Diplomarbeit

Automatisierte Merkmalsextraktion aus Audiodaten

Ingo Mierswa

Diplomarbeit
am Fachbereich Informatik
der Universität Dortmund

19. September 2004

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inform. Michael Wurst

Revidierte Fassung vom 19.09.2004

Danksagung

Mein Dank gilt zuerst Prof. Dr. Katharina Morik und Dipl.-Inform. Michael Wurst, die diese Arbeit betreut und mich durch zahlreiche Ermutigungen immer wieder auf die richtige Spur gebracht haben. Ralf Klinkenberg und Oliver Ritthoff danke ich für die Heranführung an das maschinelle Lernen. Mit Simon Fischer, Ralf und Oliver war es möglich, die Experimentierumgebung YALE zu entwickeln. Boris Schulze danke ich für seine Einführung in die Fourier-Transformation während wir vor Kroatiens Küste kreuzten und Konrad Ullrich für die aufmerksame Durchsicht der Arbeit. Der größte Dank gebürt jedoch meiner Frau Nadja, die mich während der gesamten Zeit mit unendlicher Geduld unterstützte.

Besonderheiten von Audiodaten

Diese Arbeit benutzt sowohl Methoden der statistischen Wertereihenanalyse als auch Erkenntnisse der Psychoakustik. Die meisten dieser Methoden sind für andere Datensätze bestehend aus umfangreichen Wertereihen ebenfalls verwendbar, einige wenige jedoch können nur für Audiodaten sinnvoll genutzt werden. Solche Methoden werden durch eine Note im Seitenrand gekennzeichnet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

KAPITEL 1

Einleitung

Immer häufiger werden Musikstücke in digitaler Form archiviert und verwaltet. Während noch vor wenigen Jahrzehnten analoge Aufnahmeverfahren wie Bänder und Schallplatten den Markt beherrschten, sind heute Musikstücke in digitaler Form auf CD oder in Dateien gespeichert. Diese digitalen Formate erlauben eine direkte Verarbeitung durch Rechner. Durch die weltweite Vernetzung und die Entwicklung neuartiger Kompressionsverfahren erfahren Musikdaten zudem eine immer raschere Verbreitung. Nicht selten haben Benutzer Musikdatenbanken mit mehreren Tausend Liedern angelegt und leiden unter dem Aufwand ihrer Pflege. Auch die Musikindustrie berücksichtigt diesen Trend und es werden Portale entwickelt, die es dem Anwender erlauben, die von ihm gewünschte Musik auf seinen Rechner zu laden. Eine typische Aufgabe bei der Behandlung von Musikdaten ist die Klassifikation von Audiodaten nach ihrem Genre, um unbekannte Musikstücke schnell katalogisieren zu können. Bei einer Kunstform wie Musik ist der Hörgenuss jedoch auch durch emotionale Empfindungen geprägt. Daher ist es eine zusätzliche Herausforderung, neben der Klassifikation nach Genre auch die Präferenzen eines Benutzers zu erkennen. Dies erlaubt personalisierte Zusammenstellungen von Musikstücken oder Hörvorschläge. Ein Ziel dieser Arbeit ist es, für solche Klassifikationsaufgaben eine möglichst hohe Genauigkeit in der Vorhersage zu erreichen.

Dabei werden maschinelle Lernverfahren mit dem Problem großer Datenmengen konfrontiert. Es ist nicht möglich, die Audiodaten direkt als Eingabe für ein Lernverfahren zu benutzen. Nicht alle Musikstücke besitzen die gleiche Länge und die Instanzen hätten somit eine unterschiedliche Zahl von Merkmalen, die untereinander nicht korrelieren. Der i-te Zeitpunkt eines jeden Liedes hat als Merkmal keine besondere Bedeutung für ein Lernverfahren. Es steigt also nicht nur die Laufzeit der Verfahren mit der Länge der Lieder, es ist darüberhinaus auch keine ausreichend gute Performanz zu erwarten [45].

Es sind zwei Wege denkbar, diese Problematik zu umgehen und Audiodaten als Eingabe für Lernverfahren zu benutzen. Zunächst wäre es möglich, aus den Audiodaten, welche in Wellenform vorliegen, Midistücke oder Partituren zu extrahieren. Beim Midiformat sind alle hörbaren Noten nach dem 12-Ton-System nummeriert. Jede Note in einem Musikstück umfasst die Tonhöhennummer, den Startzeitpunkt, die Dauer und die Lautstärke [28]. Partituren beschreiben ebenso die Melodien, Harmonien und rhythmische Elemente aller Instrumente. Aus diesen Angaben können dann Merkmale extrahiert werden, welche als Eingabe für ein Lernverfahren benutzt werden können [41]. Die Transkription von Audiodaten unterliegt jedoch einigen Beschränkungen. Gute Ergebnisse lassen sich nur bei einstimmigen Stücken erzielen. Für polyphone Musik gelingt eine Erkennung nur, wenn alle Stimmen von demselben Instrument gespielt werden, z. B. ein mehrstimmiges Klavierstück. Probleme bereiten die Erkennung von Oktaven und Akkorden.

Auch werden gleiche aufeinanderfolgende Töne nur selten erkannt [4, 30]. Ein Einsatz für Stücke mit reicher Instrumentierung, wie einem Orchester oder einer Band, ist daher im Augenblick nicht möglich. Andere Aufgaben, für die eine Transkription in Mididateien erforderlich ist, umfassen das Finden von Merkmalen für den musikalischen Ausdruck. Dieser wird definiert durch Variationen in Tempo, Dynamik oder Interpretation eines Spielers [59]. Zu diesem Teilgebiet gehört auch die Klassifizierung von Gesangsinterpretationen bezüglich Tonreinheit [57]. Auch für neuartige Anfragen an Multimedia-Datenbanken ist eine Transkription der Melodien erforderlich. Bekannt wurde die Anfrage durch Summen der Hauptmelodie (*query by humming*) [14]. Die Transkription von Audiodaten in das Midiformat, welche den Einsatz musikwissenschaftlichen Hintergrundwissens erlauben würde, muss für Aufgaben wie die Feststellung der Autorenschaft oder die Messung der Qualität der Interpretation oder des musikalischen Ausdrucks durchgeführt werden. Diese Art von Klassifikationsaufgaben soll in dieser Arbeit jedoch nicht untersucht werden.

Die zweite Möglichkeit für die Klassifikation von Audiodaten ist, die Merkmale direkt aus den Wellenformen zu extrahieren. Diese können sowohl in komprimierten als auch in verlustfreien Datenformaten vorliegen [46]. Eine Extraktion direkt aus der Wellenform war erst in jüngster Zeit und mit einem Anwachsen der Hauptspeicher moderner Rechner möglich. Die meisten Merkmale, die bisher aus Wellenformen extrahiert wurden, entstammen dem Bereich der Repräsentation und Verarbeitung von Sprachsignalen [6]. Man beschränkte sich zunächst auf die Klassifikation einzelner Töne, die Zuordnung zu Instrumenten oder auf die Unterscheidung zwischen männlicher und weiblicher Stimme [20]. Später erfolgte die Extraktion aufwändiger Merkmale wie Tempo [8, 10] oder Tonhöhen. Erste erfolgreiche Merkmalssätze zur inhaltsbasierten Kennzeichnung von Audiodaten lieferten [27] und [61], welche jedoch nur wenige Merkmale im Zeit- und im Frequenzraum extrahierten. Im letzteren können vor der Extraktion Anpassungen an das menschliche Ohr durchgeführt werden. Tiefe Frequenzen werden abgeschwächt und hohe angehoben, wodurch das menschliche Hörvermögen simuliert wird [35, 62]. Aufgrund der unterschiedlichen Wurzeln liefern die jüngsten Ergebnisse der Behandlung von Audiodaten eine Vielzahl von Merkmalen, die historisch aus den einzelnen Ansätzen zusammengewachsen sind [13, 15, 29, 54, 55].

Die meisten der bisher extrahierten Merkmale leiden jedoch unter einer hohen Anfälligkeit gegen Zeittranslationen oder verminderter Qualität des Ausgangsmaterials. Außerdem lässt die Beschränkung auf die beiden gängigen Räume von Audiodaten (Zeit- und Frequenzraum) und eine fehlende Systematik der Extraktion einige wichtige Merkmale vermissen. In umfassenden Übersichten der Zeitreihenanalyse wie [49] finden sich zahlreiche Methoden, die bis jetzt keine Anwendung für die Merkmalsextraktion aus Audiodaten gefunden haben. Eine hervorragende Einführung in die Analyse nichtlinearer Systeme ist in [5] gegeben. Die dort beschriebene Rekonstruktion des Zustandsraums erlaubt ebenfalls die Extraktion neuartiger Merkmale aus Audiodaten [31].

Für ein zufriedenstellendes Klassifikationsergebnis ist also eine umfassende Extraktion robuster und aussagekräftiger Merkmale erforderlich. Hierbei wird man jedoch mit zwei Problemen konfrontiert: Es kann nicht garantiert werden, dass für unterschiedliche Klassifikationsaufgaben dieselben Merkmalssätze geeignet sind, um gute Lernergebnisse zu erzielen. Merkmale, welche die Genres Klassik und populäre Musik gut zu trennen vermögen, sind kaum für eine Trennung zwischen Rock und Volksmusik geeignet. Dieses Problem wird verschärft für eine Klassifikation der Benutzerpräferenz. Die Kriterien, welche Lieder ein Hörer mag und welche nicht, sind bei jedem Hörer andere und können sich im Laufe der Zeit ändern. Bei einem umfassendem Merkmalssatz gibt es zudem das Problem, dass eine Auswahl der optimalen Merkmale von Hand äußerst aufwändig und schwierig ist. Dies gilt insbesondere, wenn der Anwender nur geringe Kenntnisse über die vorliegenden Daten mitbringt.

Der Ausweg aus dieser Situation ist eine automatisierte Erzeugung und Auswahl der besten Merkmale für die vorliegende Klassifikationsaufgabe. In dieser Arbeit wird ein neues adaptives Verfahren vorgestellt, welches auf genetischer Programmierung basiert und automatisch aus einer Menge von Methoden die besten wählt und kombiniert, um so einen optimalen Merkmalssatz zu extrahieren. Auf diese Weise kann für jede Klassifikationsaufgabe getrennt eine Menge von Merkmalen erzeugt werden, mit der die Lösung der gestellten Klassifikationsaufgabe ermöglicht wird.

Die Anwendungen sind weitreichend: Musikportale können dem interessierten Benutzer gezielte Vorschläge unterbreiten und die Pflege einer Musikdatenbank kann vereinfacht bzw. personalisiert werden [54]. Neuartige Abspielprogramme können die Hörgewohnheiten des Benutzers analysieren und die Abspiellisten entsprechend an den momentanen Geschmack anpassen. Merkmale, die sich für Klassifikationsverfahren wie k nächste Nachbarn eignen, bilden eine gute Basis für ein Ähnlichkeitsmaß oder zur Indexierung großer Musikdatenbanken [24, 44]. Der Fokus dieser Arbeit soll jedoch auf der Klassifikation von Audiodaten liegen, sowohl nach Genre als auch nach Benutzerpräferenz.

Ziel dieser Arbeit ist also die Entwicklung eines Verfahrens, welches aus einer Menge von Methoden automatisch diejenigen auswählt und kombiniert, welche am besten in der Lage sind, einen optimalen Merkmalssatz für Klassifikationsaufgaben aus umfangreichen Wertereihen zu extrahieren. Die Qualität der Methoden zur Merkmalsextraktion und der daraus resultierenden Merkmale kann anhand der folgenden Kriterien beurteilt werden:

Klassifikationsgüte: Anhand bekannter Performanzmaße wird die Nützlichkeit der extrahierten Merkmale für die gegebenen Klassifikationsaufgaben gemessen.

Verständlichkeit: Die extrahierten Merkmale sollen in der Domäne der Musikdaten erklärbar sein. Auch aufwändig konstruierte Merkmale sollen einfache natürlichsprachliche Bedeutungen erhalten.

Minimierung des Benutzeraufwandes: Durch eine automatisierte Auswahl und Kombination der Methoden wird der Benutzeraufwand für die Merkmalsextraktion minimiert.

Einfache Extraktion von Hand: Der Benutzer soll jederzeit die Möglichkeit haben, eigene oder bereits bekannte Merkmale zu erzeugen und dem automatisiert erstellten Merkmalssatz hinzuzufügen.

Vollständigkeit: Sowohl die bereits bekannten Methoden zur Merkmalsextraktion als auch neue Möglichkeiten ihrer Kombination sollen beschrieben und angewendet werden.

Systematisierung: Die bisher vorliegenden Arbeiten zur Merkmalsextraktion aus Audiodaten leiden darunter, dass die Merkmale nicht systematisch extrahiert werden. Eine gute automatisierte Merkmalsextraktion setzt jedoch eine gute Systematisierung der eingesetzten Methoden voraus.

Robustheit: Die extrahierten Merkmale müssen robust sein gegenüber verschiedener Qualitäten der Daten wie unterschiedliche Kompressionstechniken oder Verzerrung.

Effizienz: Alle eingesetzten Methoden müssen effizient arbeiten, um in zumutbarer Zeit eine Merkmalsextraktion vornehmen zu können. Als effizient werden in dieser Arbeit alle Methoden mit polynomieller oder besserer Laufzeit betrachtet.

Echtzeitfähigkeit: Die extrahierten Merkmale sollen für einen Echtzeitbetrieb nutzbar sein. Die Ergebnisse der Merkmalsextraktion müssen also übertragbar sein auf eine Anwendung während des Hörens.

Diese Arbeit beginnt in Kapitel 2 mit einer kurzen Einführung in das maschinelle Lernen aus Wertereihen und den wesentlichen Definitionen. In diesem einführenden Kapitel wird auch eine Systematisierung aller Methoden der Wertereihenanalyse vorgestellt. Diese werden unterschieden in die Gruppe der *Transformationen* und die Gruppe der *Funktionale*. Die verschiedenen Arten der Transformationen werden in den Kapiteln 3 bis 5 ausführlich behandelt. Dabei werden auch für die Analyse von Musikdaten ungewöhnliche Transformationen, wie die Rekonstruktion des Zustandsraums, beschrieben. Kapitel 6 führt das Konzept der allgemeinen Fensterung ein und diskutiert, welche Transformationen mit Hilfe der Fensterung durchgeführt werden können. Damit ist eine Simulation aller bereits existierenden Methoden und die Kreation neuer Merkmale möglich.

In den Kapiteln über Transformationen werden Funktionale bereits benutzt. Ihre exakte Definition erfolgt erst in Kapitel 7, in dem auch gezeigt wird, wie sich mit ihrer Hilfe Merkmale extrahieren lassen. Wo erforderlich lenken Querverweise die Aufmerksamkeit auf ergänzende Informationen an den betreffenden Stellen der Arbeit. Für alle Transformationen und Funktionale wird eine effiziente Laufzeit nachgewiesen.

Kapitel 8 stellt eine Möglichkeit zur automatisierten Merkmalsextraktion mittels genetischer Programmierung vor und untersucht den Suchraum der Methoden. Die Laufzeit einer Merkmalsextraktion wird analysiert. Dieser neuartige Ansatz zur adaptiven Merkmalsextraktion aus umfangreichen Wertereihen wird in Kapitel 9 anhand realer Audiodaten evaluiert. Schließlich erfolgt eine Zusammenfassung und eine Überprüfung, in wie weit die oben genannten Kriterien erfüllt wurden.

KAPITEL 2

Maschinelles Lernen und Wertereihen

Wie in der Einleitung bereits angedeutet wurde, sollen Klassifikationsaufgaben für Audiodaten mit Hilfe maschineller Lernverfahren bearbeitet werden. Es soll zunächst definiert werden, was maschinelle Lernverfahren sind und wozu sie benutzt werden können. Im Anschluss können die Besonderheiten des Lernens aus Wertereihen diskutiert werden. Dies wird dazu führen, Audiodaten als Wertereihen zu betrachten, um mit Hilfe von Techniken der Wertereihenanalyse Merkmale zu extrahieren, welche die Anwendung von Lernverfahren vereinfachen.

