeugt.
Nach dem Lauf des MPCK-Means-Algorithmus, der in Abschnitt 3.3.1 näher erläutert wird, auf der Beispielmenge und der Menge der Nebenbedingungen ist jedes Beispiel bzw. jeder Merkmalausprägungsvektor einem cluster zugewiesen worden. Analog zu der Vorgehensweise der SVM Segmentierung wird daraus eine Cluster-Id-Folge erzeugt und die schon bekannte Ausreißerbehandlung durchgeführt. Am Ende bilden die Zeitabschnitte maximaler Teilfolgen mit konstanter Cluster-Id wieder die Segmente.

1. Manuelle Teilsegmentierung
2. Gefensterte Audiomerkmalsextraktion $\rightarrow \hat{x} = (\hat{x}_j)_{j \in \{0,\ldots,n-1\}}$
3. Generierung einer Beispielmenge aus gesamt $\hat{x}$ und einer Menge von Nebenbedingungen (Must-Link- und Cannot-Link-constraints) aus den markierten Beispielen
4. Clustering mit dem MPCK-Means-Algorithmus
5. Erzeugung einer Cluster-Id-Folge $c = (c_j)_{j \in \{0,\ldots,n-1\}}$
6. Ausreißerbehandlung mittels dynamischer Programmierung
7. Zeitabschnitte maximaler Teilfolgen mit konstanter Cluster-Id bilden die Segmente

3.3.1. MPCK-Means

Der halbüberwachte MPCK-Means-Algorithmus („**M**etric **P**airwise **C**onstraints K-Means“) [BBM04] basiert auf dem K-Means-Prinzip, verwendet aber zum einen eine parametrisierte euklidsche Distanz folgender Form:

$$||x_i - x_j||_A = \sqrt{(x_i - x_j)^T A (x_i - x_j)}.$$

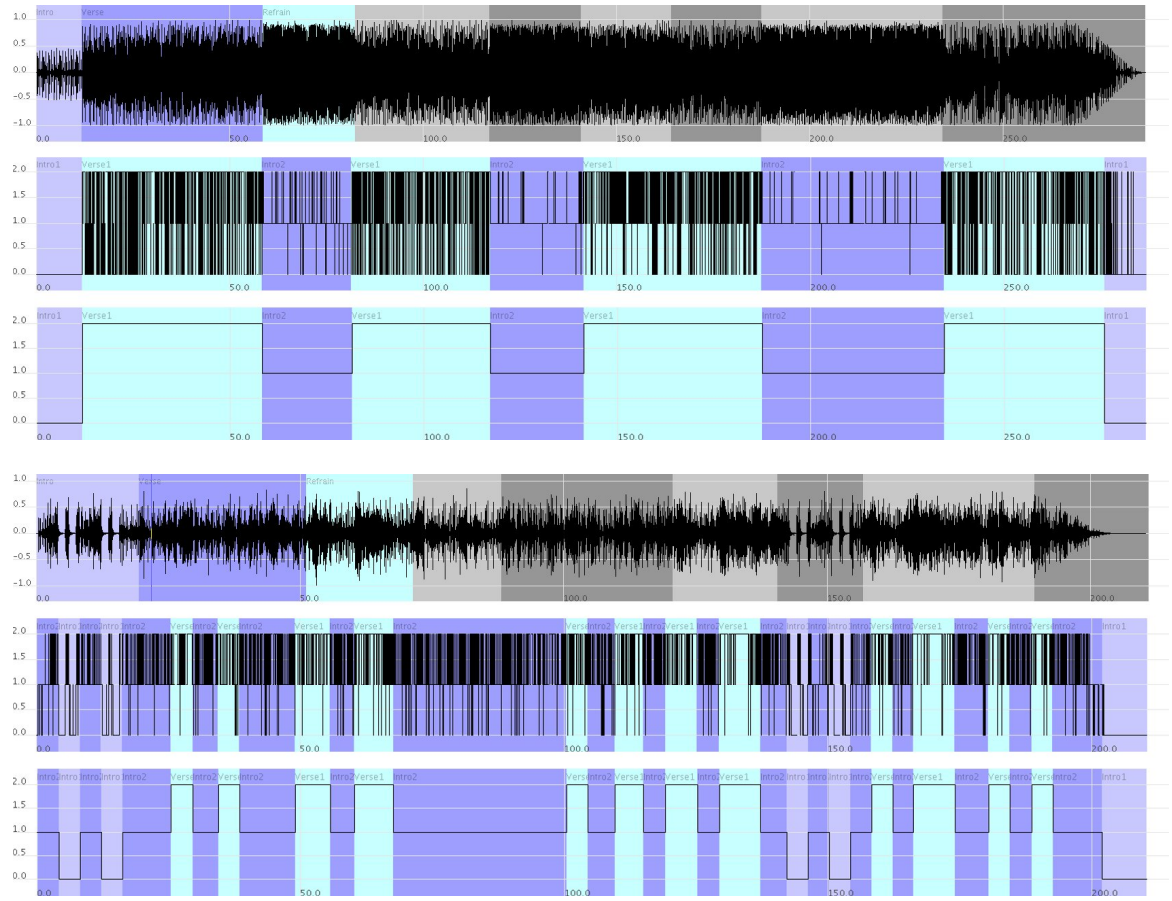


Abbildung 3.8.: Die Constrained clustering Segmentierung der Lieder „Believe“ von King's X (oben) und „Struggling Man“ von Jimmy Cliff (unten). Zu sehen sind jeweils das Audiomaterial samt manueller Teilsegmentierung (farbig) sowie die Cluster-Id-Folge vor und nach Anwendung der Ausreißerbehandlung zusammen mit dem Segmentierungsergebnis. Es wurden 200 Nebenbedingungen erzeugt. Die Parameter der Ausreißerbehandlung waren $minlength = 4s, maxlength = 40s$. Als Merkmale dienten MFCC.

Ist die Matrix A die Identitätsmatrix, dann ist obiger Ausdruck nichts anderes, als die herkömmliche euklidsche Distanz. Ist A auf eine Diagonalmatrix beschränkt, so wird durch sie eine Merkmalsgewichtung vorgenommen. Im allgemeinen Fall werden durch diese Parametrisierung neue Merkmale aus der Linearkombination der Originalen gebildet. Das Ziel dieser Parametrisierung ist es, ein lernbares Distanzmaß zum Ausgleich der Variationen der Merkmalsausprägungen, die als ähnlich angesehen werden sollen, zu liefern, d. h. durch A werden die Merkmalsausprägungsvektoren in einen Raum transformiert, in welchem sie betragsmäßig nahe beieinander liegen.
Zum anderen wird bei diesem clustering-Verfahren Vorwissen in Form von paarweisen Must-Link- und Cannot-Link-constraints in die Berechnung der Zielfunktion und des Distanzmaßes miteinbezogen, d. h. es stehen im Vorhinein Informationen darüber zur Verfügung, welche Objekte zur selben und welche keinesfalls zur selben cluster-Gruppe gehören sollen. Verletzungen dieser Nebenbedingungen werden als gewichtete Summe der Zielfunktion aufgeschlagen, die sich folgendermaßen darstellt:

$$\begin{aligned} J_{mpckm} \quad = \quad & \sum_{x_i \in \mathcal{X}} \quad \Big(||x_i - \mu_{l_i}||^2_{A_{l_i}} - \log\left(\det(A_{l_i})\right) \Big) \\ + \quad & \sum_{(x_i,x_j) \in \mathcal{M}} \quad w_{ij} f_{\mathcal{M}}(x_i, x_j) \mathbf{1}[l_i \neq l_j] \\ + \quad & \sum_{(x_i,x_j) \in \mathcal{C}} \quad \bar{w}_{ij} f_{\mathcal{C}}(x_i, x_j) \mathbf{1}[l_i = l_j] \end{aligned}$$

mit $\mathcal{X}$ Cluster, μ_{l_i} Centroid des Cluster, $\mathcal{M}, \mathcal{C}$ Mengen der Must-Link- und der Cannot-Link-constraints, $w_{ij}, \bar{w}_{ij}$ constraint-Gewichte, $f_{\mathcal{M}}, f_{\mathcal{C}}$ Distanzgewichtungsfunktionen, welche constraints, die im Falle eines Must-Link-constraint weit voneinander entfernte Objekte, im Falle eines Cannot-Link-constraint nahe Objekte betreffen, stärker gewichten, $\mathbf{1}$ Indikatorfunktion, die 1 ist, falls die Bedingung in den eckigen Klammern erfüllt ist, sonst 0, l_i, l_j Clusterzuweisungen der Objekte x_i, x_j.
Für die Berechnung der Distanzen wird für jeden Cluster eine eigene Matrix A_h verwendet. Nach der Cluster-Zuweisungsphase des Algorithmus, werden die Matrizen der Distanzmaße auf der Grundlage des aktuellen Cluster-Modells, also der Cluster-Zugehörigkeiten der Objekte, gelernt. Dies geschieht durch die partielle Ableitung der Zielfunktion nach A_h, also $\frac{dJ_{mpckm}}{dA_h}$, und die Bestimmung der Nullstelle, und resultiert in:

$$A_h = |\mathcal{X}_h| \left[\begin{array}{ll} & \sum_{x_i \in \mathcal{X}_h} \quad (x_i - \mu_h)(x_i - \mu_h)^T \\ + & \sum_{(x_i,x_j) \in \mathcal{M}_h} \quad \frac{1}{2} w_{ij} (x_i - x_j)(x_i - x_j)^T \mathbf{1}[l_i \neq l_j] \\ + & \sum_{(x_i,x_j) \in \mathcal{C}_h} \quad \bar{w}_{ij} \left((x'_h - x''_h)(x'_h - x''_h)^T - (x_i - x_j)(x_i - x_j)^T \right) \mathbf{1}[l_i = l_j] \end{array} \right]^{-1}$$

mit $\mathcal{X}_h$ Cluster h, μ_h Centroid des Cluster h, $\mathcal{M}_h, \mathcal{C}_h$ Mengen der Must-Link- und der Cannot-Link-constraints, die Objekte des Cluster h betreffen, $w_{ij}, \bar{w}_{ij}$ constraint-Gewichte,

x'_h, x''_h Objektpaar des Cluster h mit größter Distanz, **1** Indikatorfunktion, die 1 ist, falls die Bedingung in den eckigen Klammern erfüllt ist, sonst 0, l_i, l_j Clusterzuweisungen der Objekte x_i, x_j.
Da die Berechnung von A_h die Inversion der Summe der Kovarianzmatrizen beinhaltet, darf letztere nicht singulär sein. Desweiteren muß sichergestellt sein, dass nach der Inversion A_h nicht negativ definit ist, da sie ansonsten keine Distanzfunktion parametrisiert. Für beide Ausnahmefälle existieren Verfahren, welche die Matrix wieder in eine gültige Form überführen. In dieser Arbeit kommt jedoch eine einfachere Strategie zum Einsatz: Sollte A_h nicht neu berechnet werden können, so bleibt sie, wie sie ist.

Initialisierung Mit Hilfe der Informationen der Nebenbedingungen wird versucht, eine möglichst gute Initialisierung der Cluster-Schwerpunkte oder -Centroide durchzuführen. Dazu werden zunächst die Zusammenhangskomponenten des durch die Must-Link-constraints gebildeten Graphen berechnet, wodurch man die sogenannten Nachbarschaftsmengen oder *neighbourhood sets* $N = \{N_p | p = 1, \ldots, \lambda\}$ erhält, also Mengen N_p von Objekten, die jeweils als ähnlich betrachtet werden und später zusammen in einen Cluster gehören sollen. Nach der Berechnung der Centroide dieser Mengen, werden jene durch eine gewichtete farthest-first-Traversierung als initiale cluster-Centroide übernommen, wobei größere Nachbarschaftsmengen stärker gewichtet werden als kleinere. Es werden also die Centroide größten Abstands und größter Nachbarschaftsmenge ausgewählt. Sollte $|N| < K$ sein, dann werden die restlichen Centroide zufällig bestimmt.

4. Segfried

Segfried [Dax06] ist eine Java-Anwendung, die als Teil dieser Diplomarbeit entwickelt wurde, um die benötigten Funktionalitäten, wie Audiomerkmalsextraktion, manuelle Segmentierung, automatische Segmentierung und deren Evaluation, in einem Rahmenwerk bereitzustellen. Dabei wurde besonderer Wert auf ein generisches Konzept, leichte Erweiterbarkeit und die Möglichkeit zur Visualisierung gelegt.
Die Datenverarbeitung in *Segfried* basiert auf dem Konzept der Operatorbäume, welches in Kapitel 3 beschrieben wird. Nach der Implementierung geeigneter Operatoren und Funktionen ist es möglich, Operatorbäume sowohl für die Extraktion der Audiomerkmale als auch die Durchführung der Experimente mit Hilfe des Operatorbaumeditors (Abschnitt 4.4) zu erstellen.
In die Implementierung von *Segfried* wurde Programmtext aus Java-Klassen bereits existierender Software eingeflochten:

YALE Java-Anwendung zur Erstellung, Durchführung und Evaluation von Lernexperimenten, Version 3.3 [MWK+06]

jAudio Java-Anwendung zur Extraktion von Audiomerkmalen, Betaversion vom 27.08.2005 [MMFD05]

OC Volume Java-Software zur Unterstützung von automatischer Spracherkennung [SFL02]

Zur Verfeinerung der grafischen Oberfläche wurden Grafiken des KDE Symboldesigns „kids“ verwendet.

4.1. Das Hauptfenster

Nach dem Start von *Segfried* öffnet sich das Hauptfenster (siehe Abb. 4.1). Am oberen Rand befindet sich eine kleine Werkzeugleiste, welche Symbolknöpfe zur Konfiguration, Information und zum Beenden des Programmes enthält. Auf der linken Seite wird eine Liste der vorhandenen Operatorbäume angezeigt und die Möglichkeit zum Laden, Speichern und zur Neuerstellung gegeben. Rechts befinden sich die Liste der geladenen Wertefolgen und der Operatorbaumeditor, auszuwählen über die Reiter an der oberen Kante des Anzeigebereiches. Am unteren Rand schließlich sind die Schaltfläche für das Starten einer Operation,

ein Fortschrittsbalken und Informationen über den verwendeten Speicher zu sehen. Letztere bestehen aus drei Zahlen $M_{free}/M_{alloc}/M_{max}$ mit folgender Bedeutung:

M_{free} ist der von der *Java Virtual Machine* allokierte **freie** Speicher

M_{alloc} ist der von der *Java Virtual Machine* allokierte Speicher

M_{max} ist der maximale Speicher, der von der *Java Virtual Machine* allokiert werden kann

Ein linker Mausklick (eine Rechtshändermaus vorausgesetzt) auf das Speichersymbol startet die Java Speicherbereinigung („garbage collection“).

4.2. Konfiguration

Zur Ansicht und Änderung von Grundeinstellungen, öffnet sich nach einem linken Mausklick auf das Werkzeugsymbol der Werkzeugleiste das Konfigurationsfenster mit einer Liste von benutzerveränderlichen Parametern. Eine kurze Erklärung eines Parameters bzw. die Angabe möglicher Werte erscheint, wenn man den Mauszeiger kurz über der entsprechenden Zeile verweilen läßt. Zum Editieren klickt man auf den zu ändernden Wert und schließt die Eingabe mit der <return>-Taste zum Übernehmen oder der <esc>-Taste zum Abbruch des Vorgangs ab. Die Schaltfläche am unteren Rand des Fensters dient dem persistenten Sichern der Konfiguration, was jedoch auch beim (regulären) Beenden von *Segfried* geschieht. Die Konfigurationsdateien werden in ein Verzeichnis „.segfried“ im Heimverzeichnis des Benutzers geschrieben.
Mit den Konfigurationsparameter kann man einstellen:

functionDir/operatorDir Die Pfade zu Verzeichnissen, in welchen nach Funktionenklassen bzw. Operatorklassen gesucht wird. Diese werden beim Start von *Segfried* geladen und stehen dann im Operatorbaumeditor zur Verfügung

operatorTreeDir Das Hauptverzeichnis für Operatorbäume

seriesDir Das Hauptverzeichnis für Wertefolgen

loadOpTrees Ob beim Start von *Segfried* die Operatorbäume im Hauptverzeichnis automatisch geladen werden sollen

loadSeries Ob beim Start von *Segfried* die Wertefolgen im Hauptverzeichnis automatisch geladen werden sollen

seriesFormat Ob Wertefolgen als Java Objekt oder im Weka ARFF Format ([WF05]) gespeichert werden sollen

segFormat Ob Segmentierungen als Java Objekt oder im Weka ARFF Format gespeichert werden sollen

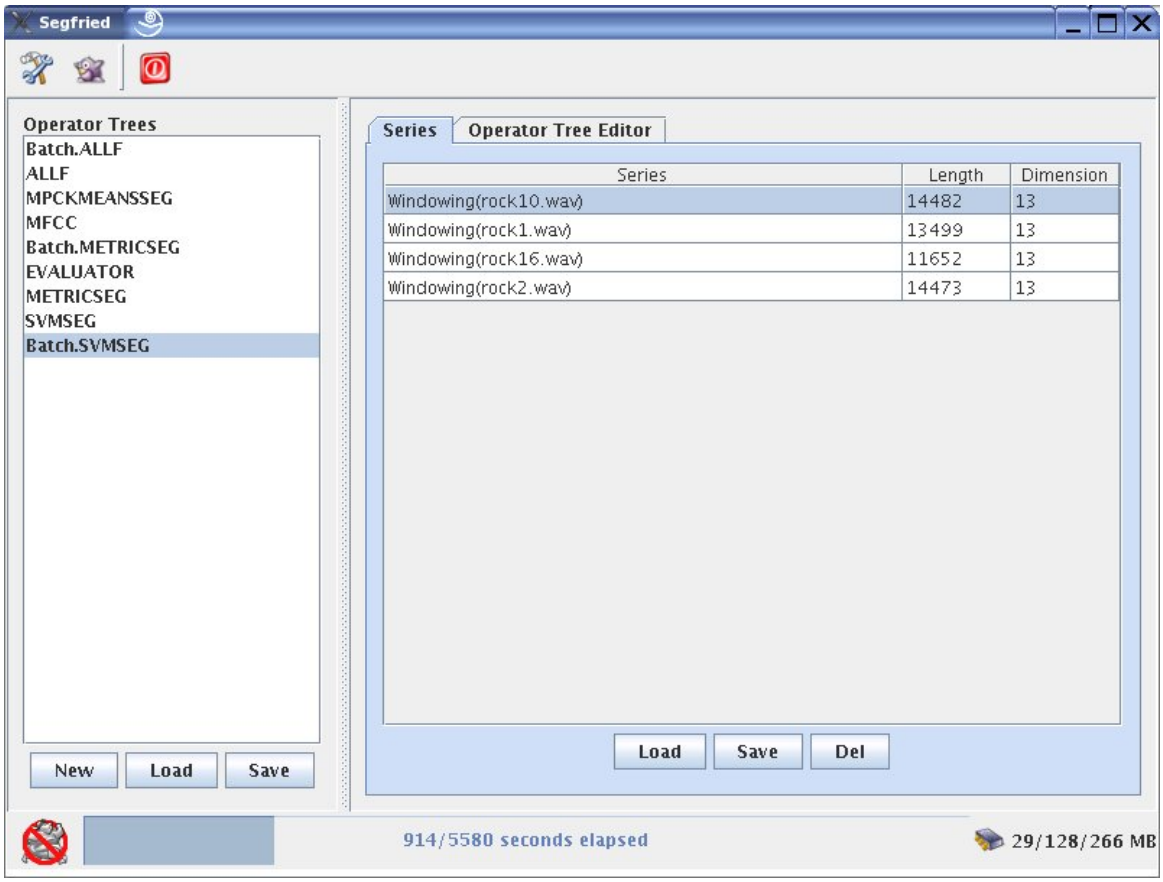


Abbildung 4.1.: Das *Segfried* Hauptfenster und die Wertefolgenliste

4.3. Wertefolgen

Alle geladenen und von *Segfried* erzeugten Wertefolgen werden in der Liste auf der rechten Seite des Hauptfensters angezeigt. Mit den Schaltflächen darunter können Folgen geladen, gespeichert oder wieder aus der Liste entfernt werden. Ein doppelter linker Mausklick auf eine Wertefolge öffnet ein Fenster mit der Standardvisualisierungskomponente, dem Plotter, ein rechter Mausklick ein Menue zur Auswahl der Anzeige einer Parameterbeschreibung, des Plotters oder der Karte. Die Parameterbeschreibung liefert Informationen über die Parameterwerte der einzelnen Operatoren und Funktionen des Operatorbaumes, welcher zur Erzeugung der Wertefolge geführt hat. Plotter und Karte werden in Abschnitt 4.3.1 näher erklärt.

Laden und Speichern Zum Laden einer Wertefolge betätigt man die Schaltfläche „Load“ unterhalb der Wertefolgenliste, und kann dann die gewünschte Datei in dem Dateiauswahlfenster auswählen. Gültige Dateiformate sind:

- .sfs (*Segfried* series)
- .wav (Microsoft wave)
- .arff (Weka ARFF)

Das Speichern einer Wertefolge erreicht man durch Markieren der gewünschten Folge, einen anschliessenden linken Mausklick auf die Schaltfläche „Speichern“ und zu guter Letzt die Angabe des Dateinamens.

4.3.1. Visualisierung

Zur Visualisierung der Wertefolgen stehen in *Segfried* zwei Komponenten zur Verfügung: der Plotter (Abb. 4.2) zur zweidimensionalen Darstellung jeder Dimension und die Karte (Abb. 4.3) zur quasi-dreidimensionalen Übersicht über eine Wertefolge. Das Fenster zur Darstellung beider Komponenten enthält am unteren Rand zusätzlich eine Funktionsleiste, mit welcher die Darstellung beeinflusst werden kann. Ganz links befindet sich die Schaltfläche „Recalc“, die den Anzeigebereich, z. B. nach einem Wechsel der Dimension, neu skaliert, um die Fläche der Komponente zur Anzeige der Werte optimal auszunutzen. Mit dem Knopf „Save Img“ darunter läßt sich die Komponente als JPEG-Bild speichern. Klickt man mit dem linken Mausknopf auf eine Komponente, dann werden die Koordinaten in den beiden Textfeldern „X“ und „Y“ angezeigt. Gleich daneben befinden sich die Schaltflächen „+“ und „-“ zum Vergrößern bzw. Verkleinern der Ansicht der entsprechenden Dimension. Dann folgen zwei Auswahlboxen, wobei mit der oberen die in Y-Richtung darzustellende Dimension der Wertefolge gewählt wird. Mit der anderen kann, im Falle einer komplexen Wertefolge, zwischen der Anzeige der Realteile, Imaginärteile und Länge gewechselt werden.
Die nun folgenden Elemente der Funktionsleiste dienen der Dar- und Erstellung von Segmentierungen. In der Auswahlbox werden die für die Wertefolge existierenden Segmentierungen namentlich angezeigt, mit dem Knopf „Add“ darüber können neue Segmentierungen hinzugefügt werden. Ein linker Mausklick auf den Namen der aktuellen Segmentierung ermöglicht die Eingabe eines Namens. Die beiden Schaltflächen „Del“ und „Save“ rechts daneben dienen der Entfernung und der Speicherung der aktuellen Segmentierung. Eine Segmentierungsdatei, welche die Endung „.seg“ erhält, wird entweder im selben Verzeichnis, wie die dazugehörige Wertefolge, falls die Wertefolge geladen wurde, oder im Hauptverzeichnis für Wertefolgen gespeichert. Mit der Auswahlbox ganz rechts lassen sich verschiedene Name/Farb-Kombinationen zur Kennzeichnung von Segmenten auswählen und mit dem Knopf darüber neue hinzufügen. Auch hier können die Namen per Eingabe geändert werden.
Im Falle einer Audiodatenfolge wird am rechten Ende der Funktionsleiste noch zusätzlich eine weiße Komponente zur Steuerung der akustischen Wiedergabe des Audiomaterials angezeigt.

Der Plotter Die Plotterkomponente (Abb. 4.2) dient der Darstellung einer Dimension einer Wertefolge in Abhängigkeit von der Zeit. Die ausgewählte Segmentierung wird als farbige Hinterlegung gezeichnet. Mit einem linken Mausklick auf den Plotter markiert man eine Position auf der Zeitachse.

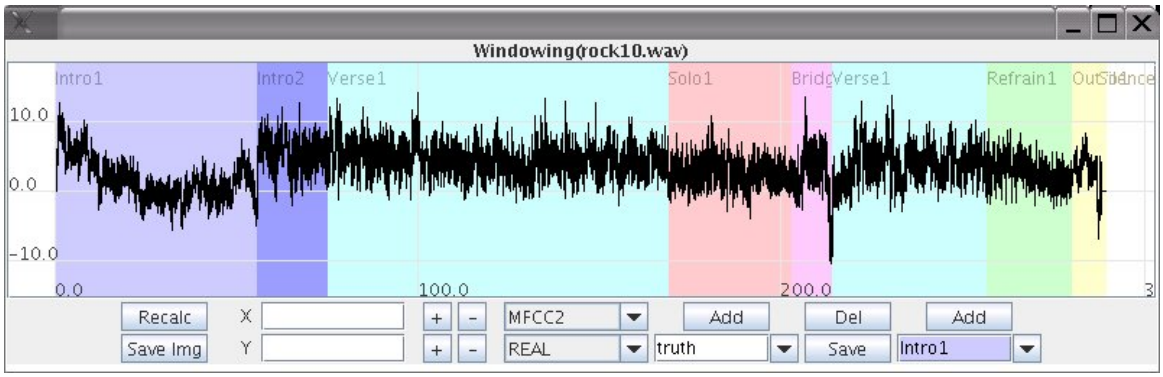


Abbildung 4.2.: Der *Segfried* Plotter

Die Karte Die Kartenkomponente (Abb. 4.3) liefert eine Gesamtübersicht über eine Wertefolge. Dabei entspricht die horizontale Achse der Zeit und die vertikale Achse den Dimensionen. Jeder Wert einer Dimension wird als Graustufenpixel dargestellt, wobei der größte Wert weiß und der niedrigste Wert schwarz gezeichnet wird. Am unteren Rand wird die ausgewählte Segmentierung angezeigt. Die Karte wurde in dieser Arbeit verwendet, um Ähnlichkeitsmatrizen zu visualisieren.

4.3.2. Segmentierung

Jede Wertefolge in *Segfried* besitzt eine Liste von Segmentierungen. Hierbei ist zu beachten, dass sich eine Wertefolge und alle aus ihr durch Anwendung eines Operatorbaumes entstandenen Wertefolgen ein und dieselbe Liste teilen, um die Ergebnisse der Transformation direkt mit der auf der ursprünglichen Wertefolge durchgeführten Segmentierung vergleichen zu können. Eine Segmentierung wiederum ist eine Liste von sich nicht überlappenden Abschnitten bzw. Segmenten auf der Zeitachse, die unterschiedlich gekennzeichnet sein können. Segmentierungen können mit Hilfe der Visualisierungskomponenten erstellt werden. Mit einem einfachen rechten Mausklick fügt man der aktuell ausgewählten Segmentierung an der entsprechenden Stelle eine Segmentgrenze hinzu, mit einem doppelten rechten Mausklick auf ein bestehendes Segment entfernt man dieses. Möchte man ein Segment kennzeichnen, dann wählt man zunächst in der Funktionsleiste die gewünschte Name/Farb-Kombination aus und führt dann einen doppelten linken Mausklick auf dem Segment durch. Es wird dann entsprechend eingefärbt und benannt. Gespeichert werden Segmentierungen durch

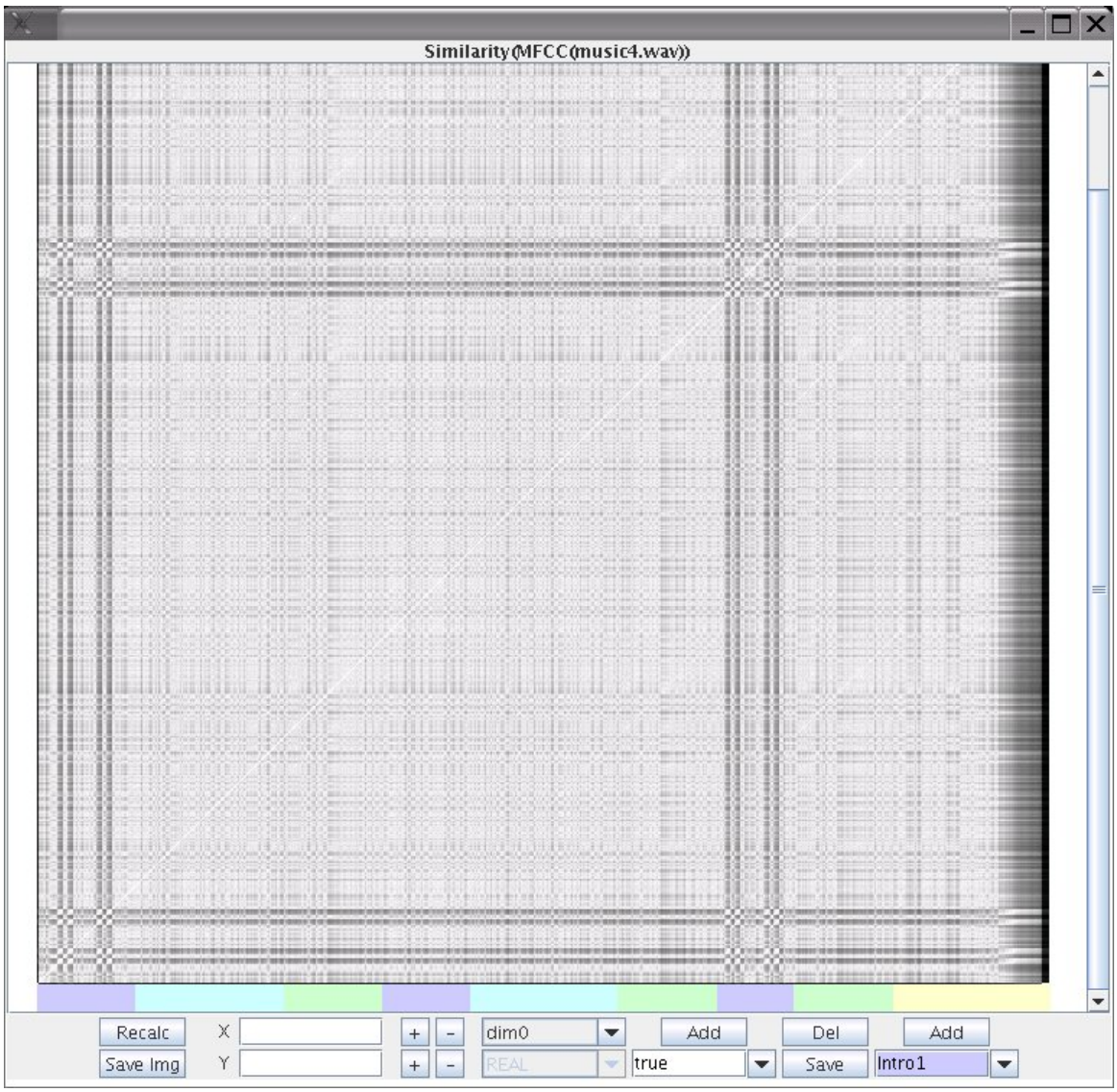


Abbildung 4.3.: Die *Segfried* Karte

einen linken Mausklick auf den „Save“ Knopf je nach Konfiguration als Java Objekt oder im Weka ARFF Format in einer Datei, deren Name aus dem Namen der Audiodatei, dem Namen der Segmentierung und der Endung „.seg“ besteht.

4.4. Operatorbäume

Operatorbäume, als zentrales Konzept der Datenverarbeitung, können mit Hilfe des Operatorbaumeditors (Abb. 4.4) erzeugt und dazu verwendet werden, Transformationen von Wertefolgen, wie z. B. die Extraktion von Audiomerkmalen, durchzuführen. Dazu stehen in *Segfried* bereits eine Anzahl von Einzeloperationen, nämlich Operatoren wie z. B. *Fast Fourier Transformation* oder *Auto Correlation* und Funktionen wie etwa *Max Finder* oder *Transpose*, zur Verfügung. Einige dieser Operationen besitzen die Eigenschaft, eine untergeordnete Operation oder Kette von Operationen auszuführen, wie z. B. der *Operator Chain* oder *Windowing* Operator, was dann zur Ausbildung der Baumstruktur führt. In Abschnitt B.2 befinden sich eine Übersicht und Erklärungen der vorhandenen Operatoren und Funktionen.

Erstellung Mit der Schaltfläche „New“ unter der Liste aller verfügbaren Operatorbäume auf der linken Seite des Hauptfensters erzeugt man einen neuen, leeren Operatorbaum mit angegebenem Namen. Wählt man einen Operatorbaum in der Liste aus, so wird seine Baumstruktur im Operatorbaumeditor angezeigt. Um eine Operation an geeigneter Stelle in einen Operatorbaum einzufügen, markiert man den übergeordneten oder vorhergehenden Knoten mit einem linken Mausklick, wählt die gewünschte Operation (Operator oder Funktion) aus und betätigt den zugehörigen „Add“ Knopf. Sollte dies nicht funktionieren, dann ist die markierte Stelle vielleicht doch eher ungeeignet, und es sollte noch einmal ein Blick in die Operator- und Funktionenreferenz (Abschnitt B.2) gewagt werden. Mit den Schaltflächen „Up“ und „Dn“ kann eine Operation innerhalb einer Kette positioniert werden, und mit dem „Del“ Knopf läßt sich ein markierter Knoten samt seiner untergeordneten Operationen aus dem Baum entfernen.
Ein rechter Mausklick auf einen Knoten bringt ein kleines Menue zum Vorschein, mit dessen Hilfe man entweder diesen Knoten, wieder samt untergeordneter Strukturen, löschen, kopieren oder eine vorher kopierte Struktur an dieser Stelle einfügen kann.
Auf der rechten oberen Seite des Operatorbaumeditors werden die veränderbaren Parametern der ausgewählten Operation angezeigt. Ein linker Mausklick auf einen Parameterwert ermöglicht die Editierung des selbigen. Läßt man den Mauszeiger kurz auf einer Parameterzeile verweilen, dann erscheint eine Beschreibung des entsprechenden Parameters.

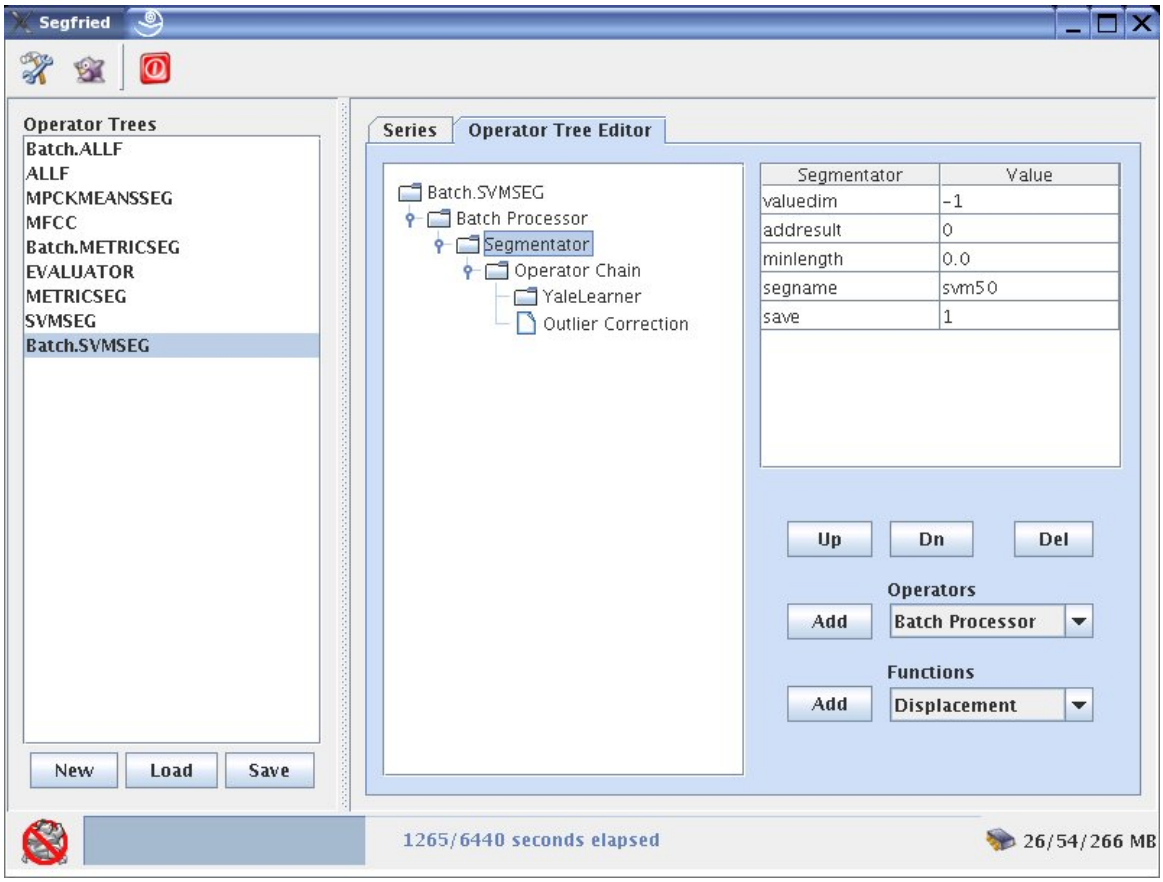


Abbildung 4.4.: Das *Segfried* Hauptfenster und der Operatorbaumeditor

Laden und Speichern Unterhalb der Liste der Operatorbäume befinden sich die Schaltflächen „Load“ und „Save“ für das Laden und Speichern. Vor dem Speichern ist darauf zu achten, dass der gewünschte Operatorbaum in der Liste ausgewählt wurde. Operatorbäume werden als Java Objekte in Dateien mit der Endung „.sot“ (*Segfried* operator tree) gespeichert.