Von *Lernen* kann stets dann gesprochen werden, wenn ein System sein Verhalten in einer Weise ändert, dass es sich in Zukunft „besser" verhält [60]. Diese allgemeine Definition hängt stark von dem gewählten Performanzmaß ab, dessen Existenz nicht immer garantiert werden kann. Dies stellt für die Praxis jedoch kaum eine Einschränkung dar, da man ohnehin häufig an einer Abschätzung der Güte des Lernergebnisses interessiert ist und somit ein Performanzmaß benötigt. Diese Definition kann leicht weiter formalisiert werden, indem das Lernen als ein Suchproblem aufgefasst wird [34]. Die Eingabe eines Lernverfahrens ist dabei in einer Repräsentationssprache L_E dargestellt. L_E beschreibt im Folgenden eine Instanzmenge E, bei der jede Instanz $e \in E$ ein Vektor von Ausprägungen einer Anzahl Merkmale darstellt. Ein Lernverfahren versucht für solche Instanzen eine Hypothese h aus einer Hypothesensprache L_H zu finden, welche den erwarteten Fehler für die Vorhersagen minimiert. Darüberhinaus ist es zumindest wünschenswert, wenn die Hypothese h Rückschlüsse über die Natur der Daten erlaubt [60].

Lernen aus Vektoren, deren Komponenten die Ausprägungen unterschiedlicher Merkmale darstellen, ist eine gut untersuchte Aufgabe. Die Ausprägungen können dabei dem numerischen Raum entstammen oder auch symbolische Begriffe darstellen. Eine große Anzahl von Lernverfahren ist entwickelt worden, um solchen Daten die inhärenten Informationen zu entlocken. Die gelernten Hypothesen sollen sowohl auf neue und ungesehene Daten anwendbar sein als auch Informationen und Erkenntnisse über die Daten und die vorliegende Lernaufgabe vermitteln. Auf die Funktionsweisen der verschiedenen Lernverfahren soll in dieser Arbeit nicht eingegangen werden. Eine gute Einführung ist gegeben in [34]. Eine ausführliche Beschreibung des bekanntesten Entscheidungsbaumlerners findet sich in [43]. Support Vector Machines, die ebenfalls im Rahmen dieser Arbeit Verwendung finden, werden in [7] erklärt.

Audiodaten können leicht in ein Vektorformat überführt werden und somit als Eingabe für Lernverfahren dienen. Dabei stellen sich zwei Probleme: Erstens sind Audiodaten äußerst umfangreich und zweitens berücksichtigen Lernverfahren nicht die zeitliche Ordnung der vorliegenden Daten. Daher wird die Information verschenkt, zu welchem Zeitpunkt ein Ereignis eintritt. In dieser Arbeit werden Audiodaten daher als geordnete Wertereihen aufgefasst. Auf diese Weise können auf den umfangreichen Daten bereits bekannte Methoden der Wertereihenanalyse angewendet werden, um das Lernen zu erleichtern. Diese Besonderheit des Lernens aus Wertereihen wird in den nächsten Abschnitten genauer untersucht.

2.1. Wertereihen

Auch wenn eine der Dimensionen meist nur implizit vorliegt, sind Reihen mindestens zweidimensional. Eine häufige Form von Wertereihen sind *Zeitreihen*, bei denen diese implizite Dimension den Verlauf der Zeit darstellt. Im *univariaten* Fall handelt es sich dabei um einfache Werte $x_1, \ldots, x_n$ einer Eigenschaft. Zu jedem Zeitpunkt i kann z.B. ein Messwert x_i datiert worden sein. Für *multivariate* Zeitreihen liegt zu jedem Zeitpunkt i ein Vektor $\mathbf{x}_i$ von Werten vor. Natürlich sind wir hierbei nicht auf numerische Domänen beschränkt, die Ausprägungen der Merkmale können durchaus nominal sein.

Definition 2.1 (Wertereihe) *Unter einer* Wertereihe *oder* Reihe *versteht man eine Abbildung*

$$x : \mathbb{N} \to \mathbb{R} \times \mathbb{C}^m$$

Anstelle von $x(n)$ schreiben wir x_n und für eine endliche Folge der Länge n schreiben wir $(x_i)_{i\in\{1,\ldots,n\}}$. [1]

Ein Vorteil dieser Definition liegt darin, dass alle Erkenntnisse aus dem Bereich der Funktionen direkt für Reihen spezialisiert werden können. Eine Reihe ist z. B. genau dann streng monoton wachsend, wenn für alle $i \in \{1, \ldots, n-1\}$ gilt $x_i < x_{i+1}$. Eine Wertereihe ist also eine Folge von Paaren, wobei die erste Komponente jedes Paares eine reelle Zahl ist und die Lage des Elements in der Indexdimension angibt und die zweite Komponente den oder die Werte des Elements.

Definition 2.2 (Indexdimension) *Die erste Komponente d_i jedes Reihenelements $x_i = (d_i, \cdot)$ gibt die Position auf einer Zahlengeraden an und heißt* Indexkomponente*. Die Zahlengerade heißt* Indexdimension*.*

Normalerweise gilt für die Indexkomponenten einer Reihe, dass sie in der gleichen Weise geordnet sind wie die Indizes der Reihenglieder, also dass für beliebige $i, j \in \{1, \ldots, n\}$ und den dazugehörigen Indexkomponenten d_i und d_j gilt:

$$i < j \Leftrightarrow d_i < d_j.$$

Durch die Anwendung bestimmter Transformationen kann sich die Ordnung innerhalb der Indexdimension jedoch auch ändern. Die Indexdimension erlaubt zudem Reihen, bei denen die Werte nicht äquidistant vorliegen.

Es wird daher einige Male von der Abbildung $index : \mathbb{N} \to \mathbb{R}$ Gebrauch gemacht. Diese liefert für jedes Element der Reihe den Wert der Indexkomponente, in der oben benutzen Notation ist $index(i) = d_i$. Bei manchen Reihen sind die Werte einfach durchnummeriert, in anderen Fällen, insbesondere bei nicht äquidistanten Reihen, liegt jeder Wert zu einem festgelegten Punkt in der Indexdimension. Bei der Implementierung muss auf eine Datenverwaltung geachtet werden, die sowohl den i-ten Wert in einer Wertedimension als auch die i-te Indexkomponente in $O(1)$ liefern kann.

[1] Die hier verwendete Definition entspricht eher der mathematischen Definition einer *Folge*. Eine mathematische Reihe ist definiert als Folge der Partialsummen einer Folge. Es hat sich allerdings der Begriff Werte*reihe* statt Werte*folge* eingebürgert.

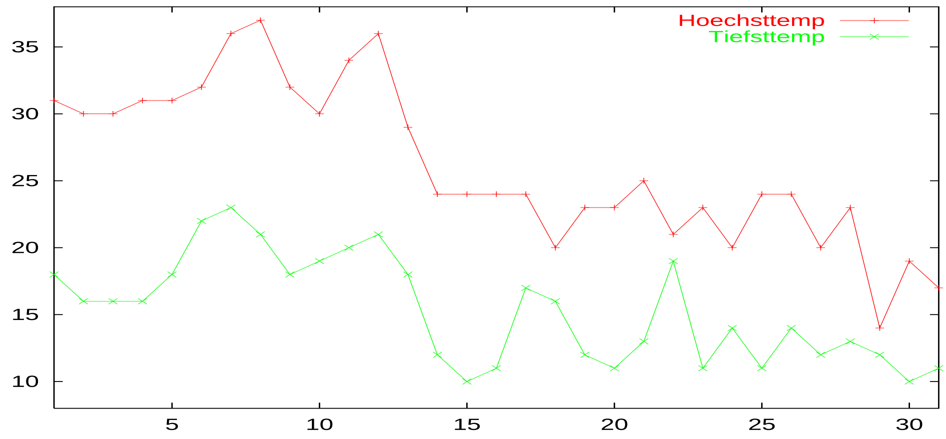


Abbildung 2.1.: Die Werte für Höchst- und Tiefsttemperatur der Stadt Dortmund im August 2003.

Definition 2.3 (Wertedimensionen) *Die zweite Komponente* $\mathbf{w}_i$ *jedes Reihenelements* $x_i = (\cdot, \mathbf{w}_i)$ *entspricht dem Wertevektor für jedes Element. Der Raum dieser Werte heißt* WERTEDIMENSIONEN.

Wir benutzen für die Wertedimensionen von Beginn an den m-dimensionalen komplexen Zahlenraum im multivariaten Fall und den Raum der komplexen Zahlen im univariaten Fall, also für $m = 1$. Verschiedene Transformationen machen Gebrauch von komplexen Zahlenräumen, auch wenn die vorliegenden Daten ursprünglich nicht komplex sind. Zum Abschluss noch zwei einfache Beispiele von Reihen:

Beispiel 2.1 *Eine Wertereihe, die für jeden Tag des Monats August 2003 die Werte für die höchste und die tiefste Temperatur in Dortmund-Wickede enthält, ist eine multivariate Zeitreihe der Länge* 31. *Abbildung 2.1 zeigt die Temperaturen des betreffenden Monats in Dortmund.*

Beispiel 2.2 *Die Wasser- und Schifffahrtsverwaltung des Bundes misst täglich den Wasserstand aller Flüsse Deutschlands. Die Wasserstände des Rheins sind für alle großen Städte abrufbar. Betrachten wir nun die Wertereihe für die Städte Mainz, Koblenz, Bonn, Köln und Düsseldorf und ordnen jeder Stadt die Pegelstände des 01.01.2003 sowie die Höchsttemperatur der Städte zu, so haben wir eine multivariate räumliche Wertereihe der Länge* 5. *Die räumliche Ordnung ergibt sich durch die Reihenfolge, in der die Städte am Rhein liegen. Die Indexkomponenten entsprechen z. B. den Flusskilometern und die Wertedimensionen stehen für Wasserstand und Temperatur.*

2.2. Merkmalsextraktion ermöglicht Lernen

Sei L_E von der Gestalt, dass eine Instanz $e \in L_E$ für jeden der n Zeitpunkte einer Zeitreihe ein Merkmal besitzt, jede Instanz also eine Wertereihe repräsentiert. Ein Lernverfahren liefert auch in diesem Fall für eine Menge von Instanzen eine Hypothese. Jedoch verschenkt man auf diese Weise Information, da eine inhärente Eigenschaft von Reihen nicht beachtet wird: Die Werte liegen geordnet vor [42]. Sei L'_E eine Sprache, deren Elemente diese Information berücksichtigen. Gesucht ist dann eine Transformation von L_E zu L'_E [37]. Dieser Vorgang stellt die Extraktion aussagekräftiger und robuster Merkmale dar. Sie sollten invariant sein gegenüber Translationen in der Zeit und verschiedene Versionen oder Qualitäten eines Musikstückes sollten zumindest nicht qualitativ zu unterschiedlichen Ausprägungen der Merkmale führen.

2.2.1. Musikstücke als Zeitreihen

Schallwellen werden durch elastische Schwingungen fester oder gasförmiger Körper erzeugt und an die umgebene Luft übertragen. Diese wird zu erzwungenen Schwingungen angeregt, bis die Wellen das menschliche Ohr erreichen. Die gezupfte Saite oder die Schwingung der Luftsäule in einer Flöte erzeugt eine solche Schallwelle; die Membran einer Lautsprecherbox simuliert mit ihrer Bewegung die Schwingung natürlicher Instrumente. Musikstücke sind also akustische Wellen und damit reellwertige Funktionen der Zeit. Um ein Lied zu digitalisieren wird die Welle in sehr viele kleine Abschnitte zerstückelt. Dies approximiert die kontinuierliche Funktion, wobei die Güte der Approximation und somit die Qualität der Digitalisierung von der Anzahl der Abschnitte abhängt (*sampling rate* [Hz]).

Definition 2.4 *Ein Musikstück wird während des Digitalisierungsprozesses in viele kleine Stücke zerteilt (*SAMPLING*). Die Anzahl der Teilstücke wird bestimmt durch die* SAMPLING RATE *[Hz], je größer die sampling rate desto besser ist die Qualität der Digitalisierung. Die kleinsten noch unterscheidbaren Zeiteinheiten eines digitalen Musikstückes sind die* SAMPLE POINTS*, also die Stellen, an denen eine Abtastung des Musikstückes vorgenommen wurde.*

Jedem dieser Zeitpunkte (*sample points*) eines Liedes wird die *Elongation* zu ebendiesem Zeitpunkt des Musikstückes zugeordnet:

Definition 2.5 (Elongation) *Die* ELONGATION *gibt die momentane Schwingungsweite wieder, also die momentane Entfernung von der Ruhelage einer Schwingung.*

Die Elongation ist nicht zu verwechseln mit der Amplitude, welche die maximale Auslenkung von der Ruhelage wiedergibt. Sie ist proportional zu der Auslenkung einer Lautsprechermembran beim Abspielen des Musikstückes. Die Diskretisierung der akustischen Welle erlaubt die Betrachtung eines Musikstückes als univariate Zeitreihe. Zu jedem Zeitpunkt i ist die Elongation x_i datiert. Abbildung 2.2 auf der nächsten Seite zeigt eine solche Wellenform. Auf der x-Achse ist der Zeitverlauf des Liedes dargestellt und auf der y-Achse die jeweilige Elongation.

Sei L_E von der Gestalt, dass eine Instanz $e \in E$ für jeden der n Zeitpunkte einer Zeitreihe ein Merkmal besitzt. Damit ist jedes Musikstück eine Instanz aus E. Angenommen, wir wollen nun Lieder mit einer Länge von drei Minuten klassifizieren. Wie viele Merkmale hat jede Instanz aus E? Die sampling rate der Wellenform in Abbildung 2.2 ist 44100 Hz, jeder sample point enthält die Daten für die beiden Stereokanäle [46]. Wir fassen beide Kanäle zu einem Monokanal zusammen und erhalten für ein Lied von drei Minuten

$$44100Hz \cdot 180s \approx 8 \cdot 10^6$$

Werte! Benutzt man diese Werte als Merkmale für die Instanzen, stellen sich mehrere Probleme:

1. Nicht alle Musikstücke haben die gleiche Länge, die Anzahl der Merkmale ist unterschiedlich für die Instanzen.
2. Die Laufzeit der Lernverfahren steigt mit der Zahl der Merkmale, die Performanz vermindert sich häufig (*curse of high dimensionality* [17]).

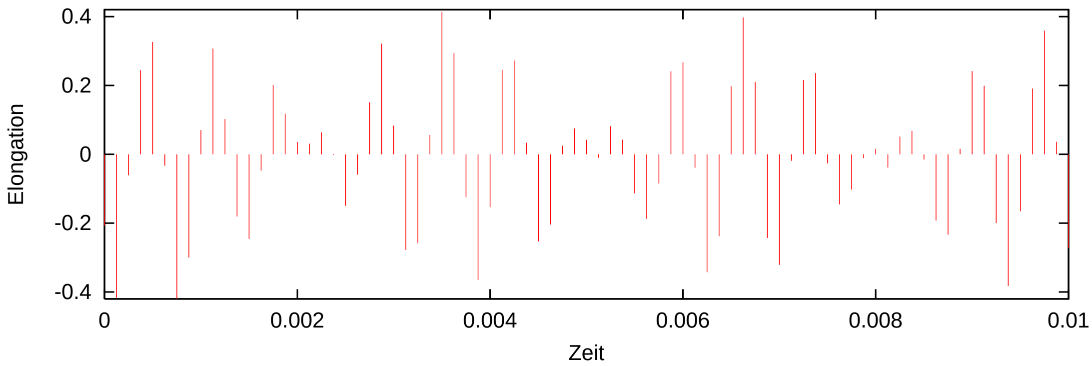


Abbildung 2.2.: Beispielhafte Wellenform eines Liedausschnittes. Die äquidistanten Samples sind deutlich zu erkennen.

3. Es gibt keinen Zusammenhang der Merkmale zwischen den Instanzen. Die Verschiebung eines Liedes um nur eine Sekunde ergibt einen vollständig anderen Merkmalsvektor.

Die ersten beiden Probleme ließen sich durch eine Beschränkung auf einen Bruchteil der Daten eindämmen, indem man nur Ausschnitte aus den Stücken betrachtet. Dies stellt allerdings keine befriedigende Lösung dar, da die Auswahl von guten Ausschnitten derzeit nicht gewährleistet werden kann. Außerdem besitzen die einzelnen Zeitpunkte der Ausschnitte immer noch keine Korrelation zu den entsprechenden Zeitpunkten der Ausschnitte der übrigen Lieder. Es ergibt sich also die Notwendigkeit, aus dieser hohen Anzahl von Daten wenige – aber aussagekräftige – Merkmale zu generieren. Diese sollen invariant gegenüber eine Translation in der Zeitdimension sein und möglichst robust gegenüber Rauschen und verschiedener Versionen des gleichen Musikstückes. Damit stellt die Extraktion der Merkmale auch eine Kompression der ursprünglichen Datenmenge dar.