Anwendung Um nun einen Operatorbaum auszuführen, markiert man ihn in der Liste und betätigt die Schaltfläche „Apply“ unten links neben dem Fortschrittsbalken. Möchte man den Operatorbaum auf eine Wertefolge anwenden, dann ist vorher ebenfalls die entsprechende Folge zu markieren.
Nachdem die Ausführung gestartet wurde, kann man den Anteil der bereits durchgeführten Berechnungen an der Gesamtoperation, die verstrichene und die geschätzte noch verbleibende Zeit am Fortschrittsbalken ablesen. Einen Abbruch der Ausführung erzielt man durch einen erneuten Mausklick auf den „Apply“ Knopf. Je nach Einstellung der Parameter, wer-

den die Ergebnisfolgen der Operatoren der Liste aller Wertefolgen hinzugefügt und stehen zur weiteren Verarbeitung bereit.

5. Evaluation

Im Folgenden wird nun die Evaluation der realisierten Segmentierungsverfahren beschrieben und ihr Ergebnis präsentiert und diskutiert. Grundlage der Evaluation war eine aus Ermangelung an mir bekannten frei zugänglichen Datensätzen, welche sowohl sprachliche als auch musikalische Audiodaten enthalten, selbst zusammengestellte und manuell segmentierte Sammlung von Audiodateien.

5.1. Datensatz

Die Audiodaten, auf welche die in Kapitel 3 dargestellten Verfahren angewendet wurden, setzen sich aus

- 100 Liedern der Stilrichtungen Klassik, Jazz, Reggae, Pop, Oldies, Rock
- 160 Minuten Radioprogramm

zusammen. Diese Auswahl deckt sowohl die Segmentierungsaufgabe im Bereich Musik als auch die Unterscheidung von Sprechern und verschiedenen Klangklassen (Sprache unterschiedlicher Qualität, Musik) ab. Die Lieder stammen fast ausschließlich von CD-Musikzusammenstellungen (Sampler), um eine möglichst große Klangvielfalt zu gewährleisten. Die Radiosendungen lagen ursprünglich als private Mitschnitte im mp3-Format vor und wurden über das Internet heruntergeladen. Alle Dateien wurden in das „Microsoft wave“-Format mit einem Kanal (mono), 16 bit Quantisierungsauflösung und 22050 Hz Samplingfrequenz transkodiert. Ca. 60 % des Audiomaterials wurde von mir, der Rest von insgesamt sieben Personen arbeitsteilig mit Hilfe von *Segfried* (Kapitel 4) manuell segmentiert, um die Vergleichsgrundlage für die automatischen Verfahren zu schaffen. Die Probanden wurden angewiesen, während des Anhörens der zu segmentierenden Audiodaten Stellen zu markieren, an denen ihrer Meinung nach ein neuer Abschnitt beginnt, wobei die Definition eines Abschnitts dem Verständnis des Probanden überlassen wurde. Die Bedienung von *Segfried* und die anschließende Klassifikation der Segmente wurde dabei von mir erledigt.
Um Inkonsistenzen in der menschlichen Auffassung darüber, an welchen Stellen ein neues Segment beginnt, auszugleichen, wurde in [TC99] das gesamte Audiomaterial (10 Audiodateien mit je einer Länge von ca. 1 Minute) von mehreren Personen segmentiert und nur solche Segmentgrenzen zur Evaluation herangezogen, über die sich eine Mehrheit (innerhalb

gewisser Toleranzgrenzen) einig waren. Diese Verfahrensweise wäre hier allerdings aufgrund der deutlich größeren Audiodatenmenge zu aufwändig gewesen.

5.2. Experimente

Die Experimente wurden auf einem Rechner mit Intel Pentium M 1500 MHz Prozessor und 512 MB Arbeitsspeicher unter einem Linux Betriebssystem (Suse Linux 9.1) durchgeführt. Als Laufzeitumgebung der Java-Software kam das *Java 2 Runtime Environment* in der Version 1.5 zum Einsatz. Mit der Software *Segfried* (siehe Kapitel 4) wurden die Audiodateien manuell segmentiert, die Segmentierungsexperimente erstellt und ausgeführt und die Evaluation vorgenommen. Für die Erledigung der Lernaufgaben innerhalb der Segmentierungsexperimente wurden von *Segfried* die Klassenbibliothek der Lernumgebung YALE [MWK+06] und entsprechende Lernexperimente, welche zuvor mit YALE erzeugt wurden, verwendet.

5.2.1. Extrahierte Audiomerkmale

Für die Beschreibung der Eigenschaften des Audiomaterials wurden drei Merkmalsätze verwendet und jedes Verfahren auf jedem Satz evaluiert. Da MFCC eine gute Charakterisierung des Audiospektrums liefern, besteht der Erste (**MFCC**) aus den ersten 13 dieser Koeffizienten. Der Zweite (**ALLF**) ist eine Erweiterung des Ersten um neun gebräuchliche und erfolgreiche Merkmale, die in der Audiosegmentierung und Audiogenreklassifikation zum Einsatz kommen, und hat einen Umfang von 22 Merkmalen. Der dritte Satz (**NGRAM**) besteht aus einem 5-Gram der ersten sechs MFCC, also insgesamt 30 MFCC Merkmalen. Dies ist der Versuch, die zeitliche Entwicklung eines Audiospektrums miteinzubeziehen.

MFCC Zur Extraktion der MFCC Merkmale wurde zunächst eine Fensterung mit einer Fenstergröße von 40ms und einer Schrittweite von 20ms durchgeführt. Für jedes Fenster wurde nun zur Vermeidung von Kanteneffekten zuerst eine Fensterfunktion (Hannfunktion) auf die Teilfolge angewendet, dann mittels einer schnellen Fouriertransformation (FFT, zeropadded) das Amplitudenspektrum (Abschnitt A.3) und abschließend die ersten 13 MFCC (23 Mel-Filter, tiefste Mittenfrequenz 133,333Hz) berechnet.

Abbildung 5.1.: Der Operatorbaum für die Extraktion des MFCC Merkmalsatzes

ALLF Die Berechnung der ALLF Merkmale geschah ebenfalls mit Hilfe einer Fensterung der Größe 40ms und Schrittweite 20ms. Für jedes Fenster wurde zunächst Zero Crossing Rate (ZCR) und Root Mean Square (RMS) als Merkmale des Zeitbereichs extrahiert. Darüber hinaus wurde die Teilfolge parallel in den Phasenraum und, nach Anwendung der Hann-Fensterfunktion und schnellen Fouriertransformation (FFT, zeropadded), in den Frequenzbereich transformiert. Im Phasenraum fand die Berechnung der Durchschnittswerte und Varianzen der Winkel zwischen den Vektoren (Zirkulärität) und der Vektorlängen statt. Im Frequenzbereich wurden der spektrale Flux (SF), Schwerpunkt (SC), Rolloff (SR) und zu guter letzt wieder die ersten 13 MFCC (23 Mel-Filter, tiefste Mittenfrequenz 133,333Hz) berechnet.

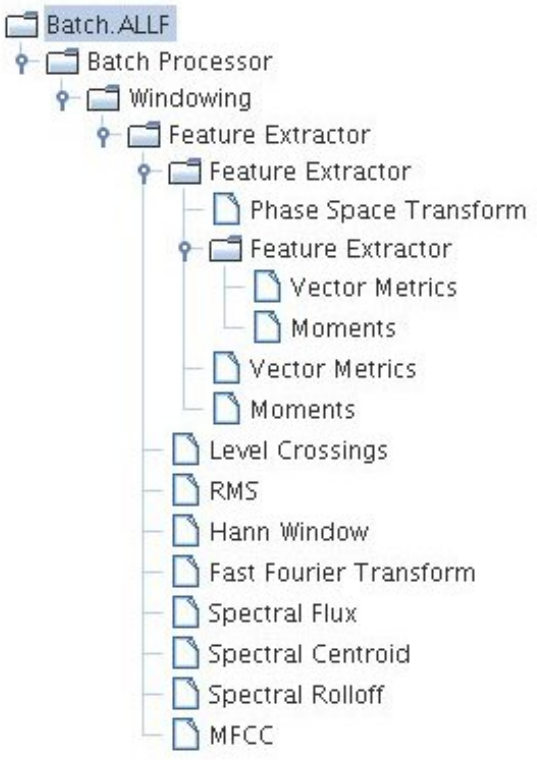


Abbildung 5.2.: Der Operatorbaum für die Extraktion des ALLF Merk-

NGRAM Der erste Schritt der Extraktion der NGRAM Merkmale wurde mit einer Fenstergröße von 100ms und Schrittweite von 50ms durchgeführt und bestand, analog zur Vorgehensweise beim MFCC Merkmalsatz, in der Berechnung der ersten 6 MFCC für jedes Fenster. Auf dieser Merkmalsausprägungsfolge erfolgte dann eine zweite Fensterung mit Größe 5 und Schrittweite 1, wobei je Fenster aus den fünf Merkmalsausprägungsvektoren mit Dimension 6 ein Vektor der Dimension 30 erzeugt wurde.

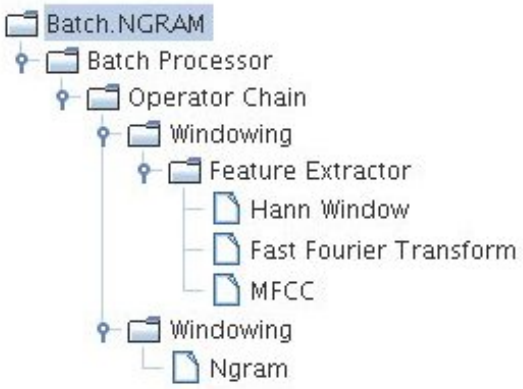


Abbildung 5.3.: Der Operatorbaum für die Extraktion des NGRAM Merkmalsatzes

5.2.2. Durchführung

Während der manuellen Segmentierung wurde für jede Audiodatei eine „wahre“ Vergleichssegmentierung „truth“ erstellt und gespeichert. Auf deren Grundlage wurden dann jeweils die Segmentierungen „truth10“, „truth30“ und „truth50“ automatisch erzeugt, welche für jede in „truth“ vorkommende Segmentmarkierung Segmente derselben Markierung enthielten, die aber je nur die ersten 10, 30 und 50 Prozent der Abschnitte einer Markierung abdeckten. Die erste Phase der Experimente bestand dann in der Extraktion der oben beschriebenen Merkmale, wobei je Audiodatei drei Merkmalsausprägungsfolgen entstanden, welche unter Übernahme der bereits existierenden Segmentierungen wiederum gespeichert wurden. In der zweiten Phase wurden mit Hilfe der realisierten Verfahren die automatischen Segmentierungen berechnet. Unter Variation eines entscheidenden Parameters wurde jedes Verfahren für jede Merkmalsausprägungsfolge dreimal gestartet und das Resultat gespeichert.

„Metrikbasierte“ Segmentierung Hierbei wurde die Fenstergröße für die Berechnung der Kriteriumsfunktion variiert. Das Verfahren lief für Fenster der Größe 4s, 10s und 16s immer mit einer Schrittweite von 0,1s. Die Bestimmung der Spitzen der resultierenden Wertefolgen geschah mit den Parametern $number = 30, minpeakdist = 2s, sloppy = 2, tolerance = 1.3$. Es wurden die Segmentierungen „metric4“, „metric10“ und „metric16“ erzeugt.

SVM Segmentierung Bei der SVM Segmentierung veränderte ich den Umfang der Trainingsmenge, welche je einmal aus der Segmentierung „truth10“, „truth30“ und „truth50“ jeder Merkmalsausprägungsfolge erzeugt wurde, d. h. die Trainingsmenge bildeten 10, 30 oder 50 Prozent aller Merkmalsausprägungsvektoren einer Klasse. Die Klasse entspricht dabei der Markierung des Segments, innerhalb dessen ein Merkmalsausprägungsvektor liegt, welche zur Markierung der Beispiele verwendet wurde. Die Parameter der Ausreißerbehandlung waren $minlength = 4s, maxlength = 40s$, nur für die Radioaudiodaten betrug aufgrund der teilweise auftretenden kurzen Sprechersegmente $minlength = 2s$. Die Ausgabe dieses Verfahrens bestand in den Segmentierungen „svm10“, „svm30“ und „svm50“.

Constrained clustering Segmentierung Der variable Parameter war hier die Anzahl der erzeugten Nebenbedingungen, welche 0, 100 oder 200 betrug. Aus der Segmentierung „truth30“ wurde eine Beispielmenge generiert, welche teilweise (zu 30 Prozent) gekennzeichnete Beispiele enthielt, woraus wiederum zu gleichen Anteilen Must-Link- und Cannot-Link-constraints gebildet wurden. Der Parameter k des MPCK-Means-Algorithmus wurde immer auf die Anzahl der tatsächlich vorhandenen Klassen, also der unterschiedlichen Segmentmarkierungen, gesetzt, und es wurde für jeden Cluster eine eigene, komplett besetzte Matrix für das Distanzmaß verwendet. Ein Lauf des Algorithmus bestand aus 2 Durchgängen mit je 10 Optimierungsschritten. Die Ausreißerbehandlung erfolgte wie bei der SVM Segmentierung

angegeben. Die berechneten Segmentierungen wurden „mpck0“, „mpck100“ und „mpck200“ genannt.

5.2.3. Laufzeit

Für eine Audiodatei der Länge von 210s betragen die Laufzeiten der Merkmalsextraktion auf dem oben beschriebenen Rechnersystem:

Merkmalsatz	MFCC	ALLF	NGRAM
Extraktionsdauer/s	15	45	17

Die Laufzeiten für die Berechnung der Segmentierungen ergeben sich für eine Merkmalsausprägungsfolge, welche durch Extraktion der ALLF Merkmale aus derselben Audiodatei entstanden ist und aus insgesamt 10510 Merkmalsausprägungsvektoren der Dimension 22 besteht, zu:

Segmentierung	Dauer/s	Segmentierung	Dauer/s	Segmentierung	Dauer/s
metric4	2	svm10	46	mpck0	53
metric10	4	svm30	142	mpck100	201
metric16	15	svm50	294	mpck200	261

5.3. Ergebnisse

Alle automatisch erzeugten Segmentierungen wurden von mir im Hinblick auf die Segmentgrenzen evaluiert. In den Tabellen 5.1, 5.2 und 5.3 sind die ausführlichen Ergebnisse für die drei realisierten Verfahren nach Merkmalsatz und Audiodatenart aufgeschlüsselt für eine feste Evaluationstoleranz von 1s zu sehen. Tab. 5.4 zeigt die Gesamtergebnisse für jedes Verfahren und jeden Merkmalsatz über alle Audiodaten, diesmal in Abhängigkeit von der Evaluationstoleranz für die Werte 0,5s, 1s und 2s.
Da die SVM Segmentierung zusätzlich eine Klassifikation der Segmente lieferte, habe ich die Segmentierungen dieses Verfahrens zusätzlich hinsichtlich der Segmentmarkierungen evaluiert. Das Ergebnis wird in Tab. 5.5 gezeigt.
Für die Bewertung habe ich die Standardmaße *precision* (PRC), *recall* (RCL) und das *F-measure* (F) verwendet. Maße wie *False Alarm Rate* (FAR) und *Missed Detection Rate* (MDR) sind im Hinblick auf die Evaluation von Segmentgrenzen ebenfalls gebräuchlich. PRC ist dabei mit der FAR negativ korreliert, d. h. je weniger falsche Segmentgrenzen erkannt wurden, desto größer ist PRC. Analog verhält es sich mit RCL und MDR, d. h. je weniger korrekte Segmentgrenzen ausgelassen wurden, desto größer ist RCL. Da es das Ziel ist, möglichst wenig falsche und gleichzeitig möglichst viele korrekte Segmentgrenzen zu erkennen, gibt es das *F-measure*, welches PRC und RCL gewichtet vereint.

5.3.1. Evaluation der Segmentgrenzen

Zur Berechnung von PRC, RCL und F wurde zunächst die Anzahl der Segmentgrenzen in der wahren („true“) und der zu evaluierenden („hypothesized“) Segmentierung, n^t und n^h , ermittelt. Danach erfolgte unter Berücksichtigung einer Evaluationstoleranz *tol* die Bestimmung der Anzahl der korrekt erkannten Segmentgrenzen n^c nach folgendem Algorithmus:

INPUT: true segmentation T, hypothesized segmentation H, evaluation tolerance *tol*
OUTPUT: number of correctly hypothesized boundaries n^c
INITIALIZATION: b^t = displacement of first true boundary, b^h = displacement of first hypothesized boundary, $n^c = 0$
$EvaluateBoundaries(T, H, tol)$
(1) DO
(2) IF $b^h \in \left[b^t - tol, b^t + tol\right]$ THEN
(3) $n^c = n^c + 1$
(4) b^t = displacement of next true boundary or NaN
(5) b^h = displacement of next hypothesized boundary or NaN
(6) ELSE
(7) IF $b^t < b^h$ THEN
(8) b^t = displacement of next true boundary or NaN
(9) ELSE
(10) b^h = displacement of next hypothesized boundary or NaN
(11) WHILE $b^t \neq$ NaN AND $b^h \neq$ NaN
(12) RETURN n^c

Für die Evaluation der Segmentgrenzen ergeben sich PRC, RCL und F somit zu:

$$PRC = n^c/n^h \qquad RCL = n^c/n^t \qquad F = \frac{2*PRC*RCL}{PRC+RCL}$$

5.3.2. Evaluation der Segmentmarkierungen

Da das Verfahren der SVM Segmentierung zusätzlich zu der Information über die Segmentgrenzen eine Klassifikation der Segmente lieferte, bot es sich an zu überprüfen, inwieweit die Markierungen der Segmente der berechneten Segmentierungen mit denen der manuell erstellten übereinstimmen. Dazu wurde jeweils die Summe der Längen aller Segmente in der wahren und in der zu evaluierenden Segmentierung l^t und l^h berechnet. Da alle Segmentierungen, abgesehen von kleinsten Unterschieden, die durch die gefensterte Berechnung der Merkmalsausprägungsfolgen entstanden, von Anfang bis Ende einer Audiodatei reichen, gilt $l^t \approx l^h$. Die Summe der Längen der Segmentabschnitte, welche korrekt markiert wurden, l^c wurde wie folgt berechnet:

ALLF

	metric4			metric10			metric16		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.25	0.58	**0.35**	0.30	0.56	**0.40**	0.31	0.44	**0.36**
jazz	0.27	0.63	**0.38**	0.26	0.53	**0.35**	0.36	0.49	**0.42**
oldies	0.38	0.64	**0.48**	0.43	0.54	**0.48**	0.43	0.41	**0.42**
pop	0.26	0.54	**0.35**	0.39	0.45	**0.42**	0.48	0.39	**0.43**
radio	0.33	0.46	**0.38**	0.49	0.27	**0.35**	0.53	0.23	**0.32**
reggae	0.21	0.54	**0.30**	0.27	0.46	**0.34**	0.38	0.43	**0.41**
rock	0.27	0.57	**0.37**	0.40	0.56	**0.47**	0.47	0.45	**0.46**
ø	0.28	0.57	**0.37**	0.36	0.48	**0.40**	0.42	0.41	**0.40**

MFCC

	metric4			metric10			metric16		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.25	0.58	**0.35**	0.30	0.58	**0.39**	0.29	0.44	**0.35**
jazz	0.27	0.63	**0.38**	0.26	0.54	**0.35**	0.31	0.47	**0.37**
oldies	0.33	0.60	**0.42**	0.38	0.55	**0.45**	0.46	0.46	**0.46**
pop	0.25	0.52	**0.33**	0.29	0.43	**0.35**	0.47	0.41	**0.44**
radio	0.34	0.49	**0.40**	0.51	0.29	**0.37**	0.60	0.27	**0.37**
reggae	0.22	0.58	**0.32**	0.24	0.50	**0.33**	0.32	0.45	**0.37**
rock	0.26	0.55	**0.35**	0.33	0.53	**0.41**	0.43	0.45	**0.44**
ø	0.27	0.56	**0.36**	0.33	0.49	**0.38**	0.41	0.42	**0.40**

NGRAM

	metric4			metric10			metric16		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.26	0.60	**0.36**	0.30	0.57	**0.39**	0.31	0.44	**0.36**
jazz	0.26	0.61	**0.36**	0.25	0.51	**0.33**	0.32	0.44	**0.37**
oldies	0.32	0.65	**0.43**	0.37	0.53	**0.44**	0.43	0.45	**0.44**
pop	0.24	0.51	**0.32**	0.29	0.43	**0.34**	0.45	0.41	**0.43**
radio	0.32	0.50	**0.40**	0.47	0.39	**0.42**	0.56	0.29	**0.38**
reggae	0.22	0.60	**0.32**	0.24	0.52	**0.33**	0.30	0.46	**0.37**
rock	0.26	0.55	**0.35**	0.35	0.56	**0.43**	0.45	0.47	**0.46**
ø	0.27	0.57	**0.36**	0.32	0.50	**0.38**	0.40	0.42	**0.40**

Tabelle 5.1.: Evaluation der Segmentgrenzen für die **„metrikbasierte“ Segmentierung**. Für jeden Merkmalsatz ALLF, MFCC und NGRAM zeigt eine Tabelle *precision* (PRC), *recall* (RCL) und *F-measure* (F) für die Segmentierungen metric4, metric10 und metric16 aufgeschlüsselt nach der Art der Audiodaten.

ALLF

	svm10			svm30			svm50		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.21	0.45	**0.29**	0.28	0.50	**0.36**	0.30	0.53	**0.39**
jazz	0.20	0.27	**0.23**	0.27	0.37	**0.31**	0.29	0.39	**0.33**
oldies	0.46	0.21	**0.29**	0.51	0.33	**0.40**	0.50	0.37	**0.43**
pop	0.32	0.20	**0.25**	0.34	0.30	**0.32**	0.39	0.31	**0.35**
radio	0.51	0.42	**0.46**	0.49	0.49	**0.49**	0.54	0.50	**0.52**
reggae	0.60	0.20	**0.30**	0.45	0.22	**0.30**	0.37	0.25	**0.30**
rock	0.29	0.22	**0.25**	0.30	0.32	**0.31**	0.30	0.32	**0.31**
ø	0.37	0.28	**0.30**	0.38	0.36	**0.36**	0.38	0.38	**0.38**

MFCC

	svm10			svm30			svm50		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.22	0.53	**0.31**	0.26	0.49	**0.34**	0.28	0.52	**0.36**
jazz	0.23	0.45	**0.30**	0.30	0.50	**0.37**	0.31	0.47	**0.37**
oldies	0.38	0.48	**0.42**	0.42	0.47	**0.44**	0.51	0.48	**0.49**
pop	0.29	0.42	**0.35**	0.32	0.38	**0.35**	0.37	0.40	**0.38**
radio	0.55	0.57	**0.56**	0.59	0.56	**0.57**	0.59	0.57	**0.58**
reggae	0.30	0.41	**0.35**	0.30	0.36	**0.33**	0.32	0.37	**0.34**
rock	0.30	0.51	**0.38**	0.33	0.49	**0.40**	0.34	0.46	**0.39**
ø	0.32	0.48	**0.38**	0.36	0.46	**0.40**	0.39	0.47	**0.42**

NGRAM

	svm10			svm30			svm50		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.23	0.61	**0.34**	0.26	0.55	**0.35**	0.33	0.60	**0.43**
jazz	0.23	0.48	**0.31**	0.30	0.52	**0.38**	0.35	0.54	**0.42**
oldies	0.38	0.50	**0.43**	0.44	0.56	**0.50**	0.46	0.53	**0.49**
pop	0.28	0.40	**0.33**	0.35	0.46	**0.40**	0.38	0.45	**0.41**
radio	0.51	0.58	**0.55**	0.64	0.57	**0.60**	0.68	0.58	**0.63**
reggae	0.30	0.40	**0.34**	0.36	0.48	**0.41**	0.39	0.49	**0.43**
rock	0.29	0.54	**0.37**	0.33	0.52	**0.40**	0.34	0.51	**0.41**
ø	0.32	0.50	**0.38**	0.38	0.52	**0.43**	0.42	0.53	**0.46**

Tabelle 5.2.: Evaluation der Segmentgrenzen für die **SVM Segmentierung**. Für jeden Merkmalsatz ALLF, MFCC und NGRAM zeigt eine Tabelle *precision* (PRC), *recall* (RCL) und *F-measure* (F) für die Segmentierungen svm10, svm30 und svm50 aufgeschlüsselt nach der Art der Audiodaten.

ALLF

	mpck0			mpck100			mpck200		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.20	0.60	**0.30**	0.20	0.58	**0.30**	0.22	0.61	**0.32**
jazz	0.16	0.57	**0.25**	0.15	0.53	**0.24**	0.16	0.52	**0.25**
oldies	0.24	0.49	**0.32**	0.25	0.48	**0.33**	0.27	0.51	**0.35**
pop	0.20	0.50	**0.28**	0.16	0.43	**0.24**	0.17	0.44	**0.24**
radio	0.19	0.59	**0.29**	0.21	0.62	**0.32**	0.21	0.61	**0.32**
reggae	0.17	0.45	**0.25**	0.18	0.50	**0.26**	0.18	0.49	**0.26**
rock	0.20	0.55	**0.30**	0.20	0.57	**0.30**	0.20	0.53	**0.29**
ø	0.19	0.54	**0.28**	0.19	0.53	**0.28**	0.20	0.53	**0.29**

MFCC

	mpck0			mpck100			mpck200		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.19	0.58	**0.29**	0.18	0.54	**0.27**	0.19	0.55	**0.28**
jazz	0.16	0.58	**0.25**	0.17	0.58	**0.26**	0.17	0.56	**0.26**
oldies	0.25	0.57	**0.35**	0.25	0.59	**0.35**	0.30	0.60	**0.40**
pop	0.18	0.43	**0.25**	0.18	0.41	**0.25**	0.20	0.47	**0.28**
radio	0.21	0.61	**0.32**	0.23	0.65	**0.34**	0.23	0.64	**0.34**
reggae	0.20	0.53	**0.29**	0.19	0.52	**0.28**	0.21	0.54	**0.30**
rock	0.19	0.59	**0.29**	0.19	0.57	**0.29**	0.21	0.58	**0.30**
ø	0.20	0.56	**0.29**	0.20	0.55	**0.29**	0.22	0.56	**0.31**

NGRAM

	mpck0			mpck100			mpck200		
	PRC	RCL	**F**	PRC	RCL	**F**	PRC	RCL	**F**
classic	0.19	0.55	**0.29**	0.20	0.55	**0.30**	0.22	0.57	**0.32**
jazz	0.15	0.52	**0.24**	0.19	0.56	**0.28**	0.18	0.51	**0.26**
oldies	0.25	0.51	**0.34**	0.27	0.53	**0.36**	0.29	0.49	**0.36**
pop	0.23	0.58	**0.33**	0.24	0.59	**0.34**	0.21	0.49	**0.30**
radio	0.24	0.59	**0.34**	0.25	0.62	**0.35**	0.26	0.64	**0.37**
reggae	0.18	0.49	**0.26**	0.21	0.49	**0.30**	0.21	0.47	**0.29**
rock	0.20	0.55	**0.29**	0.24	0.61	**0.35**	0.23	0.59	**0.33**
ø	0.21	0.54	**0.30**	0.23	0.56	**0.33**	0.23	0.54	**0.32**

Tabelle 5.3.: Evaluation der Segmentgrenzen für die **Constrained clustering Segmentierung**. Für jeden Merkmalsatz ALLF, MFCC und NGRAM zeigt eine Tabelle *precision* (PRC), *recall* (RCL) und *F-measure* (F) für die Segmentierungen mpck0, mpck100 und mpck200 aufgeschlüsselt nach der Art der Audiodaten.

	ALLF			MFCC			NGRAM		
+/- s	0.5	1.0	2.0	0.5	1.0	2.0	0.5	1.0	2.0
metric4	0.29	0.37	0.46	0.30	0.37	0.44	0.29	0.37	0.45
metric10	0.30	0.39	0.48	0.30	0.38	0.48	0.31	0.39	0.49
metric16	0.31	0.39	0.48	0.32	0.39	0.48	0.31	0.40	0.48
svm10	0.29	0.34	0.41	0.33	0.42	0.51	0.33	0.42	0.51
svm30	0.32	0.39	0.49	0.35	0.44	0.55	0.38	0.47	0.56
svm50	0.34	0.42	0.52	0.36	0.46	0.57	0.40	0.50	0.60
mpck0	0.20	0.29	0.39	0.21	0.30	0.39	0.22	0.31	0.40
mpck100	0.21	0.29	0.40	0.22	0.30	0.40	0.24	0.33	0.43
mpck200	0.21	0.30	0.40	0.23	0.32	0.41	0.24	0.33	0.43

Tabelle 5.4.: Evaluation der Segmentgrenzen für alle Verfahren und Merkmalsätze in Abhängigkeit von der Evaluationstoleranz. Die Tabelle zeigt jeweils das *F-measure* der Evaluation über alle Audiodaten.

	ALLF			MFCC			NGRAM		
	svm10	svm30	svm50	svm10	svm30	svm50	svm10	svm30	svm50
classic	0.46	0.63	0.67	0.50	0.61	0.66	0.51	0.64	0.71
jazz	0.54	0.66	0.69	0.69	0.74	0.74	0.68	0.75	0.79
oldies	0.48	0.58	0.64	0.64	0.71	0.71	0.65	0.75	0.78
pop	0.49	0.63	0.68	0.66	0.69	0.71	0.65	0.74	0.78
radio	0.72	0.79	0.82	0.85	0.88	0.88	0.85	0.89	0.91
reggae	0.51	0.58	0.61	0.67	0.69	0.69	0.67	0.77	0.78
rock	0.41	0.55	0.57	0.61	0.63	0.63	0.63	0.68	0.72
ø	0.52	0.63	0.67	0.66	0.71	0.72	0.66	0.75	0.78

Tabelle 5.5.: Evaluation der Segmentmarkierungen für die **SVM Segmentierung**. Für jeden Merkmalsatz und die Segmentierungen svm10, svm30 und svm50 zeigt die Tabelle das *F-measure* der Evaluation aufgeschlüsselt nach der Art der Audiodaten.

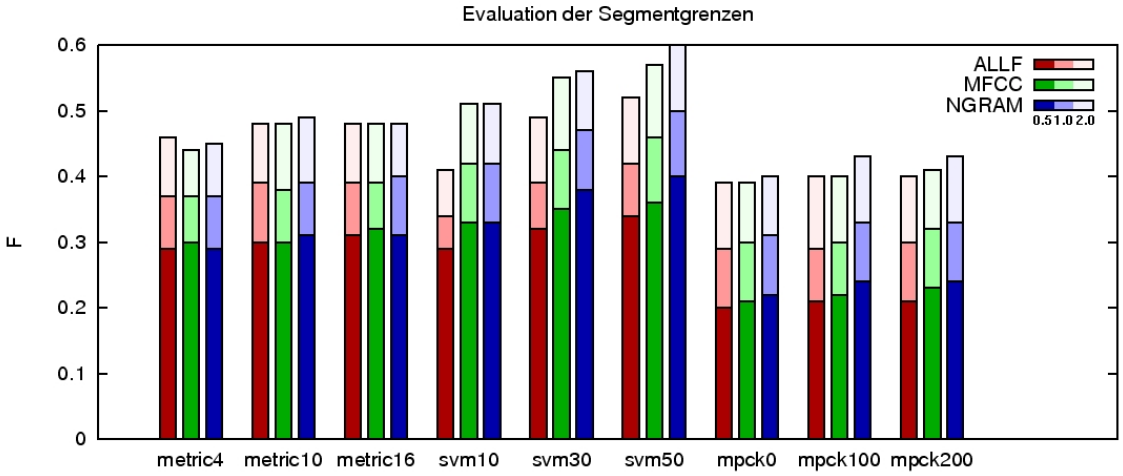


Abbildung 5.4.: Evaluation der Segmentgrenzen. Die Ergebnisse für Toleranzen von 0.5s, 1s und 2s werden in unterschiedlichen Farbhelligkeitsstufen dargestellt.

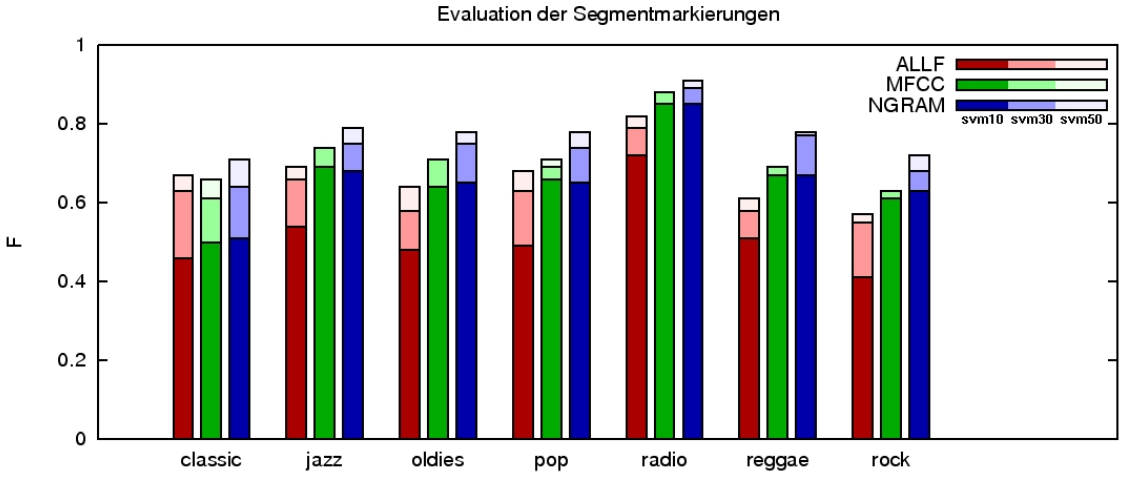


Abbildung 5.5.: Evaluation der Segmentmarkierungen. Die Ergebnisse für die Segmentierungen svm10, svm30 und svm50 werden in unterschiedlichen Farbhelligkeitsstufen dargestellt.

INPUT: true segmentation T, hypothesized segmentation H
OUTPUT: sum of segment section lengths correctly labeled l^c
INITIALIZATION: s^t = first true segment, s^h = first hypothesized segment, $l^c = 0.0$
$EvaluateLabels(T, H)$
(1) DO
(2) IF s^t and s^h overlap AND $s^t.label = s^h.label$ THEN
(3) $l^c = l^c$+ length of overlap
(4) IF $s^t.end < s^h.end$
(5) s^t = next true segment or null
(6) ELSE
(7) s^h = next hypothesized segment or null
(8) WHILE $s^t \neq$ null AND $s^h \neq$ null
(9) RETURN l^c

Für die Evaluation der Segmentmarkierungen ergeben sich PRC, RCL und F dann zu:

$$PRC = l^c/l^h \qquad RCL = l^c/l^t \qquad F = \frac{2*PRC*RCL}{PRC+RCL}$$

Da $l^t \approx l^h$, gilt hier $PRC \approx RCL \approx F$. Dieser Wert multipliziert mit 100 gibt an, zu wieviel Prozent die Segmentierungen in ihren Markierungen übereinstimmen, und kann als Akkurazität betrachtet werden.

5.3.3. Auswertung und Diskussion

Zur Evaluation der Segmentgrenzen Die **SVM Segmentierung** lieferte von den drei realisierten Verfahren insgesamt das beste Ergebnis. Für eine Evaluationstoleranz von 1s liegen die F-Werte des Gesamtergebnisses im Bereich von 0,34 bis 0,50. Schon für eine Trainingsmengengröße von 10% (svm10) schlägt das überwachte Lernverfahren, außer im Falle des Merkmalsatzes ALLF, auf dem es verglichen mit den anderen beiden Merkmalsätzen die schlechteste Leistung erzielte, die anderen Verfahren (Tab. 5.4). Wie zu erwarten war, steigt mit der Größe der Trainingsmenge auch die Güte des Ergebnisses. Es fällt dabei auf, dass für die Merkmalsätze MFCC und NGRAM schon bei der kleinsten Trainingsmenge die RCL-Werte relativ gut sind, d. h. mit der Anzahl der Trainingsbeispiele steigt vor allem die Genauigkeit. Ein besonders gutes Ergebnis ergab die Segmentierung der „radio“-Daten, was jedoch nicht verwundert, denn dabei entsprechen die Segmente größtenteils unterschiedlichen Klangklassen wie Musik oder einzelnen Sprechern, deren Trennung eine deutlich leichtere Aufgabe ist, als die Unterscheidung von Segmenten bei Musik (Tab. 5.2). Die **„metrikbasierte“ Segmentierung** erreichte mit einem F-Wertebereich von 0,37 bis 0,40, wieder für eine Evaluationstoleranz von 1s, ein mittleres Ergebnis; lediglich für den

Merkmalsatz ALLF reichen die erzielten Werte an die der SVM Segmentierung heran. Vergleicht man die Zahlen der verschiedenen Merkmalsätze, so sieht man, dass dieses Verfahren für alle drei ähnliche Ergebnisse erzielt, was darauf hindeutet, dass die Segmentgrenzenbestimmung durch den lokalen Vergleich der statistischen Eigenschaften von Merkmalsausprägungen von der Auswahl der Merkmale unabhängiger ist als es beim Lernen eines globalen Modells der Fall ist (Tab. 5.4). Weiter ist zu beobachten, dass bei zunehmender Fenstergröße (metric4 bis metric16) die PRC-Werte deutlich zunehmen, jedoch auf Kosten der RCL-Werte, d. h. die Fehlalarmrate fällt, doch es werden auch immer weniger korrekte Segmentgrenzen erkannt. Dieses gegenläufige Verhalten zeigt die parameterabhängigkeit des Kompromisses zwischen Genauigkeit und Sensitivität. Betrachtet man die F-Werte der verschiedenen Musikgenres bzw. Audiodatenarten für unterschiedliche Fenstergrößen, so fällt weiter auf, dass sich manche mit grossen, manche mit mittleren und manche mit kleinen Fenstergrößen besser segmentieren lassen. Dies zeigt, dass ein von solchen einzustellenden, den Segmentierungsprozess direkt beeinflussenden Parametern freies Verfahren von Vorteil wäre (Tab. 5.1).
Das schlechteste Ergebnis erzielte die **Constrained clustering Segmentierung**, bei welcher sich die F-Werte im Bereich von 0,29 bis 0,33 bewegen. Dabei ist eine leichte Verbesserung der F-Werte mit zunehmender Anzahl von Nebenbedingungen zu beobachten (Tab. 5.4). Dies gilt jedoch nur für das Gesamtergebnis über alle Audiodaten, da man in Tab. 5.3 sieht, dass eine gesteigerte Anzahl von Nebenbedingungen das Ergebnis nicht zwangsweise verbessert, vor allem an dem NGRAM Merkmalsatz. Auch sieht man dort, dass die RCL-Werte durchweg relativ gut sind, was jedoch an der häufig hohen Anzahl von erzeugten Segmenten liegt. Dies zeigt, dass die Beschreibung der Charakteristik eines Segments durch einen Cluster-Centroiden nicht ausreichend ist, was vor allem für musikalische Audiodaten gilt, da dort die Klangunterschiede zwischen verschiedenen Segmenten häufig geringer sind, als es bei der Unterscheidung von grundlegenden Klangklassen (wie z. B. Musik, Sprache, Geräusche) der Fall ist.
Eine Betrachtung aller Resultate unter verschiedenen Evaluationstoleranzen ergibt, dass eine Verdopplung der Toleranz, unabhängig vom Verfahren, eine Verbesserung der F-Werte durchschnittlich um geschätzte 0,09 zur Folge hat.