2.3. Raum der Methoden

Die Änderung der Repräsentationssprache L_E geschieht mit Hilfe der Merkmalsextraktion. Das Ergebnis der Vorverarbeitung und Merkmalsgewinnung soll ein Merkmalsvektor sein, der es einem Lernverfahren erlaubt, eine möglichst einfache und zuverlässige Entscheidung treffen zu können. Die Vorverarbeitung dient dabei der Reduktion von Störeinflüssen oder dem Herausarbeiten von Merkmalen.

Durch die Merkmalsextraktion wird der Merkmalsraum festgelegt, auf dem das Lernverfahren arbeitet. Findet keine Extraktion statt, so besitzen die Merkmalsvektoren genauso viele Elemente wie die vorliegende Reihe. Ist dieser Merkmalsraum einmal festgelegt, so kann die Dimension durch eine Selektion des besten Merkmale [26] reduziert werden oder der Raum wird erweitert durch die Generierung neuer Merkmale aus den bereits vorhandenen Merkmalen [45]. Beides kann dir Performanz eines Lernverfahrens entscheidend erhöhen. In dieser Arbeit wird nur der Vorgang der Merkmalsextraktion aus Wertereihen näher behandelt. Merkmalsselektion und -generierung werden im Rahmen der Experimente zwar angewendet, aber hier nicht weiter besprochen.

Eine häufige Anwendung der statistischen Zeitreihenanalyse ist die Betrachtung von Wirtschaftsdaten. Es verwundert daher nicht, dass die Komponenten des *klassischen Komponentenmodells* der Statistik [49] mit ökonomischen Begriffen benannt werden. Nach diesem Modell bestehen Wertereihen aus vier Komponenten, welche sich additiv überlagern:

Trend: langfristige, systematische Veränderung des mittleren Niveaus

Konjunkturkomponente: langfristige Schwankung, nicht unbedingt regelmäßig

Saison: wiederholende, gleichbleibende Schwankungskomponente

Restkomponente: Rauschen, d. h. nichtperiodische Störeinflüsse

Bei Musikdaten wie auch bei vielen anderen Datensätzen ist dieses Modell jedoch kaum anwendbar. Es gibt keinen Trend in der Zeitreihe eines Musikstückes. Da die Werte auf der y-Achse die Elongation der Schwingung des Musikstückes zu jedem Zeitpunkt wiedergibt, ist eine dauerhafte systematische Veränderung nicht möglich. Wir erinnern uns, dass die Elongation der Auslenkung einer Lautsprechermembran entspricht. Hätten Musikdaten einen Trend, würde dieser zur Zerstörung der Membran führen. Solange man die Wertereihe eines Musikstückes nicht in kleine Teile unterteilt und jeden Teil für sich betrachtet, gibt es auch keine Konjunkturkomponente innerhalb eines Musikstückes. Zu guter Letzt weisen Musikdaten jedoch einen großen Vorteil gegenüber anderen Daten wie z. B. Messdaten eines Experimentes auf: Damit man Audiodaten noch als hörbar bezeichnen kann, ist auch der Anteil des Rauschens deutlich geringer als der Anteil der mit Absicht erzielten Schwankungen. Interessant sind also vor allem diejenigen Methoden, die saisonale Schwankungen finden sollen.

Es wird daher weitgehend auf Techniken zum Finden eines Trends verzichtet. Auch die weitverbreiteten ARMA-Modelle[2] finden keine Anwendung. Bei diesen Modellen geht es um die Vorhersage von zukünftigen Daten einer Wertereihe, hier jedoch sollen Merkmale für Klassifikationsaufgaben extrahiert werden. Die Statistik und auch die Physik haben eine große Menge an Methoden der Zeitreihenanalyse entwickelt, die zur Extraktion von Merkmalen geeignet sind. Allerdings leiden diese darunter, dass viele Spezialfälle existieren und keine einheitliche Theorie, die diese zusammenfasst. Damit ist eine Automatisierung der Merkmalsextraktion nahezu unmöglich. Speziell bei der Analyse von Audiodaten gibt es viele alte Verfahren mit neuem Namen. Das Ziel dieser Arbeit wird sein, eine Systematik der Methoden der Wertereihenanalyse zu entwickeln mit deren Hilfe ein Ansatz vorgestellt werden kann, der sowohl eine einfache Datenanalyse von Menschenhand wie auch die automatisierte Merkmalsextraktion mit Hilfe von genetischer Programmierung erlaubt.

2.3.1. Systematik der Analysemethoden

Betrachtet man alle Methoden der Reihenanalyse und versucht diese in Gruppen einzuteilen, so stellt sich die Frage nach den Kriterien der Einteilung. Eine Systematisierung muss mächtig genug sein, um alle bekannten (und auch zukünftigen) Methoden zur Merkmalsextraktion zu umfassen. Sie muss so weitreichend sein, dass eine Automatisierung der Merkmalsextraktion ermöglicht wird. Dabei sollten die eingeordneten Methoden keine Redundanz besitzen. Das gleiche Merkmal sollte also nicht durch verschiedene Kombinationen von Methoden erzeugbar sein. Die Forderung nach der Redundanzfreiheit ist allerdings nicht zwingend. Ist die Systematisierung nicht völlig frei von Redundanz, so dauert eine Automatisierte Suche nach den optimalen Methoden unter Umständen länger und die durch redundante Methoden erzeugbaren Merkmale werden bevorzugt extrahiert. Redundanz kann aber dadurch vermieden werden, redundante Kombinationen von Methoden nicht zuzulassen.

[2] *Autoregressive Moving Average*

Zunächst wird definiert, was unter einer Methode der Reihenanalyse verstanden wird. Diese Definition erinnert an die Definition eines Operators, der eine Eingabe erwartet, seine Aktion darauf ausführt und eine Ausgabe eines bestimmten Typs erzeugt. Der Begriff Operator wird jedoch später für die Implementierung der Methoden innerhalb der Lernumgebung YALE noch verwendet. Um zwischen der mathematischen Methode und ihrer technischen Implementierung besser unterscheiden zu können, wurden die beiden Begriffe „Methode“ für das abstrakte Vorgehen und „Operator“ für eine konkrete Implementierung gewählt.

Definition 2.6 (Methode) *Eine* METHODE *der Wertereihenanalyse bekommt als Eingabe eine Wertereihe und führt eine Operation auf ihr aus. Das Ergebnis der Operation ist die Ausgabe der Methode.*

Alle Methoden arbeiten also auf Wertereihen. Die erste Möglichkeit, die Methoden anzuordnen besteht in der Betrachtung der Operation der Methoden. Es fällt jedoch schnell auf, dass bei der Vielzahl von Methoden keine gemeinsamen Ansätze innerhalb der Berechnungen existieren, die eine Einteilung in Gruppen rechtfertigen würde. Eine andere mögliche Systematisierung der Methoden benutzt die Ausgaben der Methoden, um diese in Gruppen einzuteilen. Dies bietet sich für eine Automatisierung an, da nur aufgrund des Ein- und Ausgabeverhaltens der Methoden bestimmt werden kann, ob sie miteinander kombiniert werden können. Dieser Ansatz ist, wenn auch in anderem Zusammenhang, bereits erfolgreich verfolgt worden in [12, 33]. Eine Systematisierung gemäß der Eingabe der Methoden ist sinnlos, da alle Methoden auf Wertereihen arbeiten und sich in ihrer Eingabe folglich nicht unterscheiden.

Die Menge aller Methoden werden in verschiedene Gruppen unterteilt, je nach ihrer Ausgabe:

Definition 2.7 (Transformationen) *Alle Methoden, die als Ausgabe eine (andere) Wertereihe liefern, heißen* TRANSFORMATION*.*

Transformationen ändern also die Wertereihe selbst und generieren keine einzelnen Merkmale. Dies kann dazu dienen, Merkmale überhaupt erkennen oder herausarbeiten zu können. Die Gruppe wird weiter unterschieden in *Basistransformationen* (Kapitel 3), *Filter* (Kapitel 4), und *Auszeichner* (Kapitel 5). In Kapitel 6 wird eine neue Gruppe von Transformationen betrachtet, welche aus Fensterungen der Wertereihe in Kombination mit anderen Transformationen und den Funktionalen aus Kapitel 7 gebildet werden.

Liefert eine Methode keine Reihe, sondern einzelne Zahlen ohne eine Ordnung, so nennt man diese Methode *Funktional*:

Definition 2.8 (Funktional) *Alle Methoden, die als Ausgabe einzelne, ungeordnete Zahlen liefern, heißen* FUNKTIONAL*.*

Funktionale dienen uns dazu, die eigentlichen Merkmale zu extrahieren. Wie oben beschrieben, benutzen wir als Eingabe für ein Lernverfahren einen Vektor von Merkmalen. In der Regel werden die Werte dieser Merkmale berechnet durch die Anwendung eines Funktionals. In Kapitel 7 geben wir eine genauere Definition und zahlreiche Beispiele möglicher Funktionale.

Abbildung 2.3 auf der nächsten Seite zeigt den Raum der Methoden. Funktionale und Transformationen unterscheiden sich durch den Typ ihrer Ausgabe. Beispiele für Funktionale sind die Berechnung von Durchschnitten, des Minimums / Maximums und die Anwendung anderer einfacher statistischer Funktionen. Transformationen umfassen Basistransformationen in andere Räume, wie zum Beispiel einer schnellen Fourier-Transformation oder die Anwendung eines

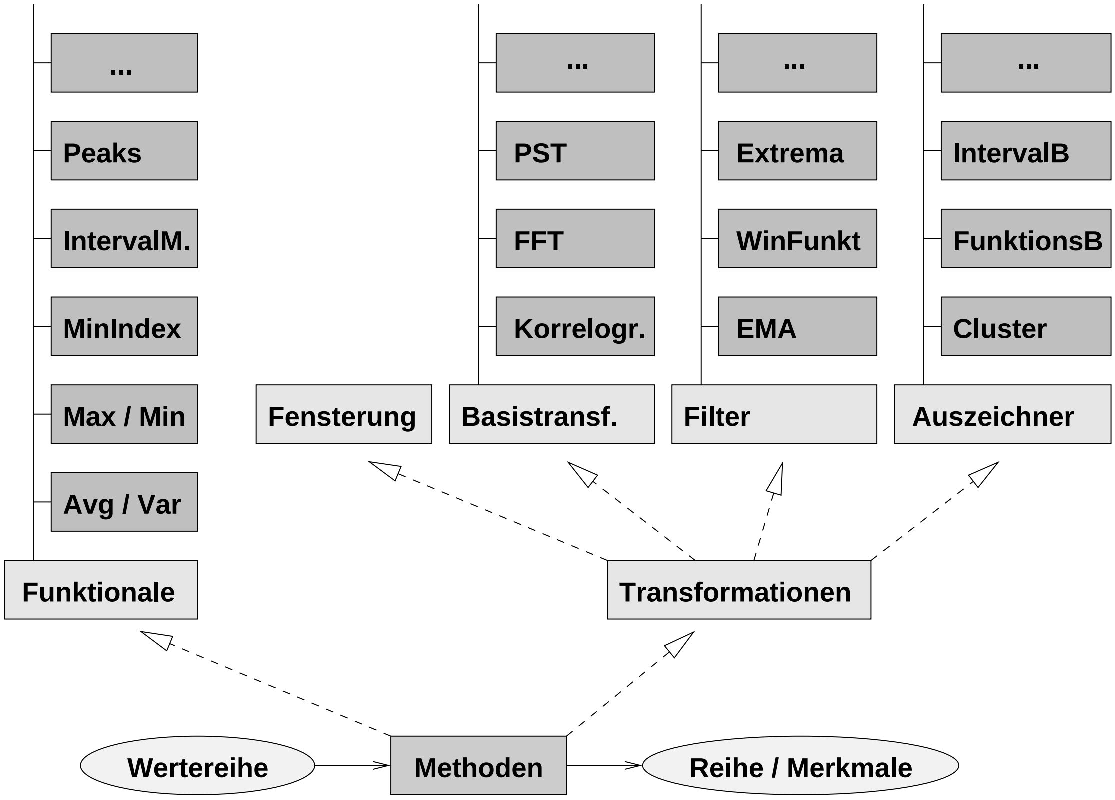
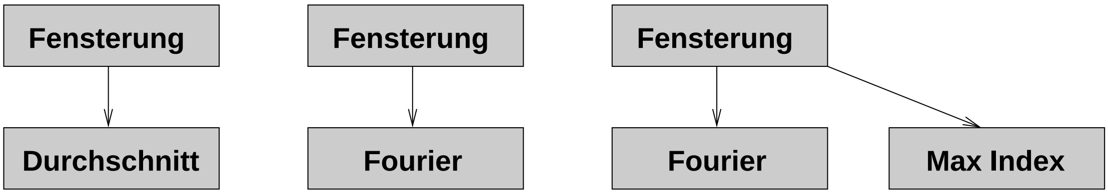


Abbildung 2.3.: Systematisierung der Methoden zur Wertereihenanalyse. Je nach Ausgabeverhalten der Methoden wird unterschieden zwischen Transformationen und Funktionalen. Die Gruppe der Transformationen wiederum kann unterschieden werden in Basistransformationen, Filter, Auszeichner und Fensterung. Für jede Gruppe sind einige Beispiele aufgezählt.



(a) Funktional (b) Transformation (c) Kombination

Abbildung 2.4.: Die Transformation „Fensterung“ in Kombination mit anderen Methoden der Systematik. Die Einbettung eines Funktionals simuliert Filter wie zum Beispiel den gleitenden Durchschnitt. Transformationen hingegen können den Datensatz stark vergrößern. Ein guter Kompromiss ist die Kombination von eingebetteten Transformationen und einem Funktional oder die Beschränkung der Transformationen auf Filter und der Verzicht auf Überlappung.

Filters. Auszeichner schließlich liefern die Eingabewertereihe, angereichert durch Bereiche, denen verschiedene Eigenschaften zugewiesen wurden. Die hier behandelten Auszeichner finden Intervalle in verschiedenen Dimensionen der Reihe.

In den Kapiteln 3 bis 7 werden die Methoden der einzelnen Gruppen genau definiert und alle Methoden auf ihre Laufzeit untersucht. Dadurch ist die in Kapitel 8 vorgestellte automatisierte Merkmalsextraktion möglich.

2.3.2. Fensterung als Basis neuer Methoden

Damit sämtliche aus der Literatur bekannten Methoden zur Merkmalsextraktion aus Wertereihen in diese Systematik eingeordnet werden können, muss eine besondere Transformation genauer betrachtet werden: Die Fensterung (engl.: *Windowing*). Mit ihrer Hilfe kann eine ganze Schar von Methoden erschlossen werden. Dabei wird ein Zeitfenster über die Reihe geschoben und in jedem Fenster dieselbe Berechnung durchgeführt. So kennt die Statistik schon lange gleitende Durchschnitte (*moving averages*, siehe Abschnitt 4.1); doch gab es in den letzten Jahren auch vermehrt Ansätze, Fourier-Transformationen u. ä. in Fenstern durchzuführen. Auch diese Verfahren lassen sich innerhalb des Raums der Methoden erklären. Die allgemeine Fensterung, die in dieser Arbeit vorgestellt wird, umfasst sowohl Basistransformationen als auch Filter. Sie wird in Kapitel 6 näher untersucht.

Abbildung 2.4 zeigt mehrere Fensterungen. Die erste Fensterung bildet in jedem Fenster einen Durchschnitt, wodurch erneut eine Reihe entsteht. Die zweite Fensterung transformiert jeden Ausschnitt der Reihe mit einer Fourier-Transformation. Dadurch kann eine dreidimensionale Reihe entstehen, da für jedes Fenster nicht nur ein einzelner Wert, sondern eine ganze Reihe entsteht. Das letzte Beispiel zeigt eine allgemeine Fensterung, die sowohl Transformationen als auch Funktionale in jedem Fenster anwenden kann.