Zur Evaluation der Segmentmarkierungen Tab. 5.5 zeigt, dass die Klassifikation der Abschnitte einer Audiodatei, welche die SVM Segmentierung lieferte, deutlich besser funktioniert, als die Bestimmung der Segmentgrenzen, wobei die exakte Erkennung der Zeitpunkte der Übergänge zwischen den Segmenten natürlich eine deutlich schwierigere Aufgabe darstellt. Gerade wenn es um die Unterscheidung von grundlegenden Klangklassen geht, werden besonders hohe Werte erreicht. Bei Verwendung des Merkmalsatzes MFCC oder NGRAM werden im Falle der „radio“ Audiodaten schon für einen 10%igen Trainingsdatensatz im

Schnitt 85% einer Audiodatei korrekt klassifiziert. Insgesamt erreicht die Evaluation für den NGRAM Merkmalsatz auch hier die besten Werte: Für eine Trainingsmengengröße von 30% (svm30) ist das durchschnittliche Ergebnis besser, als die Ergebnisse der anderen Merkmalsätze für die größte Trainingsmenge.

Zu den Merkmalen In den Resultaten zeichnet sich ab, dass der Miteinbezug der zeitlichen Entwicklung eines Spektrums, was bei dem Merkmalsatz **NGRAM** einfach durch die Bildung von N-grammen erfolgte, sich positiv auf das Segmentierungsergebnis auswirken könnte. In den allermeisten Fällen sind die Werte für diesen Merkmalsatz vergleichbar oder besser, als jene für die anderen Merkmalsätze ALLF und MFCC. Es müßte jedoch noch überprüft werden, inwieweit sich die veränderten Parameter (wie Fenstergröße und Anzahl der MFCC) bei der Merkmalsextraktion auf das Ergebnis ausgewirkt haben.
Die SVM Segmentierung zeigte von den drei Verfahren die stärkste Abhängigkeit von der Wahl der Merkmale; hier schneidet der Merkmalsatz „ALLF“ am schlechtesten ab, und der Unterschied zu den Ergebnissen für den Merkmalsatz NGRAM ist deutlich. Die Verfahren der Constrained clustering und „metrikbasierten“ Segmentierung weisen, vor allem für die Merkmalsätze MFCC und ALLF, nur eine leichte bis kaum eine Abhängigkeit auf.

Vergleich mit anderen Verfahren

Bei dem Versuch des Vergleichs mit anderen Lösungen fiel auf, dass etliche Evaluationen aufgrund der Methodik oder des Maßes gar nicht oder zumindest nicht unmittelbar mit der in dieser Arbeit durchgeführten vergleichbar sind. Teilweise erfolgten sie nicht durch Zählung der korrekten Segmentgrenzen unter Verwendung einer Evaluationstoleranz, sondern z. B. durch Aufsummierung von Segmentfehlüberlappungen ohne Beachtung von Markierungen [ANS+05] oder durch manuelle subjektive Bewertung der berechneten Segmentgrenzen [DKW99]. In [CV03] wurde, wie in dieser Arbeit bei der Evaluation der Segmentmarkierungen (Tab. 5.5) auch, ein Maß für die Summe der Länge korrekt klassifizierter Abschnitte benutzt, in welches jedoch noch die Anzahl der Klassen, die dem Leser unbekannt bleibt, miteinbezogen wurde, was einen Vergleich der Ergebnisse schwierig macht. [TC99, JLC03] verwendeten eine andere Experimentiermethodik, bei welcher dem System die Anzahl der zu erzeugenden Segmente vorgegeben und als Ergebnis die Zahl der korrekt erkannten Segmente angegeben wurde. In [LZL03] haben sich die Autoren auf die Klassifikation von Merkmalsausprägungsvektoren beschränkt, ohne eine Segmentierung zu berechnen und eine Evaluation der Segmentgrenzen durchzuführen.
Ergebnisse von vergleichbaren Evaluationen der Segmentgrenzen von Verfahren, welche auf Sprachaudiodaten getestet wurden, zeigt Tab. 5.6. Man sieht, dass diese mit F-Werten von zum Teil fast 0,9 deutlich bessere Werte erreicht haben, als die in dieser Arbeit evaluierten Verfahren. Am erfolgreichsten war die SVM Segmentierung, deren beste Werte sich je nach

Arbeit	Audiodatenart	Umfang/h	+/- Toleranz/s	Evaluationsergebnis PRC	RCL	**F**
[WH06]	Radionachrichten	9	0,5	0,84	0,89	**0,86**
[CVR05]	Telefongespräche	17	0,3	0,54	0,76	**0,63**
	Radionachrichten	1	0,5	0,89	0,90	**0,89**
[OCR05]	Radionachrichten	5,5	1,0	0,95	0,83	**0,89**
[CW04]	Radionachrichten	5	1,0	0,80	0,85	**0,82**
[KSWW00]	Fernsehnachrichten	1	1,5	0,93	0,67	**0,78**

Tabelle 5.6.: Evaluationen der Segmentgrenzen anderer Verfahren

Größe der Trainingsmenge zwischen 0,56 und 0,63 für die „radio“ Audiodaten und eine Evaluationstoleranz von 1s bewegen.
Was Evaluationen von Verfahren zur Segmentierung musikalischer Audiodaten angeht, so sind mir nur die oben schon erwähnten Arbeiten [TC99, JLC03] bekannt. Dort kamen die Autoren zu dem Ergebnis, dass bis zu 63% aller Segmente, über deren Existenz sich eine Mehrheit von manuellen Segmentierern einig waren, mit einer Toleranz von +/- 0,5s erkannt werden, wenn dem System vorher die Anzahl der zu erzeugenden Segmente mitgeteilt wird. Bei Annahme von kontinuierlichen Segmentierungen entspräche dies in etwa einem RCL-Wert von 0,68, da eine Segmentierung von s Segmenten dann $s+1$ Segmentgrenzen enthält. Mit RCL-Werten von 0,50 bis 0,53 kommt die SVM Segmentierung auch hier nicht an das Ergebnis heran; anzumerken ist jedoch, dass der Umfang des Evaluationsdatensatzes mit 10 Liedfragmenten von je 1 Minute Länge deutlich geringer war.
Die sehr guten Klassifikationsresultate aus [LZL03] lassen sich am ehesten mit der Evaluation der Segmentmarkierungen (Tab. 5.5) für die „radio“ Audiodaten vergleichen. Diese werden dort mit F-Werten von 0,85 bis 0,91, die in diesem Fall der Akkurazität entsprechen, bestätigt und zeigen, dass Support Vector Machines zur Unterscheidung von grundlegenden Klangklassen gut geeignet sind.

Fazit

Was die Berechnung der Segmentgrenzen angeht, so erreichen die in dieser Arbeit implementierten und evaluierten Verfahren auf den „radio“ Audiodaten nicht die guten Werte anderer unüberwachter Lösungen auf vergleichbaren Audiodaten. Es bleibt aber die Frage offen, welche Ergebnisse die anderen Verfahren auf meinem Datensatz, der ja zu einem großen Teil aus Musikaudiodaten besteht, bzw. die von mir implementierten Verfahren auf den anderen Sprachaudiodatensätzen erzielen würden. Die Positionen von Segmentgrenzen in musikalischen Audiodaten sind aufgrund der Komplexität musikalischer Klänge und Strukturen

und der unterschiedlichen Elemente eines Liedes, über die man die Struktur definieren kann (z. B. Gesang, Melodien, Instrumente, Rhythmus), oftmals nicht eindeutig zu bestimmen und deshalb schwierig zu evaluieren. Ein Vergleich der Ergebnisse für die Musikaudiodaten, welche auf jeden Fall verbesserungsbedürftig sind, ist leider nicht möglich, da mir dazu keine anderen aussagekräftigen Evaluationen vorliegen.
Die Ergebnisse der Übereinstimmung der Segmentmarkierungen sind dagegen deutlich besser. Sie bestätigen im Falle der „radio“ Audiodaten die sehr guten Ergebnisse anderer Arbeiten und zeigen, dass die Klassifikation der Abschnitte musikalischer Audiodateien mittels einer SVM ordentlich funktioniert.
Unabhängig von den erzielten Ergebnissen ist für die SVM Segmentierung der Vorteil der Adaptivität zu nennen. Längere Segmente wie Instrumentensoli konnten damit ebenso identifiziert werden wie kurze Sprachabschnitte, und das ohne kritische Parameter wie Schwellenwerte oder Fenstergrößen anpassen zu müssen. In diesem Zusammenhang stellt lediglich die untere Grenze der Segmentlänge für die Ausreißerbehandlung eine Limitierung dar. Das „metrikbasierte“ Verfahren ist sehr schnell und liefert, besonders für große Fenster, eine oft plausible Segmentierung, was es für die Anwendung als Navigationshilfe durchaus geeignet macht. Zur Klassifikation der Segmente muß jedoch noch ein weiterer Verarbeitungsschritt erfolgen. Das Verfahren der Constrained clustering Segmentierung hat sich in dieser Form für die Segmentierung von Audiodaten leider als nicht besonders gut geeignet erwiesen.
Die datengestützte SVM Segmentierung erreicht bei der Evaluation der Segmentgrenzen über alle Audiodaten zwar signifikant bessere Ergebnisse als das unüberwachte Verfahren, betrachtet man allerdings die einzelnen Genres und wählt für die „metrikbasierte“ Segmentierung immer das beste Ergebnis der drei Fenstergrößen, dann fällt der Vergleich nicht mehr eindeutig zugunsten des überwachten Verfahrens aus. Lediglich bei den „radio“ Audiodaten weist die SVM Segmentierung stets deutlich bessere Resultate auf. Im Rahmen meiner Experimente läßt sich also sagen, dass der Lernaufwand sich durchaus bezahlt macht, wenn es darum geht, nach gut voneinander unterscheidbaren Klangbildern (z. B. Sprecher, Musik, Geräusche) zu segmentieren. Ob man durch eine vorherige manuelle Teilsegmentierung die schon recht guten Ergebnisse anderer unüberwachter Verfahren auf Radionachrichten und Telefongesprächen noch weiter verbessern kann, ist fraglich und müßte durch Evaluationen auf den entsprechenden Datensätzen nachgewiesen werden.

6. Zusammenfassung und Vorschläge für weitere Bemühungen

Inhalt dieser Arbeit ist zunächst eine Übersicht über bereits existierende Lösungsvorschläge zum Thema Segmentierung von Audiodaten und deren Methoden. Diese Verfahren wurden entweder für die Anwendung auf musikalischen Audiodaten, sprachlichen Audiodaten oder für die Unterscheidung von grundlegenden Klangklassen vorgesehen und werden je nach verwendeter Methode als „modellbasiert“, „sequenzbasiert“ oder „metrikbasiert“ kategorisiert. „Modellbasierte“ Verfahren bedienen sich dabei überwachter, die anderen unüberwachter Lernmethoden. Die Beschränkung der Verfahren auf jeweils eine Audiodatenart war Ansporn dazu, die Segmentierung mittels adaptiver Methoden aus dem Bereich des maschinellen Lernens zu untersuchen.
Es werden Grundlagen wie die Repräsentation von Audiodaten als Wertefolge, Audiomerkmale wie *Lautstärke*, *Zero Crossing Rate*, *Spectral Flux*, *MFCC*, *Spectral Centroid*, *Spectral Rolloff*, *Bandbreite*, *N-Gramme*, *Chroma*, *Zirkularität* und deren Extraktion erläutert. Um eine Vorstellung von der prinzipiellen Vorgehensweise der Verfahren aus den oben erwähnten Kategorien zu erlangen, werden beispielhaft Lösungen anderer Arbeiten näher beschrieben und die Nach- und Vorteile der jeweiligen Methoden genannt.
Die Eigenleistung besteht zum einen in der Implementierung dreier Segmentierungsverfahren mit Hilfe des dazu entwickelten Programms *Segfried* und durch Erweiterung des Clustering-plugin der Lernumgebung YALE. Diese Verfahren verwenden jeweils einen „metrikbasierten“ Ansatz durch gefensterte Berechnung einer Kriteriumsfunktion, Klassifikation mittels Support Vector Machines (SVM) oder eine Constrained clustering Methode. Zum anderen wurde ein Datensatz erstellt, welcher aus 100 Liedern und 160 Minuten Radioprogramm besteht und manuell erstellte Vergleichssegmentierungen enthält, und die realisierten Verfahren darauf evaluiert.
Die Evaluation beinhaltet die Beschreibung sowohl der extrahierten Merkmale als auch der Durchführung der Experimente und die Präsentation und Diskussion der Ergebnisse und wurde für die drei Verfahren im Hinblick auf die Segmentgrenzen, für das Klassifikationsverfahren mittels SVM zusätzlich hinsichtlich der Segmentmarkierungen durchgeführt. Die Resultate lassen sich wie folgt zusammenfassen:

- das „metrikbasierte“ Verfahren ist schnell und liefert für große Fenstergrößen oft plausible Segmentierungen
- die SVM Segmentierung erreicht insgesamt die besten Ergebnisse und ist besonders erfolgreich, wenn es um die Unterscheidung grundlegender Klangbilder geht
- das Constrained clustering Verfahren hat sich in dieser Form als eher ungeeignet erwiesen
- die guten Ergebnisse anderer unüberwachter Verfahren, die jedoch nur auf sprachlichen Audiodaten (Radionachrichten, Telefongespräche) evaluiert wurden, werden nicht erreicht
- die Verfahren liefern für den NGRAM Merkmalsatz fast durchweg bessere Werte, als für die anderen Merkmalsätze

6.1. Und nun ?

Zunächst einmal wäre es interessant zu erfahren, welche Ergebnisse die in dieser Arbeit vorgestellten Verfahren auf anderen Datensätzen wie [FGP+98] und [PM01] erzielen. Dazu müßte *Segfried* lediglich um einige Importfunktionalitäten erweitert werden.
Was die Weiterentwicklung der Verfahren betrifft, so könnte man in der Vorverarbeitung mit verschiedenen Fenstergrößen bei der Merkmalsextraktion und dem N-Gramm-Merkmal experimentieren, denn es ist ja noch die Frage offen, ob beim NGRAM Merkmalsatz das größere Extraktionsfenster oder die N-Gramm-Bildung dem besseren Ergebnis zu Grunde liegt. Sollte sich letzteres herausstellen, so könnte es sich durchaus lohnen, den N-Gramm-Ansatz weiter zu verfolgen. Ebenso könnte sich die Kombination von Merkmalsausprägungen, welche aus unterschiedlich großen Fenstern berechnet werden, um, in Analogie zur Wavelet-Transformation, eine Charakterisierung des Audiomaterials in verschiedenen Skalierungen zu erhalten, als vorteilhaft erweisen.
Aber auch durch weitere Nachverarbeitungsschritte kann man versuchen, das Segmentierungsergebnis zu verbessern. Naheliegend wäre z. B. , die Merkmalsausprägungsteilfolgen benachbarter Segmente mittels eines Distanzmaßes zu vergleichen und die ähnlichsten Segmente miteinander zu verschmelzen, wie es bei einigen Arbeiten schon gemacht wurde. Auch könnte man versuchen, jede Segmentgrenze genauer zu positionieren, indem man ihre Umgebung näher untersucht. Solche Maßnahmen haben aber natürlich den Nachteil, dass sie wieder Schwellenwerte verwenden bzw. heuristisch sind, lediglich nachbessern können und die Laufzeit verlängern.
Da die automatische Segmentierung von sprachlichen Audiodaten bereits recht gut funktioniert, wäre mein Vorschlag, sich verstärkt mit Musikaudiodaten und der Entdeckung

wiederholt auftretender ähnlicher Sequenzen zu beschäftigen und entsprechende Verfahren aussagekräftig zu evaluieren, um dann auf eine Kombination mit klangbildorientierten Segmentierungsverfahren hinzuarbeiten.

A. Mathematische Grundlagen

Nachfolgend möchte ich für diejenigen, die mit dem einen oder anderen Begriff nicht vertraut sind, kurz grundlegende Konzepte und Methoden erläutern, denen wir begegnen, wenn wir uns mit Arbeiten zu dem Thema der Strukturierung von Audiodaten befassen. Quelle der nachfolgenden Informationen ist, sofern nicht anders angegeben, [WIK07] und das *world wide web*.

A.1. Statistik

Methoden der Statistik werden benutzt, um mit Hilfe von Beobachtungen Modelle von Prozessen zu erstellen, die die statistischen Eigenschaften, also die Verteilung im Wahrscheinlichkeitsraum dieser Beobachtungen widerspiegeln. In unserem Fall sind es akustische Merkmalausprägungen (siehe Abschnitt 2.1) bzw. Vektoren ebendieser, deren Eigenschaften modelliert werden.

Wahrscheinlichkeit und likelihood In der Wahrscheinlichkeitstheorie werden marginale Wahrscheinlichkeit $P(A)$, bedingte Wahrscheinlichkeit $P(A|B)$ und Verbundwahrscheinlichkeit $P(A, B)$ für Ereignisse A, B unterschieden. Die Wahrscheinlichkeiten von Ereignissen werden von Wahrscheinlichkeitsverteilungen beschrieben, welche im diskreten Fall durch Wahrscheinlichkeitsfunktionen, im kontinuierlichen Fall durch Wahrscheinlichkeitsdichtefunktionen (Probability Density Function, PDF) modelliert werden.
Eine marginale Wahrscheinlichkeit, die sich auf eine Zufallsvariable bezieht und wobei Vorwissen benutzt wird, welches nicht auf Beobachtungen von Ereignissen basiert, wird in der Praxis auch als *a priori* Wahrscheinlichkeit bezeichnet. Eine*a posteriori* Wahrscheinlichkeit ist dagegen eine bedingte Wahrscheinlichkeit, die sich auf eine Zufallsvariable in Abhängigkeit von einer gegebenen Beobachtung (z. B. einer anderen Zufallsvariablen) bezieht.
Likelihood, gegeben durch eine Likelihood-Funktion, kann man als eine andere Sicht auf eine bedingte Wahrscheinlichkeit $P(A|B)$ ansehen. Während eine Wahrscheinlichkeitsdichtefunktion für $P(A|B)$ eine Funktion von A ist, ist eine Likelihood-Funktion eine Funktion von B. Jede zu einer gegebenen Likelihood-Funktion für $P(A|B)$ proportionale Funktion ist dabei ebenfalls eine Likelihood-Funktion für $P(A|B)$. Es gilt:

$$L(b|A) = \alpha \cdot P(A|B = b), \forall \alpha > 0$$

Während mit P von b auf A geschlossen wird, erfolgt mit L der Umkehrschluß von A auf b. Praktisch gesehen trifft man mit P Vorhersagen über A, und mit L schließt man auf die Voraussetzungen, die zur Beobachtung von A geführt haben.
In dem *Satz von Bayes* werden marginale und bedingte Wahrscheinlichkeit miteinander verknüpft:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Dabei sind sowohl Funktionen für $P(B|A)$ als auch für $\frac{P(B|A)}{P(B)}$ Likelihood-Funktionen für $P(A|B)$.