Durch dieses Konzept der allgemeinen Fensterung können alle bereits bekannten Methoden zur Merkmalsextraktion aus Wertereihen in die vorgestellte Systematisierung eingeordnet werden. Kapitel 6 beschäftigt sich eingehend mit dieser abstrakten Form der Fensterung. Sie bildet einen

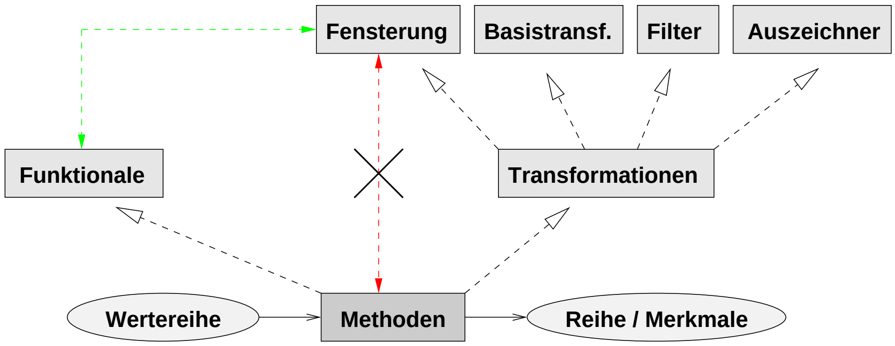


Abbildung 2.5.: Eine Fensterung kann sowohl bekannte Methoden simulieren als auch neue Methoden schaffen. Prinzipiell sind beliebige Methoden in jedem Fenster denkbar, jedoch kann die Forderung nach einem Funktional das Aufblähen der Daten um eine zusätzliche Dimension vermeiden.

Rahmen zur Entwicklung weiterer und neuer Methoden, um umfangreiche Wertereihen so zu transformieren, dass eine Extraktion neuartiger Merkmale einfach ermöglicht wird.

2.3.3. (Automatisierte) Merkmalsextraktion

Es sind zwei Formen der Merkmalsextraktion für die Klassifikation von Wertereihen denkbar. Zunächst ist es möglich, von Hand zu definieren, welche Merkmale extrahiert werden sollen. Dies bietet sich an, wenn der Benutzer ein ausreichendes Wissen in der Domäne der vorliegenden Daten besitzt. Die Systematisierung kann die manuelle Analyse von Wertereihen erleichtern. Die in Kapitel 9 vorgestellte Lernumgebung YALE unterstützt diese Systematisierung und erlaubt so eine einfache Zusammenstellung von Methoden, um die optimalen Merkmale zu extrahieren. Manchmal ist auch der Vergleich von einzelnen Methoden interessant, so z. B. bei der Frage, welche Durchschnittsfunktion innerhalb eines gleitenden Durchschnittes für die vorliegenden Daten die beste ist. Durch die vorgestellte Hierarchie lassen sich Methoden leicht austauschen und ihre Performanz miteinander vergleichen.

Oftmals besitzt ein Anwender jedoch nicht das nötige Hintergrundwissen, um die richtige Auswahl und Zusammenstellung der Methoden zu gewährleisten. Das eigentliche Ziel der Systematisierung war, dass Verfahren automatisch die besten Methoden zusammenstellen und damit eine optimale automatisierte Vorverarbeitung und Merkmalsextraktion generiert wird. Alle Transformationen liefern Wertereihen, welche einen beachtlichen Umfang haben können und sich daher nicht als Merkmal eignen (vergleiche Abschnitt 2.2.1). Am Ende eine Kette von Transformationen steht daher meist ein Funktional, welches das eigentliche Merkmal liefert. Mehrere solcher Ketten liefern Merkmale für eine gegebene Reihe. Etwas ähnliches gilt auch für die Fensterung: Innerhalb jedes Fensters können einzelne Werte mit einem Funktional berechnet werden, eine neue Wertereihe mit einer Transformation gebildet oder auch eine Kombination aus Transformationen und einem Funktional angewendet werden. Es werden also beliebige Methoden innerhalb jeden Fensters erlaubt. Dies kann jedoch dazu führen, dass die Datenmenge „aufgebläht“ wird, indem eine Fensterung mit einer eingebetteten Transformation der gegebenen Wertereihe eine neue Dimension hinzufügt. Um dieses zu verhindern, kann für eine automatisierte Merkmalsextraktion eine Forderung nach einem Funktional in jedem Fenster gestellt werden. Zusätzlich werden beliebig viele Transformationen in jedem Fenster vor Berechnung des Funktionals erlaubt. Abbildung 2.5 veranschaulicht diese Erweiterung der Fensterung.

Später in Kapitel 6 werden wir sehen, wie die Erweiterung auf beliebige Methoden in Fenstern mit einer gleichzeitigen Einschränkung durch verschiedene Bedingungen bei der Auswahl der Methoden eine Vielzahl neuer Methoden liefert, die noch dazu automatisch generiert und optimiert werden. Dabei erweist es sich als günstig, dass alle in dieser Arbeit vorgestellten Methoden effizient arbeiten, die meisten mit einer Laufzeit von $O(n)$ oder $O(n \log n)$ und einige wenige besitzen im schlimmsten Fall eine Laufzeit von $O(n^2)$. Auf diese Weise sind alle hier vorgestellten Methoden auch auf großen Datenmengen einsetzbar.

2.3.4. Audiodaten und andere Datensätze

Wie in der Einleitung bereits erwähnt wurde, hat die Beschäftigung mit Dateien im Midi-Format und auch die Analyse von gesprochener Sprache bereits Merkmale aus Audiodaten hervorgebracht. Spezielle Methoden auf Audiodaten sind allerdings häufig „getarnte" Methoden der Wertereihenanalyse. Die vorliegende Arbeit hat zum Ziel, die verwendeten Zeitreihenmethoden und ihre zugrundeliegende Hierarchie als universelles und umfassendes Konzept zur Merkmalsextraktion aus Zeitreihen darzustellen. Daher wird weitestgehend bewusst auf Audiomerkmale verzichtet. Durch dieses Vorgehen ist gewährleistet, dass die Methoden auch auf anderen Daten mit ähnlichen Eigenschaften zu akzeptablen Resultaten führen.

Um das Verständnis zu erhöhen und die Bedeutung einiger Merkmale zu verdeutlichen, wird immer wieder exemplarisch auf den zugrundeliegenden Musikdatensatz verwiesen. So ist es interessant, dass die Kombination einiger Methoden komplexe Merkmale wie Aufbauschemata von Liedern, Geschwindigkeit oder Perkussivität erzeugen kann. Wertereihen, die auf Musikdaten basieren, besitzen dabei die folgenden Eigenschaften. Sie sind

- endlich
- univariat
- und äquidistant.

Andere Daten mit diesen Eigenschaften sollten ähnlich gut bearbeitbar sein. Inwieweit sich Rauschen auf die Qualität der Merkmale und damit auf die Ergebnisse eines Lernverfahrens auswirkt, wird experimentell in Kapitel 9 untersucht.

KAPITEL 3

Basistransformationen

Nachdem die Notwendigkeit einer Merkmalsextraktion gezeigt und die zugehörigen Methoden systematisiert wurden, werden in den nächsten Kapiteln die Methodengruppen genauer definiert und Methoden aus den jeweiligen Gruppen vorgestellt. Gleichzeitig werden alle Methoden auf ihre Effizienz überprüft. In diesem Kapitel liefern wir die mathematische Grundlage für eine Untergruppe der Transformationen, die mathematisch durch einen Wechsel der Basis des betrachteten Raumes formalisiert ist. Als für Musikdaten naheliegendes Beispiel wird die Basistransformation vom Zeit- in den Frequenzraum intensiver betrachtet.

3.1. Basisvektoren und Basissysteme

Basistransformationen entsprechen Abbildungen von einem Raum in einen anderen. Der Gehalt an Information wird hierdurch nicht geändert, jedoch kann die Erkennung von Merkmalen in einem bestimmten Raum leichter sein. Die Frage, die sich also zunächst stellen sollte, ist die nach der Grundlage der Räume. Dabei wird vorausgesetzt, dass die vorliegenden Daten Elemente eines Vektorraums sind, im Falle der Musikdaten also Elemente des $\mathbb{R}^2$.

Definition 3.1 (Basis) *Sei V ein Vektorraum und B eine Menge von Vektoren aus V. B heißt dann eine* BASIS *von V, wenn jeder Vektor aus V genau eine Darstellung als Linearkombination der Vektoren aus B besitzt.*

Die Basis eines Vektorraumes kann also dazu benutzt werden, um die Vektoren des betrachteten Vektorraumes darzustellen. So ist es uns geläufig, einen Vektor des $\mathbb{R}^3$ mit Hilfe der kanonischen Basis der drei Einheitsvektoren zu bezeichnen. Der Vektor wird dabei auf die Basisvektoren projiziert und durch die entsprechenden Komponenten dargestellt. Die Anzahl der Basisvektoren stimmt mit der Zahl der Dimensionen überein. Wir gehen davon aus, dass die hier betrachteten Basen *orthonormal* sind, d. h. dass je zwei Basisvektoren senkrecht aufeinander stehen und alle eine normierte Länge von 1 aufweisen.

Im letzten Satz haben wir stillschweigend über Abstände und Winkel geredet ohne die dazu notwendige Formalisierung betrachtet zu haben. Wir stellen in diesem Zusammenhang die einzige Forderung an unseren Raum: Er muss ein Skalarprodukt besitzen.

Definition 3.2 (Skalarprodukt) *Sei V ein Vektorraum und $\langle \mathbf{x}, \mathbf{y} \rangle : V \times V \to \mathbb{R}$ mit $\mathbf{x}, \mathbf{y} \in V$ eine Abbildung mit den folgenden Eigenschaften:*

1. $\langle \mathbf{x}_1 + \mathbf{x}_2, \mathbf{y} \rangle = \langle \mathbf{x}_1, \mathbf{y} \rangle + \langle \mathbf{x}_2, \mathbf{y} \rangle$
2. $\langle \mathbf{x}, \mathbf{y}_1 + \mathbf{y}_2 \rangle = \langle \mathbf{x}, \mathbf{y}_1 \rangle + \langle \mathbf{x}, \mathbf{y}_2 \rangle$
3. $\langle \lambda\mathbf{x}, \mathbf{y} \rangle = \lambda\langle \mathbf{x}, \mathbf{y} \rangle$
4. $\langle \mathbf{x}, \lambda\mathbf{y} \rangle = \lambda\langle \mathbf{x}, \mathbf{y} \rangle$
5. $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$
6. $\mathbf{x} \neq \mathbf{0} \Rightarrow \langle \mathbf{x}, \mathbf{x} \rangle > 0$

Dann heißt die Abbildung $\langle \mathbf{x}, \mathbf{y} \rangle$ SKALARPRODUKT.

Beispiel 3.1 *Für Vektoren* $\mathbf{x}$ *und* $\mathbf{y}$ *des* $\mathbb{R}^n$ *erfüllt die folgende Abbildung die in Definition 3.2 gestellten Bedingungen:*

$$\langle \mathbf{x}, \mathbf{y} \rangle := \sum_{k=1}^{n} x_k y_k$$

Der Nachweis erfolgt durch Rückführungen der genannten Bedingungen auf Eigenschaften wie Assoziativität, Kommutativität und Distributivität der verwendeten Grundrechenarten.

Mit Hilfe des Skalarproduktes ist es nun möglich, die Länge eines Vektors sowie die Orthogonalität zweier Vektoren zu definieren. Man kann sich den Sinn der folgenden Definitionen leicht am obigen Beispiel klar machen.

Definition 3.3 (Norm) *Die* NORM $||\mathbf{x}||$ *eines Vektors* $\mathbf{x}$ *erfüllt die folgenden Eigenschaften:*

1. $||\mathbf{x}|| \geq 0$
2. $||\lambda \cdot \mathbf{x}|| = |\lambda| \cdot ||\mathbf{x}||$
3. $||\mathbf{x} + \mathbf{y}|| = ||\mathbf{x}|| + ||\mathbf{y}||$
4. $\mathbf{x} = \mathbf{0} \Rightarrow ||\mathbf{x}|| = 0$

Die Norm bildet die Grundlage zur Abstandsmessung. In den betrachteten Räumen kann die Norm durch das Skalarprodukt induziert werden mit $||\mathbf{x}|| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$. Da das Skalarprodukt $\langle \mathbf{x}, \mathbf{y} \rangle$ auch als Projektion des Vektors $\mathbf{x}$ auf den Vektor $\mathbf{y}$ aufgefasst werden kann, sollte das Skalarprodukt zwischen zwei aufeinander senkrecht stehenden Vektoren 0 ergeben.

Definition 3.4 (Orthogonalität) *Sei* V *ein Vektorraum und* $\mathbf{x}, \mathbf{y} \in V$. *Die Vektoren* $\mathbf{x}$ *und* $\mathbf{y}$ *heißen* ORTHOGONAL *zueinander, wenn* $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ *gilt.*

Fassen wir nun noch einmal unsere Anforderungen zusammen: In dieser Arbeit werden Transformationen in Räumen mit Skalarprodukt behandelt, denen eine orthonormale Basis zugrundeliegt. In einem solchen Raum bleibt ein Element stets an einer bestimmten Stelle, unabhängig davon mit welchem Basissystem man es beschreibt. Wie erhält man aber die Darstellung eines Vektors in einem bestimmten Basissystem? Jeder Vektor $\mathbf{x}$ eines n-dimensionalen Raumes besteht ja aus Komponenten $x_1, \ldots, x_n$. Für die kanonische Basis des $\mathbb{R}^3$ gibt jede der drei Komponenten den Anteil des jeweiligen Basisvektors an der Linearkombination zur Darstellung des Vektors an. Diese Idee wird nun verallgemeinert:

Definition 3.5 *Die* i*-te Komponente* x_i *(*KOEFFIZIENT*) eines Vektors* $\mathbf{x}$ *entspricht der Länge der Projektion von* $\mathbf{x}$ *auf den* i*-ten Basisvektor.*

Die Länge der Projektion eines Vektors auf einen anderen ist jedoch nichts anderes als das bereits bekannte Skalarprodukt. Um den Anteil eines Basisvektors $\mathbf{e}$ an einem Element $\mathbf{x}$ zu ermitteln, genügt also die Berechnung von $\langle \mathbf{x}, \mathbf{e} \rangle$.

3.2. Funktionenräume

Basistransformationen sind Abbildungen von Elementen in einen neuen Raum mit anderer Basis. Häufig gilt, dass eine Basistransformation nicht das Element selbst ändert, sondern lediglich die Darstellung des Elements in dem Raum auf eine andere Weise geschieht. In diesem Fall ist die Transformation bijektiv. Später wird mit Hilfe der Intervalltransformation aber auch ein nichtreversibler Wechsel der Basis durchgeführt, bei dem die Elemente nicht rekonstruiert werden können.

Bei einem Wechsel der Basis ändern sich natürlich die Koeffizientendarstellungen der Elemente, daher darf man nun nicht mehr Elemente (Vektoren) einfach mit ihren Koordinatenvektoren identifizieren, sondern muss zudem die verwendete Basis nennen. Man spricht z. B. von einer Reihe im Zeitraum oder im Frequenzraum. Neben einer Vielzahl naiver Beispiele von Basistransformationen sind die Abbildungen in eine „fremdartige“ Basis von großem Interesse, da auf diese Weise eventuell neue Merkmale erschlossen werden können. Primäres Ziel ist es, eine Transformation in den Raum der harmonischen Schwingungen durchzuführen, die sogenannte Fourier-Transformation.

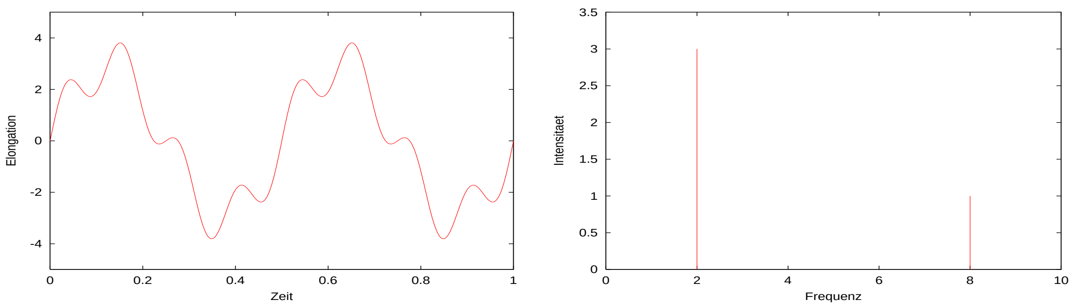
Bemerkung 3.1 *Fourier beschrieb, wie man eine Funktion f als Überlagerung (Superposition) aus Sinusschwingungen mit unterschiedlichen Frequenzen ν_k, Amplituden a_k und Phasen φ_k darstellen kann [52]:*

$$f(x) := \sum_{k=1}^{\infty} a_k \cdot \sin(\nu_k x + \varphi_k)$$

Der Vorgang der Erzeugung wird auch als Fourier-Synthese, der umgekehrte Vorgang als Fourier-Analyse bezeichnet. Die harmonischen Schwingungen sind damit die wichtigste Klasse periodischer Funktionen, da auch sehr komplizierte und nicht-sinusförmige periodische Funktionen auf diese zurückgeführt werden können.

Allgemein spricht man von *Fourier-Transformationen* stets dann, wenn die Basis, von der man ausgeht, oder die Zielbasis aus allen denkbaren Sinuskurven besteht. Dies ist eine neue Art von Raum, denn bisher haben wir in Gedanken wohl stets endliche Räume mit reellwertigen Basen vor Augen gehabt. Es gibt jedoch unendlich viele verschiedene Sinusschwingungen mit unendlich vielen Frequenzen und Phasen. Zudem besteht der Zielraum und somit natürlich auch die Basisvektoren nicht aus Zahlen, sondern aus Funktionen. Bei einer Basistransformation in diesen Raum der Sinusschwingungen erhalten wir für jede dieser unendlich vielen „Basisschwingungen“ den Anteil der Basisschwingung an der Gesamtfunktion.

Beispiel 3.2 *Wir betrachten die einfache Überlagerung von zwei Sinusschwingungen mit der Frequenz 2 Hz und der Amplitude 3 sowie der Frequenz 8 Hz und der Amplitude 1. In Abbildung 3.1(a) ist sie im Zeit-Raum dargestellt, d. h. man kann zu jedem Zeitpunkt der x-Achse die Elongation auf der y-Achse ablesen. Eine Transformation in den Frequenzraum würde für jede Frequenz liefern, wie stark diese vertreten ist, also ihre Intensität. In diesem Beispiel ist die Frequenz 2 Hz mit einer Amplitude von 3 vertreten und die Frequenz 8 Hz mit einer Amplitude von 1, alle anderen Frequenzen kommen gar nicht vor. Das Ergebnis der Transformation in 3.1(b) besteht aus zwei scharfen Peaks mit den entsprechenden Werten.*



(a) Zeit-Raum (b) Frequenz-Raum

Abbildung 3.1.: Eine Überlagerung von zwei Schwingungen mit $\nu_1 = 2$ Hz, $a_1 = 3$ sowie $\nu_2 = 8$ Hz, $a_2 = 1$. Die Abbildung rechts zeigt ihre Transformierte im Frequenzraum: Zwei Peaks bei 2 Hz und 8 Hz mit den entsprechenden Amplituden.

Das Konzept des besprochenen Vektorraums ist noch viel mächtiger. Wir werden im Folgenden sehen, dass man auch Funktionen als Elemente in einem Vektorraum ansehen kann. Bisher haben wir nur Vektorräume betrachtet, die endlich-dimensional sind. In diesem Fall ist die Zahl der Basisvektoren genau die Dimension des Vektorraums. Die Dimension kann aber auch abzählbar und sogar überabzählbar unendlich werden. Dies ist auch meist der Fall bei Räumen, deren Elemente Funktionen sind.

Definition 3.6 *Sei V ein Vektorraum mit Skalarprodukt. Sind die Elemente $f \in V$ Funktionen, so heißt V* FUNKTIONENRAUM.

Natürlich bestehen bei einem Funktionenraum nicht nur die Elemente, sondern auch die Basisvektoren aus Funktionen. Zusammen mit einem geeigneten Skalarprodukt sind auf diese Weise unterschiedliche Räume konstruierbar. In Analogie zum „normalen“ Skalarprodukt kann man als Skalarprodukt für einen Funktionenraum F das folgende wählen:

$$\langle f, g\rangle = NF \int f \cdot g$$

für zwei beliebige Funktionen $f, g \in F$. Anstelle der bekannten Projektion von zwei Vektoren entspricht dies einer Faltung der Funktion f mit der Funktion g. Der Normierungsfaktor NF sollte entsprechend gewählt werden.

Beispiel 3.3 *Wir erzeugen einen Funktionenraum P, deren Elemente sämtliche natürlichen Polynome in x darstellen:*

- *Die Basisvektoren sind gegeben durch alle x^n für $n \in \mathbb{N}$.*
- *Als Skalarprodukt benutzen wir $\langle f, g\rangle = \int\limits_{-\infty}^{\infty} f \cdot g$ für zwei Elemente $f, g \in P$.*

Das Element $f : f(x) = 5x^3 - 2x^2$ hat in P die Koordinaten $f_2 = 2$ und $f_3 = 5$, die übrigen Koeffizienten sind 0.

Die Einführung der Funktionenräume wird sich aber auch im Kapitel 7 über Funktionale noch als nützlich erweisen.

3.3. Fourier's Raum

Die Fourier-Synthese besagt, dass Funktionen als eine Überlagerung von harmonischen Schwingungen unterschiedlicher Frequenzen und Amplituden dargestellt werden kann. Mitunter sind hierzu unendlich viele verschiedene Frequenzen nötig. Es bietet sich also eine Darstellung in einem Funktionenraum an, bei dem die Basisvektoren durch eben genau diese unendlich vielen möglichen Schwingungen definiert sind. Jede Basisfunktion in diesem Funktionenraum entspricht genau einer Sinusschwingung mit einer bestimmten Frequenz, also ist die Basis durch alle $\sin \nu x$ mit $\nu \in \mathbb{R}^+$ gegeben. Um die Einführung einer Phase zu vermeiden und trotzdem Funktionen synthetisieren zu können, die nicht punktsymmetrisch zum Ursprung sind, bietet sich die Verallgemeinerung der Basisfunktionen zu

$$e^{i\nu x} \text{ mit allen } \nu \in \mathbb{R}^+$$

an. Hierbei ist $e^{i\nu x} = \cos \nu x + i \sin \nu x$ und i die Wurzel aus -1. Das Skalarprodukt schließlich wird durch

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} f \cdot g$$

definiert. Die Normierung mit $1/2\pi$ ist Konvention, man erzielt mit dieser Normierung genau die richtige Amplitude bei der Faltung mit $e^{i\nu x}$ für ein konkretes ν. Um nun eine Fourier-Transformation durchzuführen, behandelt man natürlich nicht die überabzählbar vielen möglichen Frequenzen aus $\mathbb{R}^+$ sondern vielmehr diskret ausgewählte, für die man jeweils die Faltung durchführt. Das Ergebnis ist dann ein Graph, der für die ausgewählten Frequenzen die Intensität der entsprechenden Grundschwingung an der Gesamtfunktion angibt: das *Frequenzspektrum*.

3.3.1. Eigenschaften der Fourier-Transformation

Eine geschickte Implementierung der Fourier-Transformation vermeidet die mehrfache Berechnung von Termen und erreicht eine Laufzeit von $O(n \log n)$. Diese Implementierung ist unter dem Namen *Fast Fourier Transformation* bekannt [9].

Bei der Anwendung von Fourier-Transformationen auf Wertereihen können die folgenden Erscheinungen auftreten:

Leakage: Liegt eine harmonische Schwingungkomponente der Frequenz ν vor, so erwartet man im Frequenzspektrum einen einzelnen Peak bei ν. In der Praxis zeigt sich zudem noch eine Erhöhung der umliegenden Frequenzen, was zu breiten Peaks führt. Dieses „Durchsickern“ ist auf die Endlichkeit der betrachteten Reihe zurückzuführen [49]. In der Praxis verwendet man vor einer Transformation häufig eine der in Abschnitt 4.5.2 beschriebenen Fenster-Funktionen, was zu schärfer definierten Peaks führt.

Aliasing: Die bisherigen Betrachtungen bezogen sich auf kontinuierlich definierte Funktionen. Die angewendeten Faltungen können jedoch in gleicher Weise auch für diskrete Werte benutzt werden, wie sie in Form der Audiodaten vorliegen. Allerdings ergibt sich hierbei eine Randbedingung:

Satz 3.2 (Shannon'sches Abtasttheorem) *Wenn im Zeittakt Δt abgetastet wird, ergibt sich eine Abtastfrequenz ν_{Abtast} von*

$$\nu_{Abtast} = \frac{2\pi}{\Delta t}$$

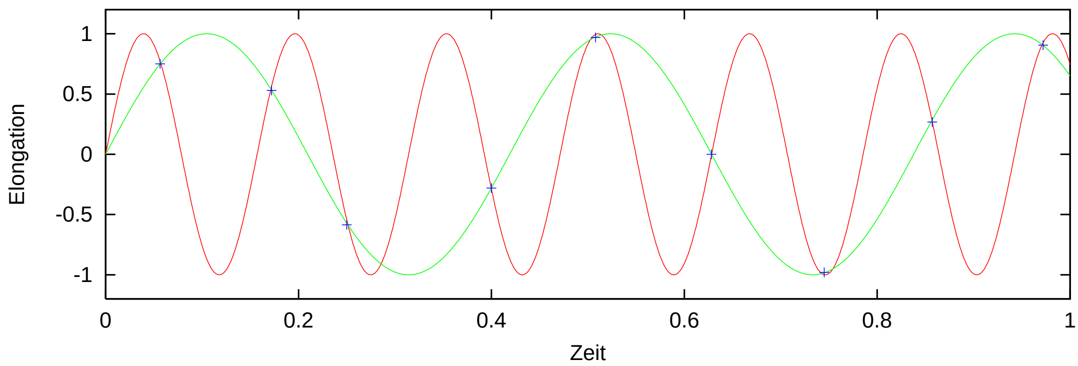


Abbildung 3.2.: Die schnelle harmonische Schwingung stellt das eigentliche Signal dar. Es wurden jedoch lediglich an den mit + markierten Stellen Werte gemessen. Verbindet man diese Punkte, so zeigt sich tatsächlich eine relativ langwellige Schwingung.

Eine vollständige Rekonstruktion der Funktion f aus den Abtastwerten ist nur dann möglich, wenn die Abtastfrequenz ν_{Abtast} mindestens doppelt so groß wie die größte in f vorkommende Frequenz ν_{max} ist, also

$$\nu_{Abtast} > 2 \cdot \nu_{max}$$

gilt.

Für die Musikdaten bedeutet dies, dass bei einer maximal hörbaren und damit auch verwendeten Frequenz von $\nu_{max} = 22.000$ Hz eine Abtastfrequenz von mindestens 44.000 Hz verwendet werden muss. Beachtet man diese Bedingung nicht, so kommt es zu dem in Abbildung 3.2 dargestellten Maskierungsproblem, dass kurzwellige Schwingungen als langwellige Schwingungen erkannt werden.

An dieser Stelle sei noch auf einige spezielle Eigenschaften der Fourier-Transformation hingewiesen, die Rückschlüsse auf die verwendeten Daten erlauben:

- Harte Kanten wie zum Beispiel die Ecken einer Rechteck- oder Sägezahnkurve oder auch scharf definierte Peaks erfordern hohe Frequenzen. Weiche Kanten oder Verläufe erfordern niedrige Frequenzen. Im Extremfall ist die konstante Funktion $f(x) = c$ zu betrachten, die als Transformierte die Amplitude c bei der Frequenz $\nu = 0$ besitzt.
- Ein Peak an der Stelle 0 erfordert *alle* Frequenzen. Man stelle sich unendlich viele Kosinus-Funktionen vor, die sich an der Stelle 0 addieren und überall sonst durch Interferenz auslöschen. Peaks an anderen Stellen entstehen durch Verschiebung. Hieraus folgt, dass scharfe Peaks im Zeit-Raum breite Kurven im Frequenz-Raum nach sich ziehen und umgekehrt.
- Aus den vorigen Punkten kann man folgern, dass lediglich Gauß-Kurven auch als Transformierte erneut eine Gauß-Kurve bilden.

Bleibt nun noch die Frage, ob man Fourier-Transformationen auf jede beliebige Funktion anwenden kann. Nach dem Satz von Dirichlet [58] ist die Entwicklung nach Sinus- und Kosinusschwingungen stets dann möglich, wenn die betrachtete Funktion stückweise stetig ist, d. h. keine Polstellen aufweist und nur endlich viele Unstetigkeitsstellen besitzt. Musikdaten erfüllen diese Bedingung natürlich, da sie keine unendlich großen Werte besitzen und naturgemäß endliche Länge besitzen. Abschließend sei erwähnt, dass sich die Anwendung der Fourier-Transformation ihrer

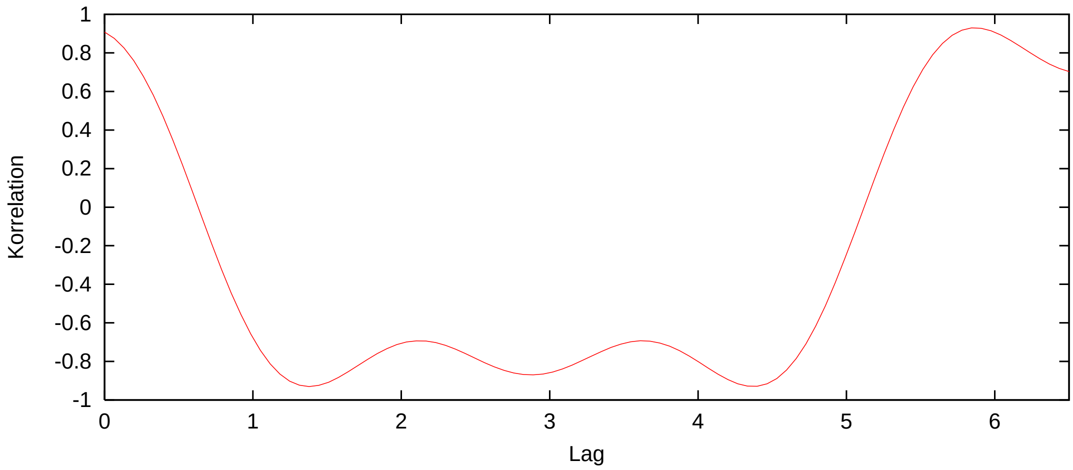


Abbildung 3.3.: Das Korrelogramm einer zusammengesetzten Welle. Werte, die dicht zusammenliegen haben einen hohen Koeffizienten. Aber auch nach einer Zeit von knapp 6 wiederholt sich die Periode.

Natur gemäß besonders für periodische Daten eignet. Da Musik gemeinhin als Überlagerung harmonischer Schwingungen aufgefasst wird, stellt sie für die Merkmalsextraktion aus Musikdaten ein wichtiges Hilfsmittel dar.

3.4. Korrelogramme

Korrelation ist ein Maß für den Zusammenhang von Änderungen zweier Variablen [56]. Bezogen auf Zeitreihen meint Korrelation häufig den Zusammenhang der Werte zu zwei Zeitpunkten (Autokorrelation). Liegt z.B. eine lineare Korrelation zwischen den Zeitpunkten i und $i+k$ vor und ist ihr Zusammenhang bekannt, so lässt sich der Wert der Reihe zum Zeitpunkt $i+k$ aus i berechnen [42].

Eine kurze Einführung für den Nachweis von Korrelation und die Berechnung von Korrelationskoeffizienten zwischen zwei Variablen liefert [50]. Der *Korrelationskoeffizient* liegt zwischen -1 und 1, wobei 0 keine Korrelation bedeutet. Ein Wert von 1 (oder nahe 1) zeigt eine hohe Korrelation zwischen den Variablen an: Sie ändern sich im gleichen Maße. Bei 0 sind die Variablen gar nicht korreliert und bei -1 steigen die Werte der einen Variable, wenn die Werte der anderen fallen und umgekehrt. Bestimmt man diesen Koeffizienten für je zwei Werte einer Reihe in variablen Schrittweiten, ergibt sich ein *Korrelogramm*. Für jede Schrittweite (*lag*) ist der Korrelationskoeffizient der Reihe aufgetragen. Abbildung 3.3 zeigt ein solches Korrelogramm.

Das Korrelogramm einer Wertereihe ist sehr eng verwandt mit dem Frequenzspektrum einer Fourier-Transformation. Schrittweiten mit hoher Korrelation korrespondieren mit den entsprechenden starken Frequenzen im Frequenzspektrum. Im Rahmen dieser Arbeit werden Korrelogramme daher nur als einfaches Verfahren eingesetzt, das Tempo eines Liedes zu bestimmen.