Kovarianz Die Kovarianz zweier Zufallsvariablen X, Y gibt an, inwiefern die Abweichungen von ihren Erwartungswerten korrelieren, und ist definiert als:

$$Cov(X,Y) = E((X - E(X)) \cdot (Y - E(Y)))$$

Mit Hilfe der *Stichproben-Kovarianz* läßt sich die Kovarianz für Stichproben mit n Elementen schätzen zu

$$Cov(X,Y) = \sum_{i=1}^{n} \frac{(x_i - \bar{x}) \cdot (y_i - \bar{y})}{n}$$

wobei $\bar{x}, \bar{y}$ die Mittelwerte der jeweiligen Stichproben sind. Eine Kovarianzmatrix ist die mehrdimensionale Entsprechung des Varianzbegriffes für Vektoren von Zufallsvariablen, deren Elemente die Kovarianzen von je zwei der Zufallsvariablen sind.

Gaussian Mixture Model (GMM) Ein Mixture Model im Allgemeinen ist eine gewichtete Summe von parametrisierten Funktionen, die eine Wahrscheinlichkeitsverteilung einer Zufallsvariable X darstellen. Im Falle eines GMM handelt es sich bei den Funktionen um gaußsche Normalverteilungen $\mathcal{N}(\mu, \sigma)$:

$$p_X(x) = \sum_{k=1}^{K} a_k \cdot \mathcal{N}(x|\mu_k, \sigma_k) = \sum_{k=1}^{K} a_k \cdot \frac{1}{\sigma_k \sqrt{2\pi}} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$$

mit $0 < a_k < 1$ und $a_1 + \ldots + a_K = 1$.

Principal Components Analysis (PCA) Die Hauptkomponentenanalyse, oder auch *Principal Components Analysis (PCA)*, ist eine orthogonale lineare Transformation, welche für einen multidimensionalen Datensatz durch Überführung in einen Vektorraum mit neuer Basis die Korrelation zwischen den Dimensionen minimiert (Hauptachsentransformation). Dies geschieht anschaulich durch Rotation der Hauptachsen, so dass jeweils die Varianz der Daten entlang der Achsen maximiert und absteigend von der ersten zur letzten Dimension geringer wird. Die PCA wird häufig als Extraktionsmethode bei der *Faktorenanalyse* (Bündelung von mehreren Variablen zu „Faktoren“) verwendet.

Technisch beinhaltet dies die Zentrierung der Daten bezüglich ihrer Erwartungswerte und die Berechnung der Eigenvektoren und -werte der Kovarianzmatrix. Die Transformationsmatrix wird dann aus den nach der Größe ihrer Eigenwerte absteigend sortierten Eigenvektoren gebildet, so dass die Varianz der transformierten Daten jeweils den Eigenwerten entspricht und somit von Dimension zu Dimension abnimmt.
Die PCA wird zur Dimensionsreduktion bzw. Vereinfachung von multidimensionalen Datensätzen eingesetzt. Betrachtet man nämlich die Varianz von Daten als Maß für ihren Informationsgehalt, so kann man nach einer PCA die letzten Dimensionen, welche am wenigsten zur Gesamtvarianz der Daten beitragen, weglassen, ohne den Informationsgehalt zu sehr zu verringern. Der Nachteil dieses Verfahrens liegt darin, dass die Daten bei der Transformation häufig ihre inhaltliche Interpretierbarkeit verlieren.

Log Likelihood Ratio (LLR) Ein statistischer *Likelihood Ratio* Test liefert ein Maß für die Wahrscheinlichkeit der Korrektheit einer Hypothese H_0 über einen Datensatz X im Vergleich zu einer anderen Hypothese H_1. Sind $P(H_1), P(H_0)$ die a priori Wahrscheinlichkeiten der beiden Hypothesen und $p(H_0|X), p(H_1|X)$ jeweils die Likelihood-Werte der Daten unter den Hypothesen, dann wird H_0 als wahrscheinlicher betrachtet, wenn gilt:

$$\frac{p(H_0|X)}{p(H_1|X)} \geq \frac{P(H_1)}{P(H_0)}$$

Da $P(H_0), P(H_1)$ und die Likelihood-Funktionen oft nicht (exakt) bestimmbar sind, wird in der Praxis einfach ein vernünftig festzulegender Schwellenwert τ verwendet. Desweiteren ist es vorteilhaft, den Logarithmus dieses Quotienten zu betrachten, wenn sich die Hypothesen auf gaußsche Modelle beziehen:

$$\ln \frac{p(H_0|X)}{p(H_1|X)} = \ln p(H_0|X) - \ln p(H_1|X) \geq \tau$$

Der Ausdruck auf der linken Seite wird *Log Likelihood Ratio (LLR)* genannt, dessen Wert nach einer maximum likelihood Schätzung der Modellparameter für H_0 und H_1, die zur Maximierung von $p(H_0|X)$ und $p(H_1|X)$ führt, bestimmt wird [AMB04].

Bayesian Information Criterion (BIC) Der BIC-Wert eines Modells M für gegebene Daten X mit N Beobachtungen ist ein maximaler log Likelihood-Wert $p(X|M)$, welcher durch die Anzahl der freien Modellparameter $|M|$ bestraft wird:

$$BIC_M = \ln p(X|M) - \frac{|M|}{2} \ln N$$

Ein Vergleich zweier Modelle M_1, M_2 durch Differenzbildung führt zum sogenannten ΔBIC-Wert:

$$\Delta BIC = \ln p(X|M_1) - \ln p(X|M_2) - \frac{||M_2| - |M_1||}{2} \ln N$$

Dies ist prinzipiell ein LLR-Wert, welcher durch die Differenz der Anzahl freier Parameter der Modelle bestraft wird [AMB04].

A.2. Informatik

Klassifikation Das Problem der Klassifikation besteht darin, eine Menge von Objekten $x \in X$ auf eine Menge von Markierungen oder *labels* $y \in Y$, welche die unterschiedlichen Klassen benennen, abzubilden. Dieses wird in einem überwachten Verfahren mit Hilfe eines Klassifizierers bzw. Klassifikationsalgorithmus gelöst, welcher auf eine Menge von Trainingsbeispielen $(x_0, y_0), (x_1, y_1), \ldots$ zurückgreifen kann bzw. damit vorher trainiert wird. Beispiele für Klassifizierer sind K-Nearest-Neighbour, Künstliche Neuronale Netzwerke, Entscheidungsbäume und Support Vector Machines.

Clustering Clustering ist das Problem der Gruppierung von in zu definierender Weise ähnlichen Objekten, wobei die Anzahl und Art der zu bildenden Gruppen, also die Elemente und Kardinalität der Markierungsmenge Y, i. Allg. nicht bekannt ist, was den Unterschied zur Klassifikation ausmacht. Die Wahl des Ähnlichkeitsmaßes ist dabei natürlich von zentraler Bedeutung. Es existieren unüberwachte und halbüberwachte Clustering-Verfahren, wobei letztere sich z. B. Vorwissen darüber zunutze machen, welche Objekte zur selben und welche keinesfalls zur selben Gruppe gehören sollen (must-link- und cannot-link-constraints). Beispiele für Clustering-Verfahren sind centroid-basierte Verfahren wie K-Means und MPCK-Means, top-down und bottom-up Clustering.

Multi Layer Perceptron (MLP) Ein Multi Layer Perceptron ist ein besonderer Typ eines künstlichen neuronalen Netzwerkes (Artificial Neuronal Network, ANN), dessen Neuronen in mehreren Schichten strukturiert sind und jedes Neuron einer Schicht mit Neuronen der benachbarten Schichten verbunden ist. Oftmals verlaufen alle Verbindungen nur in eine Richtung, und zwar von der Eingabe-Schicht (input layer) über eine oder mehrere verborgene Schichten (hidden layers) hin zur Ausgabe-Schicht (output layer), was man dann als „feed-forward-Netzwerk“ bezeichnet. Der *Satz der universellen Approximation* besagt, dass ein MLP mit nur einer verborgenen Schicht jede Funktion reeller Zahlen beliebig genau approximieren kann. Um dies zu erreichen, muß es zunächst trainiert werden. Einer der bekanntesten Algorithmen dafür ist der *back propagation* Algorithmus.
In [AMB04] z. B. werden MPLs zur Klassifizierung eingesetzt, d. h. es wird ein MLP darauf trainiert, die a posteriori Wahrscheinlichkeit $P(q_k|x_i)$ zu schätzen, dass ein Merkmalausprägungsvektor x_i zu einer Klasse q_k gehört.

Hidden Markov Model (HMM) Markov Models werden verwendet, um „gedächtnislose“ Markov Prozesse zu modellieren, deren Parameter unbekannt sein können. In einem regulären Markov Model sind die Zustände direkt beobachtbar, in einem HMM nur indirekt über Ausgaben von Zufallsvariablen, die beim Erreichen eines Zustandes erfolgen. Das bedeutet, dass bei einem HMM nicht nur die Übergangswahrscheinlichkeiten zwischen den Zuständen, sondern auch die Ausgabe- oder Emissionswahrscheinlichkeiten zu den unbekannten Parametern gehören.
In der Anwendung von HMMs existieren drei Haupt-Problemkonstellationen:

- bei bekannten Parametern schätze die Wahrscheinlichkeit einer bestimmten Ausgabesequenz (siehe **Viterbi- und Forward-Algorithmus**)
- bei bekannten Parametern und einer gegebenen Ausgabesequenz finde die wahrscheinlichste Zustandsfolge (siehe **Viterbi- und Forward-Algorithmus**)
- für eine gegebene Ausgabesequenz schätze die unbekannten Parameter (siehe **E-M-Algorithmus**)

Eine häufige Verfahrensweise im Zusammenhang mit dem Thema der Segmentierung ist, dass bei vorher festgelegter HMM-Topologie mit Hilfe der akustischen Merkmalsausprägungen zunächst die Parameter geschätzt werden und dann die wahrscheinlichste Zustandsfolge berechnet wird.

E-M-Algorithmus Der Ausdruck „E-M-Algorithmus“ bezeichnet eine Klasse von Algorithmen, die zur Maximum-Likelihood-Schätzung der Parameter probabilistischer Modelle verwendet werden, wenn eine einfache, analytische Berechnung des Maximums der Likelihood-Funktion nicht möglich ist. Dabei wird davon ausgegangen, dass sowohl bekannte als auch fehlende oder verborgene Daten existieren. Die Beobachtungen X mit einer Verteilung $p(x|\theta)$ stellen die bekannten, aber unvollständigen Daten dar. Wenn der vollständige Datensatz mit $Z = (X, Y)$ bezeichnet wird, dann repräsentieren die Daten Y mit der Verteilung $p(y|\theta)$ die verborgenen Daten. θ ist der Parametersatz, den es zu schätzen gilt.
Die Verteilung $p(z|\theta)$ charakterisiert also unseren vollständigen Datensatz und kann auch geschrieben werden als

$$p(z|\theta) = p(x, y|\theta) = p(y|x, \theta) \cdot p(x|\theta)$$

Man definiert nun eine Likelihood-Funktion $L(\theta|Z) = L(\theta|X, Y) = p(X, Y|\theta)$, die likelihood der vollständigen Daten, welche aufgrund der fehlenden Daten Y eine Zufallsvariable ist. Weil man sie deshalb nicht maximieren kann, maximiert man im Verlauf des Algorithmus stattdessen ihren Erwartungswert.
Nachdem die Parameter θ des Modells mit zufälligen oder vernünftigen Werten initialisiert worden sind, wiederholt der Algorithmus dabei iterativ zwei Arbeitsschritte:

1. **E***stimation*-Schritt: Dieser beinhaltet die Berechnung einer Funktion $Q(\theta)$, welche für Parameterwerte θ aufgrund der aktuellen Parameterschätzung θ^i und der beobachteten Daten X den bedingten Erwartungswert der log likelihood der vollständigen Daten liefert:
$$Q(\theta) = E\left[\ln p(X, Y|\theta) \mid X, \theta^i\right]$$

2. **M***aximization*-Schritt: Hier wird die Funktion $Q(\theta)$ und damit der Erwartungswert maximiert und die entsprechenden Parameterwerte als neue Schätzung übernommen:
$$\theta^{i+1} = \operatorname*{argmax}_{\theta} Q(\theta)$$

Ein häufig verwendeter E-M-Algorithmus ist der *Baum-Welch-Algorithmus*, der den *Forward-Backward-Algorithmus* dazu benutzt, die unbekannten Parameter eines HMM zu berechnen [Bil97, Rab89].

Viterbi- und Forward-Algorithmus Viterbi- und Forward-Algorithmus basieren auf dynamischer Programmierung und sind in ihrer Vorgehensweise ähnlich [Rab89].
Der Forward-Algorithmus berechnet, ausgehend vom ersten Symbol o_1 der gegebenen Ausgabesequenz $O = o_1, o_2, \ldots, o_T$, iterativ die *Vorwärtswahrscheinlichkeiten*, für den Präfix $o_1, o_2, \ldots, o_t$ zum Zeitpunkt t im Zustand s_i zu sein und dann für $t = T$ als Summe über alle Zustände der Vorwärtswahrscheinlichkeiten die Wahrscheinlichkeit für die komplette Ausgabesequenz O. Analog dazu läßt sich auch eine rückwärts gerichtete Prozedur, der Backward-Algorithmus, definieren, welche entsprechend die Rückwärtswahrscheinlichkeiten berechnet.
Der Viterbi-Algorithmus dagegen berechnet iterativ die *größte* Vorwärtswahrscheinlichkeit, für den Präfix $o_1, o_2, \ldots, o_t$ zum Zeitpunkt t im Zustand s_i zu sein und merkt sich jeweils, welche Zustandssequenz $q_1, q_2, \ldots, q_t = s_i$ dahin geführt hat. Für $t = T$ erhält man so die wahrscheinlichste Zustandssequenz für ganz O.

Minimum Description Length (MDL) Minimum Description Length benennt ein Prinzip, nach welchem die beste Hypothese über das Zustandekommen von Daten diejenige ist, welche zu der größten Kompression der Daten führt, und kann als Formalisierung von *Occam's Razor* angesehen werden: „Entitäten sollten nicht über das Notwendige hinaus vermehrt werden“. Sinngemäß bedeutet dies, hat man mehrere Theorien oder Modelle, welche den selben Sachverhalt oder die selben Daten beschreiben, dann sollte man die einfachste Theorie oder das einfachste Modell auswählen. Das Wort „einfach“ ist hierbei nicht immer klar definierbar, im Falle von statistischen Modellen z. B. ist das einfachste Modell zumeist Jenes mit den wenigsten Parametern.

A.3. Akustik und Signalverarbeitung

Fourier Transformation und Spektrum eines Signals Allgemein betrachtet ist die Fourier Transformation eine Integraltransformation der Form $(Tf)(u) = \int_{t_1}^{t_2} f(t)K(t,u)dt$, welche einer Funktion f eine andere Funktion (Tf) zuordnet, welche man als Fouriertransformierte bezeichnet. K ist dabei eine sinusförmige Elementarschwingung der Frequenz u, weshalb man (Tf) auch als Zerlegung von f in Frequenzkomponenten betrachten kann.
Je nach Art der Funktion f existieren verschiedene Varianten der Fourier Transformation. Die Transformation einer zeit- und wertediskreten univariaten Wertefolge x der Länge N geschieht mit der sogenannten *Diskreten Fourier Transformation (DFT)*:

$$X_k^v = \sum_{j=0}^{N-1} x_j^v e^{-2\pi ijk/N}$$

für $k = 0, \ldots, N-1$. Die Ergebnisfolge X wird auch als *Spektrum* von x bezeichnet. Berechnet man die Folge der Beträge der komplexen Koeffizienten von X, also $|X_k^v|$, so erhält man das *Betragsspektrum*, welches von den Phaseninformationen abstrahiert, die oft nicht benötigt werden. Die Folge aus $2|X_k^v|$ wird auch als *Amplitudenspektrum* bezeichnet.
Ist x das Ergebnis einer äquidistanten Abtastung eines Zeitsignals, z. B. eines Audiosignals, dann ist die Differenz aller benachbarten Werte in der displacement Dimension von x eine Zeitkonstante T, der Kehrwert der Abtastfrequenz, und man kann nun den Zusammenhang der Koeffizienten mit bestimmten Frequenzen herstellen. Sind $F, T \in \mathbb{R}^{>0}$ mit $FT = \frac{1}{N}$ positive Zahlen, dann gilt mit $t_j = jT$ und $\omega_k = k(2\pi F)$ auch:

$$X_k^v = \sum_{j=0}^{N-1} x_j^v e^{-i\omega_k t_j}$$

F entspricht analog zu T der Frequenzauflösung in X, und dem Koeffizienten X_k^v wird die Frequenz kF für $k = 0, \ldots, \frac{N}{2} - 1$ bzw. $(k-N)F$ für $k = \frac{N}{2}, \ldots, N-1$ zugeordnet. Da das Spektrum für reellwertige Funktionen symmetrisch ist und die negativen Frequenzen in der Praxis kaum eine Rolle spielen, werden häufig nur die ersten $\frac{N}{2}$ Koeffizienten von X betrachtet.