3.4.1. Tempo eines Liedes

Es gibt verschiedene Ansätze, das Tempo eines Liedes und den Rhythmus zu bestimmen [10, 8].

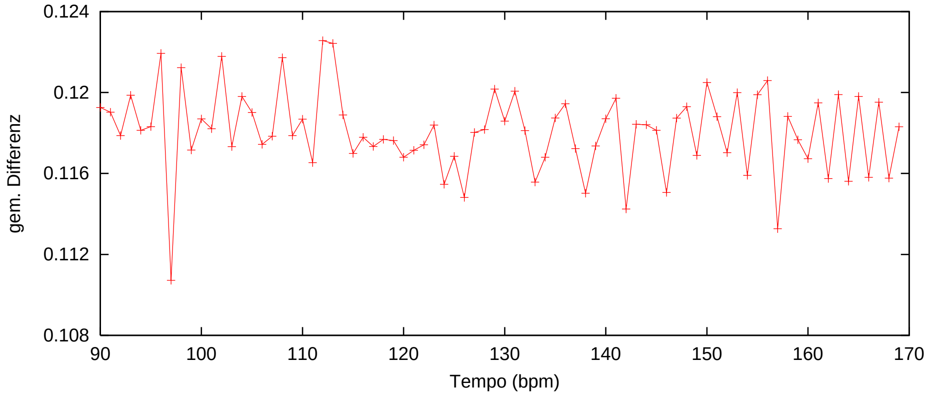


Abbildung 3.4.: Die Werte der gemittelten absoluten Differenzen für Geschwindigkeiten zwischen 90 und 170 bpm. Die minimale Differenz bei 97 bpm ist deutlich zu sehen.

Sie alle sind in der Lage, die genaue Position des ersten Schlages eines Taktes vorherzusagen. Diese Information ist nicht invariant gegenüber einer Verschiebung der Zeit und daher nicht als Basis für Merkmale geeignet. Daher wird ein einfacheres Verfahren behandelt, welches nur die Geschwindigkeit und nicht die Position des ersten Schlages bestimmt.

Das Vorgehen basiert auf einer Phasenverschiebung der Musikstücke. Sei T die erwartete Zahl von Schlägen pro Takt und SR die *sample rate* des Stückes, also die Anzahl von Werten pro Sekunde. Iterativ wird nun das gesamte Stück um T Schläge nach rechts verschoben. Dies bedeutet für eine Geschwindigkeit von X Schlägen pro Minute (*bpm*: *beats per minute*) eine Verschiebung um

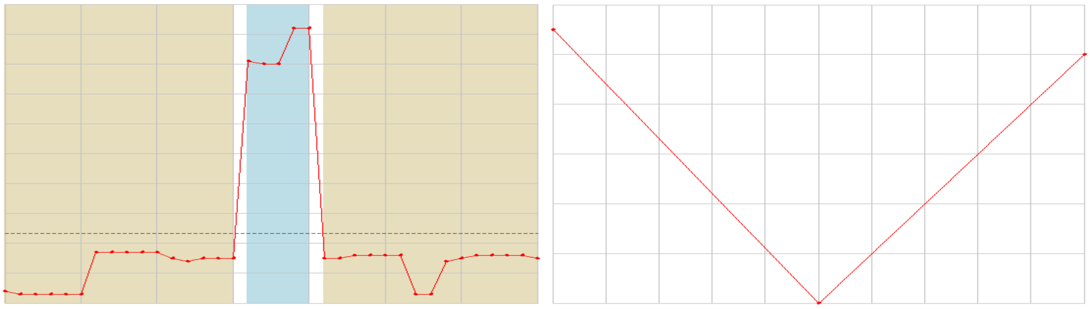
$$v = T \cdot SR \cdot \frac{60}{X}$$

Werte. Für jede Verschiebung berechnet man nun die gemittelte quadratische Differenz zum Original und erhält so ein Maß, wie gut das verschobene Stück autokorreliert. Iteriert man dieses Vorgehen für $X \in [start, ende]$, so erhält man für jede mögliche Geschwindigkeit zwischen *start* und *ende* einen Wert der Autokorrelation. Die gesuchte Geschwindigkeit entspricht der Stelle mit der minimalen Differenz, d. h. maximaler Korrelation:

$$Tempo((x_i)_{i \in \{1,\ldots,n\}}) = \min_{index} \left\{ \frac{1}{n-v} \sum_{i=1}^{n-v} (x_i - x_{i+v})^2 \,\middle|\, X \in [start, end] \right\}$$

Die Berechnung des Minimums und des Indexwertes des Minimums geschieht mit Hilfe der in Abschnitt 7.1.5 definierten Funktionale. Tests ergeben, dass dieses Verfahren für 85% aller Lieder die richtige Geschwindigkeit liefert und zudem mit Pausen innerhalb der Lieder umgehen kann.

Abbildung 3.4 zeigt die Werte der absoluten Differenzen für Geschwindigkeiten zwischen 90 und 170 bpm. Die Varianz dieser Korrelationstransformation gibt die Sicherheit an, mit der das Tempo bestimmt wurde. Sei $B = ende - start$ die Anzahl an Iterationsschritte, die das Verfahren durchführt und n die Länge der Wertereihe. Die Laufzeit ist dann gegeben durch $O(n \cdot B + B) = O((n+1)B) = O(nB)$ und damit pseudopolynomiell in n und B.



(a) Intervalle in Indexdimension (b) Intervall-Transformation

Abbildung 3.5.: Die Intervall-Transformation erzeugt für jedes Intervall in der Indexdimension ein Element der neuen Reihe. In diesem Fall entspricht der Wert jedes Elements der Länge des korrespondierenden Intervalls.

3.5. Transformationen nach Auszeichnung

Nach einer Auszeichnung der Reihe mit einer der in Kapitel 5 vorgestellten Methoden ist es möglich, eine neue Reihe auf Grundlage der ermittelten Auszeichnungen zu bilden. Die Indexdimension wird dabei nicht verändert, jedoch ermittelt man für jedes Element der neuen Reihe einen Wert in Abhängigkeit von dem Intervall, zu dem er gehört. Daher gehört diese Transformation zur Gruppe der Basistransformationen.

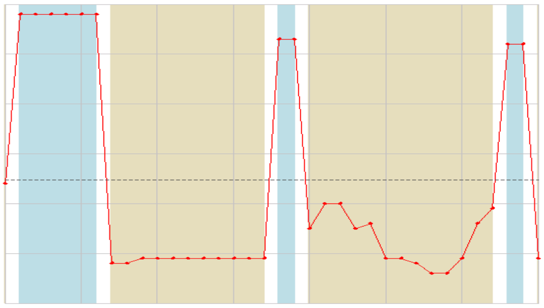
Definition 3.7 *Sei die vorliegende Reihe mit Intervallen in der Indexdimension ausgezeichnet. Die* Intervall-Transformation *konstruiert eine neue Reihe, indem für jedes Intervall in der Indexdimension in der Reihenfolge ihres Auftretens entweder*

- *der Typ*
- *die Länge oder*
- *die Dichte*

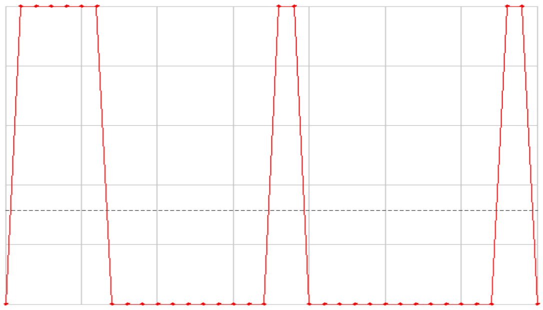
als neuer Wert benutzt wird.

Für diese Intervall-Transformation ist es nötig, die zu transformierende Reihe mit Intervallen in der Indexdimension auszuzeichnen. Dies kann entweder direkt oder nach Bildung von Intervallen in einer Wertedimension geschehen. Die neuen Werte können mit der Dichte des jeweiligen Intervalls gewichtet werden, entweder proportional oder reziprok. Auf diese Weise werden starke bzw. schwache Intervalle stärker berücksichtigt. Gewichtet man den Typ der Intervalle, werden zudem größere Schwankungen erzeugt. Abbildung 3.5 zeigt das Ergebnis der Intervall-Transformation. Da der Reihenausschnitt drei Intervalle in der Indexdimension besitzt, wird die neue Reihe drei Elemente besitzen. Jedes von diesen entspricht der Länge des korrespondierenden Intervalls.

Definition 3.8 *Sei die vorliegende Reihe mit Intervallen in der Indexdimension oder in den Wertedimensionen ausgezeichnet. Die* vollständige Intervall-Transformation *konstruiert eine neue Reihe, indem für jedes Element der Typ seines Intervalls als neuer Wert benutzt wird.*



(a) Intervalle in Indexdimension



(b) Vollständige Intervall-Transformation

Abbildung 3.6.: Die Intervall-Transformation erzeugt für jeden Wert der alten Reihe ein Element in der neuen Reihe abhängig von dem Typ des Intervalls des Wertes.

Das Ergebnis einer vollständigen Intervall-Transformation hat also genauso viele Elemente wie die zu transformierende Reihe. Auch hier ist es möglich, den Typ des Intervalls mit seiner Dichte zu gewichten. Die vollständige Intervalltransformation erlaubt zudem die Anwendung auf die Intervalle in den Wertedimensionen. Dies ist bei der normalen Intervall-Transformation, die die neuen Werte aus den Intervallen konstruiert, nicht möglich, da die Intervalle in den Wertedimensionen keine Ordnung in der Indexdimension aufweisen. Diese Ordnung wird bei der vollständigen Intervall-Transformation durch den Bezug zu den Elementen der Reihe hergestellt. Abbildung 3.6 zeigt das Ergebnis einer vollständigen Intervall-Transformation. Nach der Transformation besitzt die Reihe genauso viele Werte wie zuvor, jeder neue Wert entspricht jedoch dem Intervalltyp des alten Wertes. Dadurch wird eine intervallweise Glättung durchgeführt.

Durch die Kombination der Intervall-Transformationen mit den in Kapitel 7 beschriebenen Funktionalen wird die Extraktion zusätzlicher Merkmale aus ausgezeichneten Wertereihen ermöglicht. Auf diese Weise wird auch die Einbindung weiterer Auszeichner erlaubt, ohne dass Funktionale angepasst werden müssen. Die Intervall-Transformation arbeitet mit einer Laufzeit, die von der Anzahl der Intervalle abhängt, da für jedes Intervall ein neuer Wert gebildet wird. Im schlimmsten Fall bildet jeder Punkt ein eigenes Intervall, womit diese Transformation in Zeit $O(n)$ arbeitet. Dies gilt ohnehin für die vollständige Intervall-Transformation, wobei sichergestellt werden muss, dass die Zuordnung des Intervall zu einem Element der Reihe in $O(1)$ geschehen kann.

3.6. Nichtlineare dynamische Systeme

Dynamische Systeme ändern ihr Verhalten im Verlauf der Zeit. Lineare dynamische Systeme können durch Differentialgleichungen beschrieben werden, welche proportional von den betrachteten Größen abhängen. Nichtlineare dynamische Systeme werden durch Differentialgleichungen beschrieben, die zudem Produkte, Potenzen oder andere nichtlineare Funktionen beinhalten.

Definition 3.9 *Ein* NICHTLINEARES DYNAMISCHES SYSTEM *kann durch nichtlineare Differentialgleichungen beschrieben werden.*

Häufig ist es sinnvoll, eine Dimension des Systems anzugeben. Dies ist gleichzusetzen mit der Anzahl der Größen, die das System vollständig beschreiben.

Definition 3.10 *Die* DIMENSION *eines dynamischen Systems entspricht der Anzahl der* ZUSTANDSVARIABLEN, *d. h. der Anzahl von Größen, die zu einer vollständigen Beschreibung des Systems nötig sind.*

Beispiel 3.4 *Eine Masse, die an einem Pendel unter dem Einfluss der Gravitation schwingt, gehorcht der nichtlinearen Differentialgleichung*

$$\ddot{\theta}(t) = -mg \sin \theta(t)$$

Die Dimension dieses Systems ist 2. Die beiden Größen, die es vollständig beschreiben sind der Auslenkungswinkel θ und seine Geschwindigkeit $\dot{\theta}$. Wir werden uns auch im folgenden auf endlich-dimensionale nichtlineare Systeme beschränken.

In der Kinetik trifft man auf das Problem, das Verhalten der Bestandteile komplexer nichtlinearer Systeme aus einer einzelnen Zeitreihe zu rekonstruieren [52]. Betrachten wir einen mechanischen Apparat, bestehend aus mehreren Massen, Federn und Antriebsgeneratoren, und zeichnen lediglich die Bewegung einer einzigen Masse auf. In solch einem Fall kann die Transformation in den Zustandsraum Informationen über die Bestandteile eines solchen komplexen Systems liefern.

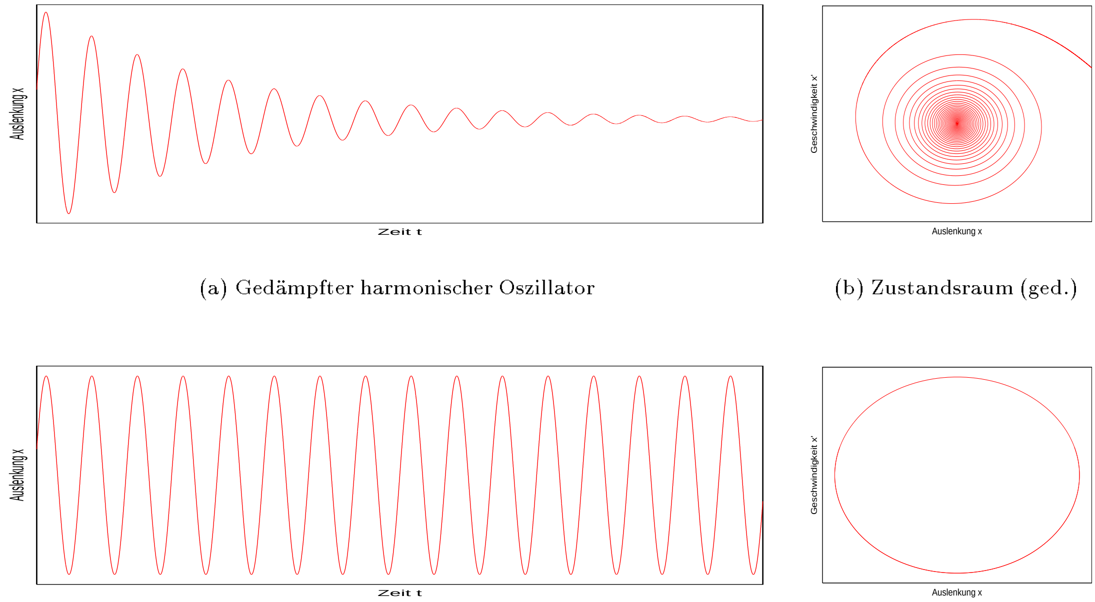
Definition 3.11 (Zustandsraum) *Der* ZUSTANDSRAUM *eines dynamischen Systems besitzt als Basisvektoren seine Zustandsvariablen. Die Elemente des Raumes sind die Werte der Zustandsvariablen zu jedem beobachteten Zeitpunkt.*

Für lineare Systeme genügen die bisherigen Transformation in den Frequenzraum und zurück in den Zeitraum. In beiden Räumen können hinreichend gute Merkmale durch Berechnung von Kenngrößen, wie Durchschnitte oder Anpassung von Funktionskurven, extrahiert werden. Für nichtlineare dynamische Systeme gilt dieses nicht mehr. Wie in Abschnitt 3.3.1 bereits angedeutet wurde, handelt es sich bei Musik um eine Superposition harmonischer Schwingungen, wodurch sich eine Transformation in den Frequenzraum anbietet. Darüberhinaus ist ein Musikstück ein komplexes System mehrerer nichtlinear arbeitender Erzeuger. Dies legt, wie auch bei anderen physikalischen Systemen, eine Transformation in den Zustandsraum nahe, um weitere Merkmale zu extrahieren.