Psychoakustische Frequenzskalen Da die physikalische Meßgröße der Frequenz eines Tones nicht proportional zur menschlichen Wahrnehmung der Tonhöhe ist, existieren Abbildungen der Hertz-Frequenzskala auf Skalen, die versuchen, den Zusammenhang zwischen Tönen und der Wahrnehmung ihrer Höhe annähernd zu linearisieren und auf empirischen Untersuchungsergebnissen beruhen, wobei ein „Ton“ eine einzelne sinusförmige Schwingung bezeichnet. Diese Ergebnisse zeigten, dass die Wahrnehmung der Tonhöhe sich logarithmisch verhält und nicht nur die Frequenz, sondern auch die Lautstärke und Dauer eines Tones, wenngleich nur geringen, Einfluß darauf haben. Die

SPINC-Skala (*Spectral Pitch Increment*) basiert auf der Bestimmung der „gerade wahrnehmbaren Frequenzänderung“: $\phi(f) = 1414 \arctan \frac{f}{1414}$

Mel-Skala (Stevens, 1937) resultiert aus Untersuchungen von Relationen zwischen Tönen bezüglich ihrer Höhe (Bestimmung der doppelten Tonhöhe, Tonhöhe mittig zwischen zwei Referenztönen): $M(f) = 1127,01 \ln(1 + \frac{f}{700})$

Bark-Skala (Zwickler, 1961) basiert auf der Bestimmung der „kritischen Bandbreiten“ auditiver Filter: $Z(f) = 13 \arctan(0,00076f) + 3,5 \arctan(\frac{f}{7500})^2$

ERB-Skala (*Equivalent Rectangular Bandwidth*) beruht auf der Umrechnung der kritischen Bandbreiten auditiver Filter in äquivalente Bandbreiten von Rechteckfiltern: $EZ(f) = 21,4 \log_1 0(0,00437f + 1)$

Singular Value Decomposition (SVD) Die Singulärwertzerlegung, oder *Singular Value Decomposition (SVD)*, einer Matrix $A \in \mathbb{C}^{m \times n}$ mit Rang r ist die Faktorisierung

$$A = U \Sigma V^*,$$

wobei $U \in \mathbb{C}^{m \times m}$, $V \in \mathbb{C}^{n \times n}$ unitäre Matrizen sind, und V^* die Adjungierte (komplex konjugierte der Transponierten) zu V ist. Die ersten r Einträge auf der Hauptdiagonalen der Matrix $\Sigma \in \mathbb{R}^{m \times n}$ sind die Singulärwerte $\sigma_1, \ldots, \sigma_r$, alle anderen Einträge von Σ sind 0. Die Singulärwerte charakterisieren, wie die Eigenwerte auch, Eigenschaften der Matrix A.
Die SVD existiert für beliebige Matrizen und zeigt, dass jede von einer Matrix durchgeführte Transformation als Rotation mit anschließender Skalierung und abschließender Rotation dargestellt werden kann. Sie ist mathematisch verwandt mit der PCA und wird z. B. in der Bildverarbeitung zur Rauschreduktion und Bildkompression eingesetzt.

Hough Transformation Die *Hough Transformation* ist ein Verfahren, welches ursprünglich für die Entdeckung von geraden Linien in Bildern entwickelt und dann zur Suche nach beliebigen parametrisierbaren geometrischen Objekten erweitert wurde.
Hat man eine Parametergleichung für das zu suchende Objekt, dann wird zur Erkennung ein Dualraum (Parameterraum oder Hough-Raum) erschaffen, in welchem jeder Punkt einem möglichen geometrischen Objekt im Bildraum entspricht. Für jeden zu untersuchenden Punkt (x_i, y_i) im Bildraum betrachtet man die Parametergleichung nun als Funktion, in welcher die Werte für x_i und y_i Konstanten und die Parameter die Variablen darstellen, und zeichnet diese im Parameterraum. Je mehr Funktionen sich dann in einem Punkt im Parameterraum schneiden, desto stärker ausgeprägt ist ein diesem Punkt entsprechenden Objekt im Bildraum.
Geraden werden z. B. durch ihre Hessesche Normalform

$$d = x \cos \alpha + y \sin \alpha$$

mit den Parametern Winkel des Normalenvektors α und Abstand zum Ursprung d repräsentiert. Der Parameterraum ist somit 2-dimensional, und für alle zu untersuchenden Punkte (x_i, y_i) im Bildraum wird die Funktion

$$D(\alpha) = x_i \cos\alpha + y_i \sin\alpha$$

im Parameterraum gezeichnet.

A.4. Distanz- und Ähnlichkeitsmaße

Distanz- und Ähnlichkeitsmaße sind Funktionen, welche Objekte auf einen numerischen Wert abbildet, der dann als Abstand bzw. Ähnlichkeit zwischen den Objekten interpretiert wird. Dabei verhalten sie sich meistens, jedoch nicht zwingend, in dualer Art und Weise, d. h. je größer die Ähnlichkeit, desto geringer die Distanz und umgekehrt. Während Distanzmaße nicht immer aber normalerweise die axiomatischen Bedingungen für Metriken erfüllen, gilt dies für Ähnlichkeitsmaße seltener.

Euklidsche Distanz Die euklidsche Distanz ist eine Ausprägung der L_p-Distanz (Minkowski-Metrik) für den Fall $p = 2$ und für uns Menschen als Länge einer geraden Verbindung zwischen zwei Punkten $x, y \in \mathbb{R}^n$ wohl die natürlichste Form der Abstandsmessung. Die L_p-Norm ist definiert als

$$d_{L_p}(x, y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} \ .$$

Cosinus-Distanz Der Abstand zweier Vektoren $x, y \in \mathbb{R}^n$ wird bei der Cosinus-Distanz über den Winkel, den sie einschließen, definiert:

$$d_{cos}(x, y) = \cos\alpha(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

Mahalanobis Distanz Die Mahalanobis Distanz zweier Vektoren $x, y \in \mathbb{R}^n$ ist eine mit der Kovarianzmatrix Σ einer n-dimensionalen Normalverteilung parametrisierte euklidsche Distanz, welche Korrelationen zwischen den Dimensionen miteinbezieht:

$$d_{mah}(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$

Kullback-Leibler Divergenz Die Kullback-Leibler Divergenz, auch *Relative Entropie* genannt, ist ein Maß für die Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen p, q, hat aber nicht die Eigenschaften einer Metrik. Sie ist definiert als

$$d_{kl}(p, q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \ .$$

Eine symmetrisierte Variante berechnet sich folgendermaßen:

$$d_{kl2}(p,q) = \frac{1}{2}\int_{-\infty}^{\infty} p(x)\log\frac{p(x)}{q(x)} + q(x)\log\frac{q(x)}{p(x)}dx$$

T^2-Distanz Die *T^2-Distanz* basiert auf Hotellings T^2-Statistik und wird für zwei Stichproben X_1, X_2 mit Größen N_1, N_2, welche durch gaußsche Normalverteilungen $\mathcal{N}_1(\mu_1, \Sigma), \mathcal{N}_2(\mu_2, \Sigma)$ mit gleicher Kovarianzmatrix modelliert werden können, berechnet zu [HH04]:

$$d_{T^2}(X_1, X_2) = \frac{N_1 N_2}{N_1 + N_2}(\mu_1 - \mu_2)^T \Sigma^{-1}(\mu_1 - \mu_2)$$

B. Mehr Segfried

Segfried wurde mit Hilfe des *Java Development Kit 5.0 (JDK5.0)* aus dem *Java 2 Standard Edition (J2SE)* Paket entwickelt. Es wird daher die Java Version 1.5 oder höher empfohlen. Ich möchte auch darauf hinweisen, dass die folgenden Kommandozeilenbefehle Linuxspezifisch sind, die Vorgehensweise in anderen Betriebssystemumgebungen aufgrund der vereinheitlichenden Java-Technologie aber analog ist.
Die Installation geschieht durch einfaches Entpacken des *Segfried* Archivs in ein beliebiges Verzeichnis. Nach dem Entpacken des *Segfried* Archivs ist das Wurzelverzeichnis für die Java-Klassen das Verzeichnis „Segfried“, dessen Pfad in der Liste aller Klassenpfade („classpath“) enthalten sein muß. Im einfachsten Fall erreicht man dies, indem man in das *Segfried* Verzeichnis wechselt und dort Java unter der Angabe der Hauptklasse „Segfried“ des Pakets „segfried“ startet:

```
user@linux:~/Segfried> java segfried.Segfried
```

Ansonsten kann man auch den Pfad des *Segfried* Verzeichnisses, also z. B. „/home/user/Segfried“, beim Start als Argument übergeben:

```
user@linux:~> java -classpath /home/user/Segfried segfried.Segfried
```

Sind weitere Klassenpfade anzugeben, so kann man diese, durch „:“ voneinander getrennt, hinzufügen. Klassenpfade können übrigens sowohl Verzeichnisse als auch Java-Archivdateien (Dateien mit der Endung „.jar“) sein.
Möchte man *Segfried*s YALE-Operatoren verwenden, die es ermöglichen, Lernexperimente auf den Wertefolgen durchzuführen und Modelle anzuwenden, dann muß eine YALE Installation vorhanden sein, die Pfade zum YALE Klassenarchiv, z. B. „/home/user/Yale-3.3/lib/yale.jar“, und zum YALE Clustering Plugin, z. B. „/home/user/Yale-3.3/lib/plugins/yale-clustering-3.3.jar“, angegeben und die Java Umgebungsvariable „yale.home“ gesetzt werden:

```
user@linux:~> java \
-classpath /home/user/Segfried:/home/user/Yale-3.3/lib/yale.jar:\
/home/user/Yale-3.3/lib/plugins/yale-clustering-3.3.jar \
-Dyale.home=/home/user/Yale-3.3 segfried.Segfried
```

Hat man Großes vor, dann kann auch noch der maximale Speicher, den die *Java Virtual Machine* für *Segfried* zur Verfügung hat, mit der Option „Xmx“ spezifiziert werden:

```
user@linux:~> java \
-classpath /home/user/Segfried:/home/user/Yale-3.3/lib/yale.jar:\
/home/user/Yale-3.3/lib/plugins/yale-clustering-3.3.jar \
-Dyale.home=/home/user/Yale-3.3 \
-Xmx256M segfried.Segfried
```

für einen maximalen Speicher von 256 Megabyte.

B.1. Segfried erweitern

Um neue Operatoren und Funktionen in *Segfried* zu integrieren, müssen die Klassen „Operator“ bzw. „Function“ erweitert werden. Wenn nach der Kompilierung der neuen Klasse die Klassendatei in dem durch den Konfigurationsparameter „operatorDir“ bzw. „functionDir“ angegebenen Verzeichnis liegt, dann steht sie nach dem Programmstart im Operatorbaumeditor zur Verfügung.

B.1.1. Parameter

Sowohl für Operatoren als auch Funktionen können beliebig viele Parameter der Datentypen *INTEGER*, *DOUBLE* und *STRING* spezifiziert werden. Dies geschieht in der *setup*-Methode der entsprechenden Klasse, wobei man für jeden hinzuzufügenden Parameter den Namen, eine Beschreibung, den Datentyp und ein Objekt, welches die Voreinstellung des Parameterwertes kapselt, angibt:

```
this.getParameters().addParameter("name", "description",
  Parameter.DOUBLE_TYPE, new Double(0.0));
```

Um den Wert eines Parameters zu erhalten, ruft man unter Angabe des Names eine dem Datentyp des Parameters entsprechende Methode auf:

```
this.getParameters().getDoubleParameter("name");
```

Dies sollte vorzugsweise in der *fetch*-Methode passieren, welche vor der ersten Ausführung des Operators bzw. der Funktion einer Operatorbaumanwendung oder nach einer Änderung eines Parameterwertes aufgerufen wird.

B.1.2. Einen Operator implementieren

Das Grundgerüst einer Operatorklasse (Abb. B.1) besteht aus einem Konstruktor und drei Methoden:

NeuerOperator() Der Konstruktor eines Operators sollte den Konstruktor der Oberklasse unter Angabe des Operatornamens aufrufen

setupOperatorParameters() Hier werden die Parameter des Operators wie oben beschrieben spezifiziert

fetchOperatorParameters() Diese Methode wird vor der ersten Ausführung des Operators einer Operatorbaumanwendung oder nach einer Änderung eines Parameters aufgerufen und dient dem Auslesen der Parameterwerte

operate(ValueSeries series) Dies ist die Methode, die auf der Eingabewertefolge operiert und eine Ergebniswertefolge zurückliefert

```
import segfried.operators.Operator;
import segfried.data.Parameter;
import segfried.data.ValueSeries;

public class NeuerOperator extends Operator {

  public NeuerOperator() {
    super("NeuerOperator");
  }

  protected void setupOperatorParameters() { }
  protected void fetchOperatorParameters() { }
```

```
  protected ValueSeries operate(ValueSeries series) throws Exception { }
}
```

Abbildung B.1.: Grundgerüst einer Operatorklasse

B.1.3. Eine Funktion implementieren

Das Grundgerüst einer Funktionenklasse (Abb. B.2) enthält einen Konstruktor und vier Methoden:

NeueFunktion() Der Konstruktor einer Funktion sollte den Konstruktor der Oberklasse unter Angabe des Funktionsnamens aufrufen

setupFunctionParameters() Hier werden die Parameter der Funktion wie oben beschrieben spezifiziert

fetchFunctionParameters() Diese Methode wird vor der ersten Ausführung der Funktion einer Operatorbaumanwendung oder nach einer Änderung eines Parameters aufgerufen und dient dem Auslesen der Parameterwerte

function(ValueSeries series) Dies ist die Methode, die auf der Eingabewertefolge operiert und einen Ergebnisvektor zurückliefert

getDimensionNames() Diese Methode soll die Namen der Dimensionen des Ergebnisvektors liefern

```
import segfried.functions.Function
import segfried.data.Parameter;
import segfried.data.ValueSeries;
import segfried.data.Vector;

public class NeueFunktion extends Function {

  public NeueFunktion() {
    super("NeueFunktion");
  }

  protected void setupFunctionParameters() { }
  protected void fetchFunctionParameters() { }
  protected Vector function(ValueSeries series) throws Exception { }
```

```
  public String[] getDimensionNames() { }
}
```

Abbildung B.2.: Grundgerüst einer Funktionenklasse

B.2. Segfried Operator- und Funktionenreferenz

B.2.1. Operatoren

In diesem Abschnitt werden die zur Zeit in *Segfried* vorhandenen Operatoren und ihre Parameter kurz beschrieben. Alle Operatoren besitzen die folgenden Parameter:

valuedim: setzt die für diesen Operator zu verwendende Dimension der Eingabewertefolge (Wertedimension) für den Fall, dass der Operator nur eindimensional arbeitet (-1: keine Veränderung)

addresult: gibt an, ob die Ausgabewertefolge des Operators zur Liste aller Wertefolgen hinzugefügt werden soll (0: nicht hinzufügen, 1: hinzufügen)

AffineTransform Verschiebt (translatiert) und skaliert eine Wertefolge multidimensional

scaledis: Skalierung der displacement Werte

scaleval: Skalierung der Wertevektoren

transdis: Verschiebung der displacement Werte

transval: Verschiebung der Wertevektoren

AutoCorrelation Berechnet eine Autokorrelationsfunktion

mode: Modus der Berechnung. Im Differenzmodus wird anstatt $AC_i^v = \sum_{j=0}^{N-1} x_j^v \cdot x_{j+i}^v$ folgendes berechnet: $AC_i^v = \sum_{j=0}^{N-1}(a^2 - (x_j^v - x_{j+i}^v)^2)$, wobei a die Amplitude der Folge x ist (0: normal, 1: Differenzmodus)

minlag: der kleinste Versatz in der displacement Dimension

maxlag: der größte Versatz in der displacement Dimension

maxlength: ob zur Berechnung je Versatz die maximal zur Verfügung stehende Länge der Wertefolge verwendet werden soll (variierende Anzahl der Summanden) oder nur die halbe Länge der Wertefolge (konstante Anzahl der Summanden) (0: halbe Länge, 1: maximale Länge)

bpm: ob eine Umrechnung der displacement Dimension in BPM erfolgen soll (0: displacement, 1: BPM)

stepsize: Anzahl der array Indizes um die der Versatz jeweils erhöht werden soll (zur Beschleunigung der Berechnung)

BatchProcessor Führt den Kindoperator auf allen in dem angegebenen Verzeichnis rekursiv gefundenen Wertefolgen aus und speichert gegebenenfalls jeweils die Ergebniswertefolge

inputpath: das Verzeichnis, in dem rekursiv nach Wertefolgen gesucht wird

outputpath: das Verzeichnis, in dem die Ergebniswertefolgen gespeichert werden sollen (kann auch leer bleiben)

Derivative Berechnet die *order*-te Ableitung

order: die Ordnung der Ableitung

DWT Berechnet die Diskrete Wavelet Transformation

direction: die Transformationsrichtung (0: vorwärts, 1: invers)

mother: das Mutter-Wavelet (0: HAAR, 1: D-4)

type: Typ der Transformation. Bei der normalen Transformation wird eine Wertefolge erzeugt, welche alle Koeffizienten der verschiedenen Skalen in nur einer Dimension enthält. Bei der maximum overlap DWT wird für jede Skala eine eigene Dimension erzeugt (0: normal, 1: maximum overlap DWT)

Evaluator Evaluiert Segmentierungen hinsichtlich der Segmentgrenzen oder Segmentmarkierungen

trueseg: Name der wahren Vergleichssegmentierung

evalsegs: Namen der zu evaluierenden Segmentierungen durch „,“ getrennt

inpath: das Verzeichnis, in dem rekursiv nach Wertefolgen gesucht wird, für welche die Segmentierungen evaluiert werden sollen

tolerance: die tolerierte Abweichung von den wahren Segmentgrenzen

mode: Evaluation der Segmentgrenzen oder der Segmentmarkierungen (0: Segmentgrenzen, 1: Segmentmarkierungen)

eval: berechne eine Evaluation je Wertefolge, je Verzeichnis oder für alle Wertefolgen (0: jede Wertefolge, 1: jedes Verzeichnis, 2: alle Wertefolgen)

save: ob die berechneten Evaluationen gespeichert werden sollen (0: nicht speichern, 1: speichern)

FFT Berechnet die Diskrete Fouriertransformation Transformation mittels FFT

direction: die Transformationsrichtung (0: vorwärts, 1: invers)

spectrum: das Spektrum, welches ausgegeben werden soll (0: komplett, 1: Amplitudenspektrum, 2: Power-Spektrum, 3: Phasenspektrum)

scale: Ausgabe der Koeffizientenwerte (0: Werte, 1: dB)

zeropad: ob eine Eingabewertefolge, deren Länge keine Zweierpotenz ist, abgeschnitten oder mit „0“en aufgefüllt werden soll (0: abschneiden, 1: mit „0“en auffüllen)

Integral Berechnet das *order*-te Integral

order: die Ordnung des Integrals

constant: die Integrationskonstante

Labelero Erzeugt aus einer vollständigen Segmentierung eine Teilsegmentierung

origSeg: Name der vollständigen Segmentierung

newseg: Name der zu erzeugenden Teilsegmentierung

inpath: Verzeichnis, in dem nach Wertefolgen gesucht wird, für welche die Teilsegmentierungen erzeugt werden sollen

percentage: erzeuge für jede Markierung der vollständigen Segmentierung Segmente, die diesen Bruchteil der Summe der Längen aller Segmente dieser Markierung abdecken

segpercent: benutze von jedem Segment einer Markierung der vollständigen Segmentierung maximal diesen Bruchteil

Normalize Normalisiert eine Wertefolge

normval: der Wert, auf den normalisiert werden soll, also der maximale Wert

mode: normalisiere alle Werte proportional, jede Dimension unabhängig oder jeden Wertevektor unabhängig (0: proportional, 1: Dimensionen unabhängig, 2: Wertevektoren unabhängig)

OperatorChain Gibt die Eingabewertefolge einfach an die Kette der Kindoperatoren weiter und die Ergebniswertefolge des letzten Kindes zurück

OutlierCorrection Führt eine Ausreißerbehandlung nach Art der dynamischen Programmierung auf einer Wertefolge durch, welche nur eine begrenzte Anzahl verschiedener Werte enthält

maxdiffval: Anzahl der in der Wertefolge vorkommenden verschiedenen Werte

minlength: minimale Länge der entstehenden Segmente in der displacement Dimension

maxlength: maximale Länge der entstehenden Segmente in der displacement Dimension

resolution: die Schrittweite in der displacement Dimension zur Kontrolle der Laufzeit

ParameterIterator Führt seine Kindfunktion auf der Eingabewertefolge wiederholt aus und iteriert dabei einen Parameter eines beliebigen Kindobjektes (Operator oder Funktion)

objectname: Name des Kindobjektes, dessen Parameter iteriert werden soll

paraname: Name des Parameters, der iteriert werden soll

values: einzelne Werte durch „ “ getrennt oder ein Intervall (a-b)

stepsize: die Schrittweite bei Angabe eines Intervalls

PreEmphasis Berechnet die „pre-emphasis“ für eine Audiodatenfolge

alpha: der Parameter *alpha*

PST Berechnet die Phase Space Transformation, die Transformation in den Phasenraum

d: erzeuge Vektoren mit Verzögerung d („delay“)

m: Anzahl der Dimension des Phasenraums

Segmentator Erzeugt aus der Ausgabe des Kindobjekts (Operator oder Funktion) eine Segmentierung. Ist die Ausgabe eine Wertefolge, so werden die Segmente aus maximalen Teilfolgen mit konstanten Werten gebildet. Ist die Ausgabe ein Vektor, dann werden die Vektorelemente als Segmentgrenzen interpretiert

minlength: es werden nur Segmente erzeugt, die mindestens diese Länge in der displacement Dimension haben

segname: der Name der zu erzeugenden Segmentierung

save: ob die erzeugte Segmentierung gespeichert werden soll (0: nicht speichern, 1: speichern)

Similarity Berechnet mit Hilfe des angegebenen Distanzmaßes eine zweidimensionale Ähnlichkeitsmatrix

mode: Berechnung im displacement/displacement oder displacement/Versatz Modus. Im ersten Fall steht die Distanz zweier Vektoren mit Indizes i, j an (i, j) bzw. (j, i), im zweiten Fall an $(i, (j - i) mod n)$ bzw. $(j, (i - j) mod n)$

measure: das zu verwendende Distanzmaß, die euklid'sche Distanz oder cosinus Distanz (0: euklid, 1: cosinus)

bytes: Anzahl der bytes für jeden Wert der Ähnlichkeitsmatrix. Bestimmt den Speicherverbrauch

SimilarSegment Berechnet eine Wertefolge, welche die Ähnlichkeit der Wertefolge mit einer durch das erste Segment der Segmentierung gekennzeichneten Teilfolge für alle Stellen angibt

distmeasure: das zu verwendende Distanzmaß, die euklid'sche Distanz oder cosinus Distanz (0: euklid, 1: cosinus)

VectorMetrics Berechnet eine Wertefolge, welche entweder die Längen der Differenzvektoren von oder die Winkel zwischen allen Nachbarvektorpaaren der Eingabewertefolge angibt.

mode: erzeuge die Differenzvektorlängen oder Winkel zwischen den Vektoren (0: Längen, 1: Winkel)

Window Wendet eine Fensterfunktion auf die Wertefolge an

window: die anzuwendende Fensterfunktion (1: Hann Fenster, 2: Hamming Fenster)

Windowing Teilt die Eingabewertefolge in Fenster bzw. Teilfolgen auf, führt für jede Teilfolge die Kindfunktion aus und erzeugt aus den Ausgabevektoren wieder eine Wertefolge

sizein: ob die Angaben der Fensterbreite und Schrittweite in array Indizes oder in der displacement Dimension angegeben werden (0: array Indizes, 1: displacement)

windowsize: die Fensterbreite in array Indizes oder in der displacement Dimension

stepsize: die Schrittweite in array Indizes oder in der displacement Dimension

displace: ob für jede Teilfolge die gleichen displacement Werte oder die originalen displacement Werte verwendet werden sollen (0: die gleichen, 1: die originalen)

maxwindows: hiermit kann man die maximal zu erzeugenden Teilfolgen begrenzen (-1: alle möglichen Teilfolgen erzeugen)

YaleLearner Dieser Operator erzeugt aus der Eingabewertefolge oder aus mehreren zu ladenden Wertefolgen eine Beispielmenge, führt ein YALE Experiment darauf aus und gibt eine Markierungs-, Cluster-Id- oder Beispielmengenfolge zurück

expath: Pfad zu einem YALE Experiment

inpath: Verzeichnis, in dem nach Wertefolgen gesucht wird, aus denen die Beispielmenge generiert werden soll

outpath: Verzeichnis, in dem das gelernte Modell gespeichert werden soll (kann leer sein)

labelatt: ob ein „label“ Attribut (Markierung) erzeugt werden soll (0: nein, 1: ja)

predatt: ob ein „predicted label“ Attribut erzeugt werden soll (0: nein, 1: ja)

idatt: ob ein „id“ Attribut erzeugt werden soll (0: nein, 1: ja)

segname: der Name der Segmentierung, aus welcher die Markierungen bestimmt werden sollen

YaleModelApplier Läd ein Vorhersagemodell, wendet es auf die Eingabewertefolge an und gibt eine Markierungsfolge zurück

modelpath: Pfad zu einem YALE Vorhersagemodell

B.2.2. Funktionen

In diesem Abschnitt werden die zur Zeit in *Segfried* vorhandenen Funktionen und ihre Parameter kurz beschrieben. Alle Funktionen besitzen den folgenden Parameter:

valuedim: setzt die für diese Funktion zu verwendende Dimension der Eingabewertefolge (Wertedimension) für den Fall, dass die Funktion nur eindimensional arbeitet (-1: keine Veränderung)

Area Berechnet die Fläche unter einer Wertefolge, den Centroiden dieser Fläche oder den Rolloff, d. h. die Stelle in der displacement Dimension, die die Fläche in einem gewissen Verhältnis teilt

mode: berechne nur den Centroiden, die Fläche, den Rolloff oder alle Werte (0: Centroid, 1: Fläche, 2: Rolloff, 3: alle)

fraction: gibt für die Berechnung des Rolloff den Bruchteil der Fläche an, welcher links von der Rolloff-Stelle liegen soll

ChromaVector Gibt einen 12-dimensionalen Vektor zurück, dessen Elemente den Anteil der 12 Tonklassen A, A#, H, C, C#, D, D#, E, F, F#, G, G# an einem Spektrum repräsentieren

octaves: über wieviele Oktaven aufsummiert werden soll

start: die Startoktave

CriterionFunction Berechnet eine Kriteriumsfunktion und liefert ein Maß für die Ähnlichkeit der ersten und zweiten Hälfte einer Wertefolge

criterion: die zu verwendende Kriteriumsfunktion (0: Korrelationskriterium, 1: Fisher's Kriterium)

sum: liefere einen Vektor mit Werten für jede Dimension der Wertefolge oder die Summe dieser Werte zurück (0: Vektor, 1: Summe aller Vektorelemente)

Displacement Liefert das arithmetische Mittel des ersten und letzten Wertes der displacement Werte der Wertefolge zurück

EvaluatorFunction Evaluiert Segmentierungen und gibt einen Performanzvektor mit Precision, Recall und F-Measure zurück

trueseg: Name der wahren Vergleichssegmentierung

evalseg: Name der zu evaluierenden Segmentierung

inpath: das Verzeichnis, in dem nach Wertefolgen gesucht wird, für welche die Segmentierung evaluiert werden soll

outpath: das Verzeichnis oder die Datei, in die das Ergebnis geschrieben werden soll

tolerance: die tolerierte Abweichung von den wahren Segmentgrenzen

labels: evaluiere die Segmentgrenzen oder Segmentmarkierungen (0: Segmentgrenzen, 1: Segmentmarkierungen)

FeatureExtractor Gibt die Eingabewertefolge an die Kette der Kindobjekte (Operatoren und Funktionen) weiter, sammelt die Ergebnisvektoren aller Kindfunktionen und liefert einen Vektor zurück, welcher eine Konkatenation jener Vektoren ist

Flux Gibt die Summe der quadrierten, je Wertefolgenindex berechneten Differenzen aller Werte der Wertedimension zweier aufeinanderfolgender Wertefolgen zurück, für die diese Funktion aufgerufen wird

LevelCrossings Berechnet die Anzahl der Über- bzw. Unterschreitungen eines bestimmten Referenzwertes in einer Wertefolge, z. B. für den Wert 0,0 die Anzahl der Nulldurchgänge

level: der Referenzwert

MaxFinder Findet maximale Werte in einer Wertefolge und liefert sie oder ihre Stellen in der displacement Dimension zurück

mode: finde die *number* größten Werte oder Spitzen in einer Wertefolge (0: maximale Werte, 1: Spitzen)

number: die maximale Anzahl der zu findenden Werte

sloppy: für die Bestimmung von Spitzen die tolerierte Anzahl von falsch, also nach oben gerichteten Werten eines Spitzenrückens

tolerance: für die Bestimmung von Spitzen das Verhältnis vom Wert zum aktuellen Spitzen-Durchschnittswert, unterhalb dessen der Wert noch zur Spitze gezählt wird

minpeakdist: der minimale Abstand zwischen zwei Spitzen

return: gib die Maximalwerte oder ihre Stellen in der displacement Dimension zurück (0: displacement Werte, 1: Maximalwerte)

constlen: ob der Ergebnisvektor genau *number* Dimensionen haben soll oder auch weniger haben kann (0: erlaube weniger Dimensionen, 1: genau *number* Dimensionen)

MFCC Berechnet für ein Spektrum die Mel Frequency Cepstral Coefficients

cepstra: Anzahl der zu berechnenden Koeffizienten

lowerfilter: die Mittenfrequenz des untersten Filters

melfilters: die Anzahl der Mel-Filter

Moments Berechnet für eine Wertefolge multidimensional die statistischen Kenngrößen Mittelwert, Standardabweichung, Schiefe und Wölbung (mean, standarddeviation, skewness, kurtosis)

moments: die Anzahl der zu berechnenden Momente pro Dimension (1-4)

Ngram Liefert einen Vektor zurück, welcher eine Konkatenation aller Vektoren der Wertefolge ist

Sum Berechnet die Summe oder den Root Mean Square aller Werte der Wertefolge

return: gib die Summe oder den RMS zurück (0: Summe, 1: RMS)

Transpose Erzeugt aus der Wertedimension der Wertefolge einen Vektor

dims: die maximale Anzahl der Dimensionen des Ergebnisvektors (-1: gesamte Länge der Wertefolge)

C. Erweiterung des YALE Clustering-plugin

Zur Implementierung des MPCK-Means-Algorithmus wurde das YALE Clustering-plugin um folgende Klassen erweitert:

- Paket **edu.udo.cs.yale.clustering.clusterer**:

  MPCKMeans Der Operator des MPCK-Means Algorithmus

- Paket **edu.udo.cs.yale.clustering.constraints**:

  ClusterConstraint Die Superklasse aller Cluster-constraints

  ClusterConstraintList Die Superklasse aller Cluster-constraint-Listen

  LinkClusterConstraint Modellierung eines Must-Link- oder Cannot-Link-constraint

  LinkClusterConstraintList Datenstruktur zur Verwaltung einer Menge von Link-Cluster-constraints

  ExampleSet2ClusterConstraintList Erzeugung einer Cluster-constraint-Menge aus einer Beispielmenge mit markierten Beispielen

- Paket **edu.udo.cs.yale.clustering.evaluation**:

  ClusterConstraintsEvaluator Bestimmung der Performanz eines Cluster-Modells im Hinblick auf eine Cluster-constraint-Menge

Abbildungsverzeichnis

Tabellenverzeichnis

Stichwortverzeichnis

Literaturverzeichnis

[AMB04] J. Ajmera, I. McCowan, and H. Bourlard. *Robust Audio Segmentation*. PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, June 2004. thesis # (IDIAP-RR 04-35).

[ANS+05] Samer A. Abdallah, Katy Noland, Mark Sandler, Michael Casey, and Christophe Rhodes. Theory and evaluation of a bayesian music structure extractor. In *ISMIR*, pages 420–425, 2005.

[AS01] J.-J. Aucouturier and M. Sandler. Segmentation of musical signals using hidden markov models. In *Proceedings of the Audio Engineering Society 110th Convention*, May 2001.

[AS02] J.-J. Aucouturier and M. Sandler. Finding repeating patterns in acoustic musical signals. In *AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio*, 2002.

[BBM04] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning, ICML*, pages 81–88, Banff, Canada, July 2004.

[Bia03] Konstantin Biatov. The method of an audio data classification and segmentation. NATO Advanced Study Institute - Computative Noncommutative Algebra and Applications, Italy, 2003.

[Bil97] Jeff A. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, University of Berkeley, 1997.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[CV03] Wei Chai and Barry Vercoe. Structural analysis of musical signals for indexing and thumbnailing. In *JCDL*, page 27. IEEE Computer Society, 2003.

[CVR05] M. Cettolo, M. Vescovi, and R. Rizzi. Evaluation of bic-based algorithms for audio segmentation. *Computer Speech and Language*, 19(2):147–170, April 2005.

[CW04] Shih-Sian Cheng and Hsin-Min Wang. Metric-seqdac: A hybrid approach for audio segmentation. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP*, Jeju Island, Korea, October 2004.

[Dax06] Alexander Daxenberger. Segfried - audio feature extraction and segmentation utility. http://segfried.sourceforge.net, October 2006.

[DH] Roger B. Dannenberg and Ning Hu. Discovering musical structure in audio recordings. *Lecture Notes in Computer Science*, 2445.

[DH02] Roger B. Dannenberg and Ning Hu. Pattern discovery techniques for music audio. In *ISMIR*, 2002.

[DKW99] P. Delacourt, D. Kryze, and C. Wellekens. Speaker-based segmentation for audio data indexing. In *Proceedings of the ESCA ETRW workshop*, pages 437–440, Cambridge, UK, 1999.

[FC03] Jonathan T. Foote and Matthew L. Cooper. Media segmentation using self-similarity decomposition, April 2003.

[FGP+98] Jonathan Fiscus, John Garofolo, Mark Przybocki, William Fisher, and David Pallett. 1997 english broadcast news speech (hub4), 1998.

[FGZ01] Zheng Fang, Zhang Guoliang, and Song Zhanjiang. Comparison of different implementations of mfcc. In *Journal of Computer Science and Technology*, volume 16, 2001.

[Got03] Masataka Goto. A chorus-section detecting method for musical audio signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, volume 5, pages 437–440, 2003.

[Got04] Masataka Goto. A predominant-f0 estimation method for polyphonic musical audio signals. In *Proceedings of the 18th International Congress on Acoustics, ICA*, pages 1085–1088, April 2004.

[HH04] R. Huang and J.H.L Hansen. Unsupervised audio segmentation and classification for robust spoken document retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, volume 1, May 2004.

[JLC03] M.-H. Jian, C.-H. Lin, and A. Chen. Perceptual analysis for music segmentation. In M.-M. Yeung, R.-W. Lienhart, and C.-S. Li, editors, *Storage and Retrieval Methods and Applications for Multimedia 2004. Proceedings of the SPIE*, volume 5307, pages 223–234, December 2003.

[KS03] H.-G. Kim and T. Sikora. Automatic segmentation of speakers in broadcast audio material. In M.-M. Yeung, R.-W. Lienhart, and C.-S. Li, editors, *Storage and Retrieval Methods and Applications for Multimedia 2004. Proceedings of the SPIE*, volume 5307, pages 429–438, December 2003.

[KSWW00] Thomas Kemp, Michael Schmidt, Martin Westphal, and Alex Waibel. Strategies for automatic segmentation of audio data. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, volume 3, pages 1423–1426, 2000.

[Log00a] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the first International Symposium on Music Information Retrieval, ISMIR*, Plymouth, Massachusetts, October 2000.

[Log00b] Beth Logan. Music summarization using key phrases. In *Proceedings of the IEEE International Conference on Audio, Speech and Signal Processing, ICASSP*, 2000.

[LZL03] Lie Lu, Hong-Jiang Zhang, and Stan Z. Li. Content-based audio classification and segmentation by using support vector machines. *Multimedia Systems*, 8(6):482–492, 2003.

[Mie03] Ingo Mierswa. Beatles vs. bach: Merkmalsextraktion im phasenraum von audiodaten. In *LLWA 03 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivitaet*, 2003.

[Mie04] Ingo Mierswa. Automatisierte merkmalsextraktion aus audiodaten. Master's thesis, Fachbereich Informatik, Universitaet Dortmund, 2004.

[MM05] Ingo Mierswa and Katharina Morik. Automatic feature extraction for classifying audio data, 2005.

[MMFD05] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle. jaudio: A feature extraction library. In *International Conference on Music Information Retrieval*, pages 600–603, 2005.

[MN05] Hugo Meinedo and Joao Neto. A stream-based audio segmentation, classification and clustering pre-processing system for broadcast news using ann models. In *INTERSPEECH*, pages 237–240, 2005.

[MU04] Fabian Moerchen and Alfred Ultsch. Mining hierarchical temporal patterns in multivariate time series, 2004.

[MWK+06] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, 2006.

[OCR05] M. Omar, U. Chaudhari, and G. Ramaswamy. Blind change detection for audio segmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, volume 1, pages 501–504, March 2005.

[PE04] R. Mitchell Parry and Irfan Essa. Feature weighting for segmentation. In *ISMIR*, 2004.

[PLBR02] Geoffroy Peeters, Amaury La Burthe, and Xavier Rodet. Toward automatic music audio summary generation from signal analysis. In *ISMIR*, 2002.

[PM01] Mark Przybocki and Alvin Martin. 2000 nist speaker recognition evaluation, 2001.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, February 1989.

[SFL02] Chien-Wen Danny Su, Keith Fung, and Andrei Leonov. Oc volume - java speech recognition engine. http://ocvolume.sourceforge.net, 2002.

[TC99] G. Tzanetakis and P. Cook. Multifeature audio segmentation for browsing and annotation. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, New Paltz, NY, 1999.

[TC02] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. In *IEEE Transactions on Speech and Audio Processing*, volume 10, July 2002.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[WH06] Chung-Hsien Wu and Chia-Hsin Hsieh. Multiple change-point audio segmentation and classification using an mdl-based gaussian model. *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):647–657, March 2006.

[WIK07] Wikipedia - the free encyclopedia. http://www.wikipedia.org, April 2007.

[ZK99] Tong Zhang and C.-C. Jay Kuo. Heuristic approach for generic audio data segmentation and annotation. In *Proceedings of the 7th ACM International Multimedia Conference (MM-99)*, pages 67–76, N.Y., November 1999. ACM Press.

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst, keine anderen als die angegebenen Hilfsmittel verwendet und sämtliche Stellen, die den benutzten Werken dem Wortlaut oder dem Sinn nach entnommen sind, mit Quellenangaben kenntlich gemacht habe. Alle wörtlich entnommenen Stellen habe ich als Zitate markiert. Diese Versicherung gilt auch für Zeichnungen, Skizzen und sonstige bildliche Darstellungen.

Der Datenträger, auf dem der Text dieser Arbeit gespeichert wurde, befindet sich in meinem Besitz.

Dortmund, 15.05.2007

Alexander Daxenberger