Attraktoren in Zustandsräumen

Ist ein dynamisches System verlustbehaftet, z. B. durch Reibung, so gibt es *Attraktoren*, also Grenzwerte der Bewegung. Takens [51] hat die Zustandsraumtransformation dazu benutzt, solche Attraktoren in der chaotischen Bewegung von Flüssigkeiten zu finden. Abbildung 3.7 auf der nächsten Seite zeigt die Schwingung eines gedämpften harmonischen Oszillators, welcher sich durch Reibung langsam in der Ruhelage der Schwingung einpendelt (a). Rechts in (b) sieht man dieselbe Schwingung nach einer Transformation in ihren Zustandsraum. Die x-Achse gibt die Auslenkung x der Masse und die y-Achse ihre Geschwindigkeit $\dot{x}$ wieder. Der zeitliche Verlauf tritt nicht mehr so deutlich hervor, dafür ist die Lage des Nullpunktes gut zu erkennen. Erneut gilt, dass durch eine Basistransformation andere Informationen und Merkmale erkennbar sind. Das Zentrum der Spirale kennzeichnet den Ruhezustand der Bewegung. Das System wird sich im Nullpunkt befinden und keine Geschwindigkeit mehr besitzen.

Konservative Systeme hingegen behalten ihre Ausdehnung im Zustandsraum und besitzen keine Attraktoren. Der untere Teil aus Abbildung 3.7 zeigt eine reibungsfreie Schwingung in (c) und den zugehörigen Zustandsraum (d). Die Spiralform ist nicht mehr vorhanden, statt dessen bildet die Kurve im Zustandsraum ein Ellipsoid.



(a) Gedämpfter harmonischer Oszillator

(b) Zustandsraum (ged.)

(c) Harmonischer Oszillator

(d) Zustandsraum

Abbildung 3.7.: Die Schwingungen von harmonischen Oszillatoren und ihre Transformierten im Zustandsraum mit den Zustandsvariablen Ort x und Geschwindigkeit $\dot{x}$.

Existieren für ein dynamisches System mehrere Attraktoren, kennzeichnet jeder von ihnen ein Gebiet. Je nach Anfangsbedingungen ist ein anderer Attraktor Grenzwert. Ein Windstoß genügt, um einen Regentropfen auf einer Bergspitze verschiedene Seiten herunterlaufen zu lassen.

3.6.1. Transformation in einen unbekannten Zustandsraum: Der Phasenraum

Kann man die Zustandsvariablen messen, so erlaubt dies eine Transformation in den Zustandsraum. Oft ist es jedoch nicht einmal möglich, die Zustandsvariablen eines Systems explizit zu benennen. Ein zum Zustandsraum topologisch identischer Raum kann durch die Auswahl zeitlich verzögerter Teilstücke konstruiert werden: der Phasenraum. Gegeben ist eine univariate Zeitreihe $(x_i)_{i \in \{1,\ldots,n\}}$ der Länge n, die der Messung einer Zustandsvariablen eines k-dimensionalen nichtlinearen dynamischen Systems entspricht.

Definition 3.12 (Phasenraum) *Aus der Reihe werden Vektoren konstruiert, deren Komponenten Ausschnitte der ursprünglichen Reihe sind:*

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

Dabei stellt d die zeitliche Verzögerung (delay) und m die Dimension des entstehenden Phasenraums dar. Die Menge

$$P_{d,m} = \{\mathbf{p}_i | i = 1, \ldots, n-(m-1)d\}$$

bildet die Transformierte der Serie $(x_i)_{i \in \{1,\ldots,n\}}$ im Phasenraum.

Eine Implementierung dieser Transformation arbeitet in Laufzeit $O(n)$, da die Reihe lediglich einmal durchlaufen werden muss. Bei Musikdaten sind die Zustandsvariablen nicht bekannt und eine Transformation in den Zustandsraum ist nicht direkt möglich. Einige nützliche Informationen über die internen Vorgänge des Systems können jedoch mit einer Transformation in den Phasenraum gewonnen werden. Dieses Vorgehen ist auch als *Zustandsraumrekonstruktion* bekannt.

Interpretation des Phasenraumes

Die Messung einer einzelnen Zustandsvariablen entspricht einer Projektion des k-dimensionalen Zustandsraums auf eine einzelne Achse. Die oben beschriebene Rekonstruktion entfaltet diese Projektion auf verschiedene Achsen, abhängig von der Wahl der Verzögerung und der Anzahl der Dimensionen. Statt auf die k wahren Achsen der Zustandsvariablen projizieren wir die Reihe auf die m Achsen der Rekonstruktion. Die Beziehung zwischen dieser Rekonstruktion und dem wahren Zustandsraum stellten unabhängig voneinander [51] und [40] her: Sind genug Dimensionen m und die richtige Verzögerung gewählt, so sind die Kurve im rekonstruierten Phasenraum und die wahre, unbeobachtete Dynamik des Systems topologisch identisch. Damit sind alle Folgerungen im Phasenraum auch für das ursprüngliche System gültig. Die Aufgabe ist es, die optimalen Parameter d und m zu finden. Dabei gilt, dass m im schlimmsten Fall mindestens zweimal so groß sein muss wie die tatsächliche Zahl der Dimensionen k [5, 48]. Eine Übersicht von Heuristiken, diese Parameter abzuschätzen findet sich in [1].

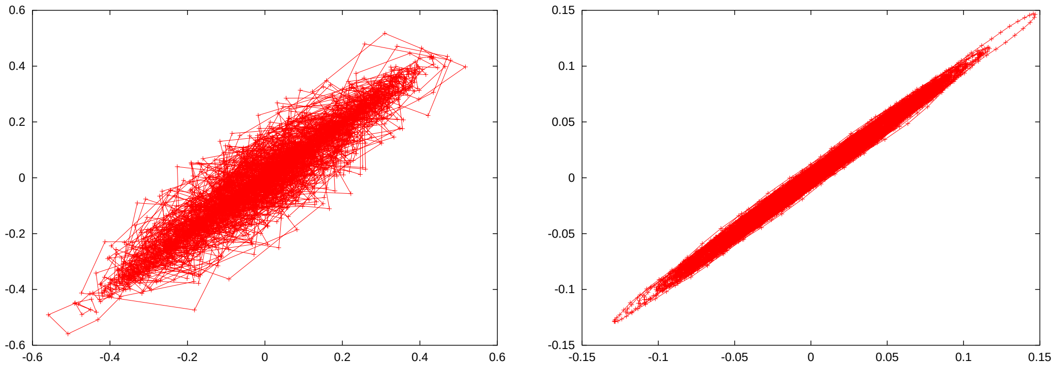
Die Kurve im Phasenraum gibt die internen Abläufe eines dynamischen Systems wieder. Dies scheint auf den ersten Blick zu verwundern. Bei genauerer Betrachtung zeigt sich jedoch, dass die Zustandsvariablen eines nichtlinearen Systems zeitlich miteinander verbunden sind. Betrachten wir zum Beispiel zwei mit einer Feder verbundene Massen und messen lediglich die Bewegung der einen Masse, so wissen wir doch, dass die zweite Masse eine ähnliche Bewegung zeitlich verzögert durchführt. Diese Vorstellung von der Projektion auf zeitlich verzögerte Achsen soll das Theorem von Takens und Packard veranschaulichen, den Beweis findet man in [40].

3.6.2. Merkmalsextraktion im Zustandsraum

Wie oben bereits erwähnt wurde, ist eine Bestimmung der Attraktoren im Zustandsraum besonders einfach. Da der Zustandsraum und der Phasenraum zwar ähnlich, aber nicht äquivalent sind, besteht eine Möglichkeit der Merkmalsextraktion darin, die Invarianten der Attraktoren zu bestimmen. Zwei bekannte Größen sind der Lyapunov Exponent und die gebrochene Dimension [1].

Definition 3.13 (Lyapunov Exponent) *Der* LYAPUNOV EXPONENT *misst die Geschwindigkeit, mit der sich verschiedene Bewegungskurven in den verschiedenen Dimensionen voneinander entfernen.*

Für die Berechnung des Lyapunov Exponenten ist die Erstellung einer Matrix nötig, die die Reaktion des Systems auf kleine Variationen enthält. Dies ist sinnvoll bei experimentellen Versuchen, nicht jedoch im Rahmen der Musikdatenanalyse. Die vorliegenden Daten sind nicht veränderlich und reagieren nicht auf Änderungen in den Anfangsbedingungen. Daher wird dieses Merkmal im Rahmen dieser Arbeit nicht untersucht.



(a) Pop

(b) Klassik

Abbildung 3.8.: Die Abbildungen (a) und (b) zeigen das Ergebnis einer Phasenraumtransformation mit $d = 1$ und $m = 2$. Bei dem klassischen Stück aus Abbildung (b) ist deutlich die höhere Zirkularität zu beobachten.

Definition 3.14 (Gebrochene Dimension) *Die* GEBROCHENE DIMENSION *misst, wieviel der k Dimensionen (Zustandsvariablen) eines Systems von einer Bewegungskurve (einem Musikstück) wirklich benutzt werden.*

Zur Berechnung der gebrochenen Dimensionen teilt man den Zustandsraum in ϵ-Bereiche auf und zählt die Anzahl der Bereiche, die nötig sind, um die Bewegungsbahn, also das Musikstück als vorliegende Zeitreihe, zu überdecken. Der Grenzwert für $\epsilon \to 0$ liefert die gewünschte Dimension.

Die Transformation in den Phasenraum hat in den letzten Jahren viele weitere Anwendungen gefunden. Außer bei der Attraktorbestimmung hat die Zustandsraumrekonstruktion auch bei einer Fülle weiterer statistischer Aufgaben wie z. B. dem Aufspüren von Ausreißern geholfen [19, 3, 38]. In den nächsten Abschnitten werden neuartige Merkmale vorgestellt, die sich im Phasenraum extrahieren lassen und die sich bei Musikdaten bewährt haben.

Winkel im Phasenraum

Abbildung 3.8 zeigt das Ergebnis der Transformation eines typischen Popmusikstückes sowie eines klassischen Stückes in den 2-dimensionalen Phasenraum. Auf den ersten Blick fällt auf, dass das klassische Stück wesentlich rundere, harmonischere Kurven im Phasenraum beschreibt, während das populäre Lied zackigere Bahnen aufweist. Dieser Unterschied soll durch zwei Merkmale gemessen werden, die den Grad der Zirkularität wiedergeben.

Zunächst soll der durchschnittliche Winkel zwischen den Teilstücken bestimmt werden. Für jeden Punkt $\mathbf{p}_i$ existieren zwei Verbindungen zu anderen Knoten:

$$\begin{aligned} \mathbf{s}_{iv} &= \mathbf{p}_{i-1} - \mathbf{p}_i \\ \mathbf{s}_{in} &= \mathbf{p}_{i+1} - \mathbf{p}_i \end{aligned}$$

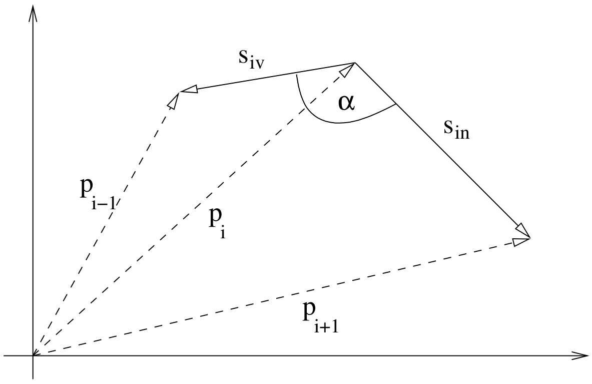


Abbildung 3.9.: $\mathbf{p}_{i-1}$, $\mathbf{p}_i$ und $\mathbf{p}_{i+1}$ sind drei aufeinanderfolgende Vektoren im Phasenraum. Sie definieren die Teilstücke $\mathbf{s}_{iv}$ und $\mathbf{s}_{in}$, welche den Winkel α einschließen.

Die Kante $\mathbf{s}_{iv}$ verläuft von $\mathbf{p}_i$ zu seinem Vorgänger und die Kante $\mathbf{s}_{in}$ zu seinem Nachfolger. Der zwischen $\mathbf{s}_{iv}$ und $\mathbf{s}_{in}$ eingeschlossene Winkel sei α. Abbildung 3.9 zeigt die verwendeten Vektoren. Mittels der geometrischen Interpretation des Skalarproduktes lässt sich α bestimmen durch

$$\alpha = \arccos \frac{\langle \mathbf{s}_{iv}, \mathbf{s}_{in} \rangle}{||\mathbf{s}_{iv}|| \cdot ||\mathbf{s}_{in}||}$$

Der Durchschnitt über die Winkel zwischen allen Teilstücken bildet somit das Merkmal:

$$AVG_{PR-\alpha} = \frac{1}{|P_{d,m}| - 2} \sum_{i=2}^{|P_{d,m}|-1} \arccos \frac{\langle \mathbf{s}_{iv}, \mathbf{s}_{in} \rangle}{||\mathbf{s}_{iv}|| \cdot ||\mathbf{s}_{in}||}$$

Es bietet sich an, die Varianz der Winkel im Phasenraum ebenfalls als Merkmal zu ermitteln. Die Umwandlung der Teilstücke in Winkel wird mittels einer weiteren Transformation, der sogenannten *Winkeltransformation*, durchgeführt. Die Berechnung des Durchschnitts entspricht dem einfachen Funktional.

Experimente zeigten, dass für für die Extraktion dieser Merkmale aus Musikdaten eine Transformation mit minimaler Verzögerung, also mit $d = 1$, in einen 2-dimensionalen Phasenraum eine gute Ausgangsbasis für die Extraktion weiterer Merkmale aus diesen Daten sind.

Durchschnittliche Länge

Bei der Betrachtung der Phasenraumdiagramme fällt weiterhin auf, dass die Längen der Teilstücke zwischen den Vektoren unterschiedlich stark schwanken. Es ist daher sinnvoll, die durchschnittliche Länge der Teilstücke

$$AVG_{PR-L} = \frac{1}{|P_{d,m}| - 1} \sum_{i=2}^{|P_{d,m}|} ||\mathbf{s}_{iv}||$$

und ihre Varianz ebenfalls als Merkmal zu benutzen. Analog zu dem Winkelmerkmal werden die Abstände der Elemente durch eine Transformation gewonnen und der Durchschnitt durch die Anwendung des Funktionals gebildet.

Die Basisvektoren im Zustandsraum entsprechen den Zustandsvariablen des dynamischen Systems. Jedes Element im Phasenraum entspricht also einer Zustandskonfiguration des Systems.

Je enger die Elemente des Phasenraums zusammen liegen, desto ähnlicher sind sich die Werte der Zustandsvariablen zu den beiden Zeitpunkten. Kleine Abstände deuten also auf kleine Änderungen der bestimmenden Größen des Systems hin.

Ähnlich verhält es sich mit den Winkeln im Phasenraum. Kleine Winkel im Phasenraum deuten an, dass eine gleichmäßige Änderung *aller* Zustandsvariablen vorliegt. Große Winkel bzw. Richtungsänderungen stehen dagegen für starke Änderungen einzelner Zustandsvariablen. Dies erklärt auch, warum sich die Winkel und Längen im Phasenraum gut für eine Trennung von populärer und klassischer Musik eignen. Letztere wird hauptsächlich mit natürlichen Instrumenten gespielt, welche ein streng harmonisches Klangbild aufweisen (vergleiche den Zustandsraum der Sinusschwingung aus Abbildung 3.7 auf Seite 28). Populäre Musik ist hingegen wesentlich perkussiver, wodurch eine schnellere Änderung der Zustandsvariablen erreicht wird. Dadurch ergibt sich das zackigere Bild des Phasenraums.

KAPITEL 4

Filter

Basistransformationen ändern die Basis des verwendeten Raums. Häufig bleiben die Elemente an der gleichen Stelle im Raum, lediglich ihre Darstellung ändert sich. Daher ist diese Form der Transformation auch meist umkehrbar.

Nun werden Transformationen beschrieben, bei denen sich Basis und Raum nicht ändern, aber das Element einen anderen Platz im betrachteten Raum einnimmt. Transformationen, die ein neues Element im gleichen Raum liefern, werden als *Filter* bezeichnet. Motiviert werden kann der Begriff durch bekannte Filter wie Hochpass oder Tiefpass. Durch herausgefilterte Frequenzen ändern sich die Elemente und einmal herausgefilterte Frequenzen können nicht mehr rekonstruiert werden: Die Filterung ist im allgemeinen nicht umkehrbar. Solange nichts anderes angegeben wird, arbeiten die hier vorgestellten Filter in $O(n)$, da ein einziger Durchlauf durch die Daten genügt.

4.1. Gleitende Durchschnitte

Ein häufiges Problem im Bereich des Data Mining sind fehlerhafte oder verrauschte Daten, häufig in Form von *Outliern*, d. h. einzelnen Werten die weit von ihrem eigentlichen Wert abweichen. Probleme dieser Art kann man mit einer Glättung der Daten vermindern.

Definition 4.1 GLÄTTUNG *einer Wertereihe bedeutet Verminderung von irregulären Schwankungen und Outliern.*

Das grundsätzliche Vorgehen ist aus der digitalen Bildverarbeitung als Weichzeichner bekannt. Hier lässt man eine $n \times m$-Matrix über das Bild laufen und ersetzt das mittlere Pixel durch den mittels der Matrix gewichteten Durchschnitt der umliegenden Pixel. Allgemein spricht man von einer *lokalen Approximation*. Im Falle von Wertereihen liegt es nahe, zur Glättung ein lokales arithmetisches Mittel zu verwenden:

$$y_i = \frac{1}{2|N|+1} \sum_{k=-N}^{+N} x_{i+k}$$

Es ergibt sich eine neue Wertereihe $(y_i)_{i \in \{1,\ldots,n\}}$ durch die Berechnung aller Werte y_i für $i \in \{1,\ldots,n\}$. Handelt es sich bei den Wertereihen um Zeitreihen, so spricht man auch davon, ein *Zeitfenster* über die Werte laufen zu lassen und in jedem dieser Fenster den Durchschnitt zu berechnen. Es liegt nahe, dass Vorgehen zu verallgemeinern und in solchen Fenstern auch gewichtete Durchschnitte zu erlauben, um z. B. jüngere Werte stärker zu gewichten.

Definition 4.2 (Gleitender Durchschnitt) *Die Berechnung einer lokalen Approximation einer Reihe* $(x_i)_{i\in\{1,\ldots,n\}}$ *durch Anwendung der Rechenvorschrift*

$$y_i = \sum_{k=-N}^{+N} a_k x_{i+k} \quad \mathit{mit} \quad \sum a_k = 1$$

in Fenstern der Größe $2N$ *heißt* GLEITENDER DURCHSCHNITT.

Wir werden die Idee der Zeitfenster später in Kapitel 6 noch aufgreifen und die Anwendung allgemeinerer Funktionen für jedes Fenster erlauben. Damit ist sowohl der hier beschriebene gleitende Durchschnitt wie auch eine erweiterte Klasse von Transformationen konstruierbar.

4.2. Rekursive Filter

In den bisher betrachteten Filtern wird ein Zeitfenster mit festen Gewichten über die vollständige Wertereihe verschoben. Oftmals ist es von Interesse, inkrementell vorzugehen und neue Beobachtungen mit einzubeziehen. Dazu wird das Ergebnis eines Filters auf den ersten $N-1$ Werten $x_1,\ldots,x_{N-1}$ mit der aktuellen Beobachtung x_N kombiniert. Man nennt solche Filter daher *rekursive Filter*.

Als einführendes Beispiel betrachten wir eine rekursive Bestimmung des arithmetischen Mittels $\bar{x}_N$ der ersten N Werte einer Wertereihe:

$$\begin{aligned} \bar{x}_N &= \frac{1}{N}\sum_{k=1}^{N} x_k \\ &= \frac{N-1}{N}\bar{x}_{N-1} + \left(1 - \frac{N-1}{N}\right) x_N \end{aligned}$$

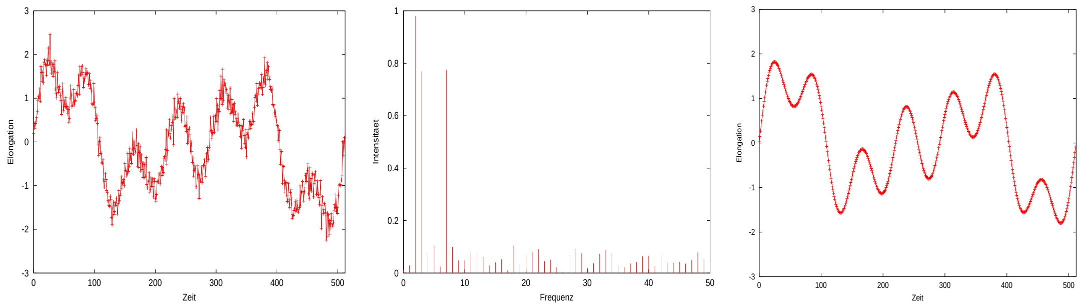
Erlaubt man allgemeinere Gewichte für das bisherige Mittel und den neuen aktuellen Wert, so erhält man eine

Definition 4.3 (Exponentielle Glättung) *Ein rekursiver Filter mit der Rechenvorschrift*

$$\bar{y}_i = \beta \cdot \bar{y}_{i-1} + (1-\beta)\cdot x_i \quad \mathit{mit} \quad 0 < \beta < 1$$

und dem Startwert $x_0 = x_1$ *heißt* EXPONENTIELLE GLÄTTUNG. β *bezeichnet man auch als Schwanzgewicht,* $1-\beta$ *als Kopfgewicht.*

Trotz des für Informatiker erschreckenden Namens arbeitet dieser Filter in $O(n)$ für eine Wertereihe mit Länge n. Der Name entsteht durch eine alternative Rechenvorschrift, bei der β im Exponenten vorkommt.



(a) Reihe (b) Spektrum (c) Glättung

Abbildung 4.1.: Die verrauschte Überlagerung von drei Sinusschwingungen mit den Frequenzen 2 Hz, 3 Hz und 7 Hz aus der linken Abbildung wird in ihr Frequenzspektrum überführt. Durch Ausblenden aller kleinen Werte entsteht die geglättete Reihe aus der rechten Abbildung nach einer Rücktransformation in den Zeit-Raum.

4.3. Ausblendung kleiner Werte

Nach der Fourier-Transformation einer Wertereihe liefert das Frequenzspektrum eine Abbildung der Frequenzen auf die Intensität, mit der die jeweilige Frequenz vorkommt. Rauschen innerhalb der Reihe ist im Frequenzspektrum an einem Grundteppich von vorkommenen Frequenzen zu erkennen, die häufig wesentlich niedrigere Intensitäten aufweisen als das eigentliche Signal. Der nun folgende Filter entfernt in einem Frequenzspektrum alle Frequenzen, deren Intensität kleiner ist als ein Anteil p der maximalen Intensität:

Definition 4.4 *Unter* Ausblendung kleiner Werte *versteht man einen Filter, der das Maximum* $\max((x_i)_{i\in\{1,\ldots,n\}})$ *einer Reihe bestimmt und für jeden Wert der Reihe*

$$y_i = \begin{cases} x_i & \textit{falls } x_i > p \cdot \max((x_i)_{i\in\{1,\ldots,n\}}) \\ 0 & \textit{sonst} \end{cases}$$

berechnet.

Nach einer solchen Ausblendung für einen Schwellenwert p kann man die Reihe wieder zurücktransformieren und erhält so ebenfalls eine geglättete Reihe. Abbildung 4.1 zeigt den Effekt einer Ausblendung und das Ergebnis der Glättung. Das zufällig hinzugefügte Rauschen betrug bis zu 20% der minimalen Amplitude. Das Verfahren arbeitet ebenso gut für größere Fehler und ist besonders dann zu empfehlen, wenn der maximale Anteil des Rauschens bekannt ist.

4.4. Differenzenfilter

Möchte man den Trend einer Wertereihe und damit die Reihe selbst mit einem Polynom beschreiben, so stellt sich zwangsläufig die Frage nach dem günstigsten Grad des Polynoms. Bei der Suche nach diesem Grad hilft der folgende Satz:

Satz 4.1 *$f(x)$ sei eine Polynom vom Grad $p > 0$. Dann ist*

$$g(x) = f(x) - f(x-1)$$

ein Polynom vom Grad höchstens $p - 1$.

Der einfache Beweis erfolgt durch Ausführung der Differenz. Die Summanden mit dem Exponenten p heben sich dabei auf [49]. Wendet man also einen Filter an, der von jedem Wert den Wert des Vorgängers abzieht, so reduziert man den Grad des Polynoms um 1. Ist die ursprüngliche Funktion ein Polynom von Grad p, so erhält man nach $p - 1$ Anwendungen dieses Filters einen (bis auf Rauschen) konstanten Wert.

Definition 4.5 (Differenzenfilter) *Der Filter, der für alle Werte einer Reihe*

$$y_i = x_i - x_{i-1}$$

berechnet, heißt DIFFERENZENFILTER.

Dieser Filter ist natürlich gut zur Entfernung eines Trends zu benutzen: Man wendet ihn solange an, bis die Werte um 0 schwanken. Darüberhinaus kennt man nun den Grad des Polynoms, welches den Trend am besten approximiert. Da Musikdaten keinen Trend besitzen, wird dieser Filter in dieser Arbeit nur zur lokalen Trendbestimmung oder nach Anwendung anderer Transformationen benutzt.

4.5. Funktionsfilter

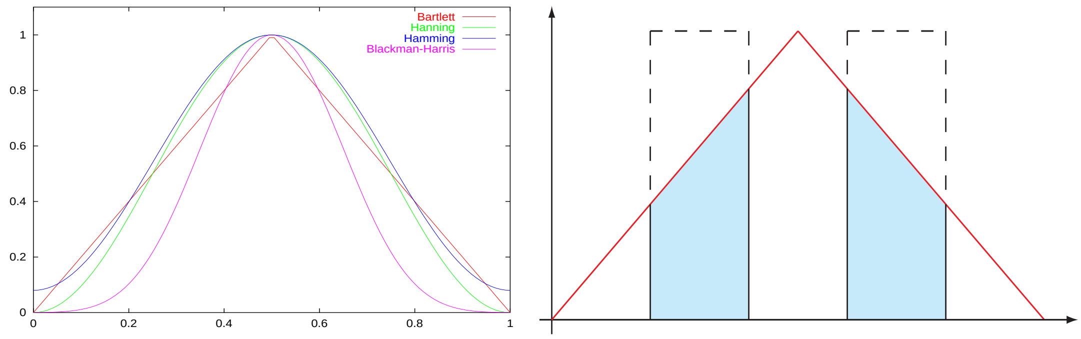
Funktionsfilter sind allgemein Filter, die eine neue Reihe durch Anwendung einer beliebigen Funktion auf die Werte der Ursprungsreihe bilden. Häufig ist die Anwendung einer Funktion auf die Reihenwerte vergleichbar mit einer Gewichtung in Abhängigkeit einer bestimmten Größe. Wir betrachten Filter, die in Abhängigkeit von der Stelle in der Reihe eine Gewichtung vornehmen und solche, die jeden Wert durch die Anwendung einer einfachen mathematischen Funktion neu skalieren. Wir definieren eine allgemeine Klasse von Filtern, welche auf jeden Wert x_i der Reihe eine Funktion f anwenden:

Definition 4.6 (Funktions-Filter) *Der* FUNKTIONS-FILTER *wendet auf jeden Wert x_i der gegebenen Reihe $(x_i)_{i \in \{1,\ldots,n\}}$ eine Funktion f an. Die Werte der entstehenden Reihe $(y_i)_{i \in \{1,\ldots,n\}}$ sind $y_i = f(x_i)$.*

4.5.1. Skalierungsfilter

Eine Skalierung in einer Achse erlaubt häufig eine bessere Darstellung von Zusammenhängen. Weit verbreitet ist die Anwendung von logarithmischen Skalen. Der Unterschied zu den Fensterfunktionen ist, dass die Funktion eines Skalierungsfilters nicht von der Stelle innerhalb der Reihe sondern nur von dem aktuellen Wert abhängt.

Die Funktion f, mit der skaliert wird, ist prinzipiell beliebig. Bewährt haben sich die Funktionen $f(x_i) = \log x_i$, $f(x_i) = e^{x_i}$ und die Skalierung um einen Faktor c, also $f(x_i) = c \cdot x_i$.



(a) Fensterfunktionen (b) Verzerrung

Abbildung 4.2.: Abbildung (a) zeigt die Fensterfunktionen Bartlett, Hanning, Hamming und Blackman-Harris. Alle sorgen für eine Abschwächung der Werte an den Rändern. Die Beschneidung einer Rechteckfunktion wie in Abbildung (b) führt allerdings zu völlig anderen Frequenzspektren.

4.5.2. Fensterfunktionen

Liegt eine Wertereihe mit n Werten vor, so sind auch Gewichtungen denkbar, deren Berechnung von der Stelle in der Wertereihe abhängt. Auf diese Weise können die Reihen an den Rändern abgeflacht werden.

Definition 4.7 (Fensterfunktion) *Gegeben sei eine Wertereihe $(x_i)_{i\in\{1,\dots,n\}}$ mit n Werten. Ein Filter*

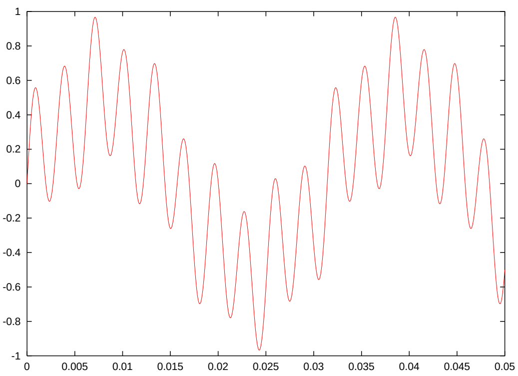
$$y_i = f_w(i) \cdot x_i$$

heißt FENSTERFUNKTION. *Die Gewichtungsfunktion f_w hängt ausschließlich von der Stelle i innerhalb der Wertereihe ab.*

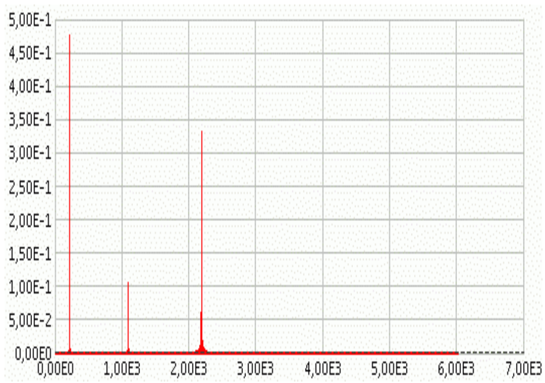
Abbildung 4.2(a) zeigt die folgenden in der Praxis erprobten Gewichtungsfunktionen, n ist die Länge der betrachteten Reihe:

$$\begin{aligned}
\textbf{Bartlett:} \quad f_{Bartlett}(i) &= 1 - \left|2 \cdot \frac{i-n/2}{n}\right| \\
\textbf{Hanning:} \quad f_{Hanning}(i) &= 0,5 - 0,5 \cdot \cos\frac{2\pi i}{n} \\
\textbf{Hamming:} \quad f_{Hamming}(i) &= 0,54 - 0,46 \cdot \cos\frac{2\pi i}{n} \\
\textbf{Blackman-Harris:} \quad f_{BH}(i) &= 0,35875 - 0,48829 \cdot \cos\frac{2\pi i}{n} \\
&\quad +0.14128 \cdot \cos\frac{4\pi i}{n} - 0.01168 \cdot \cos\frac{6\pi i}{n}
\end{aligned}$$

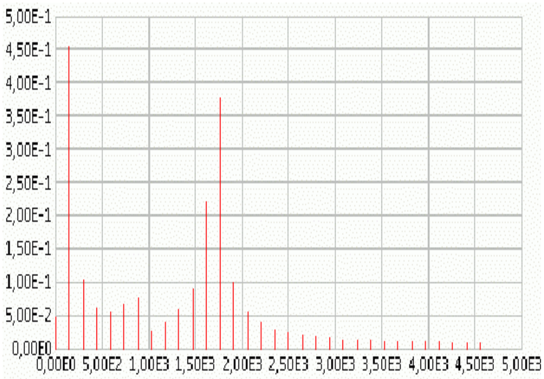
Fensterfunktionen beeinflussen nachfolgende Transformationen wesentlich. Wie bereits in Abschnitt 3.3.1 angedeutet wurde, führen scharfe Kanten einer Funktion zu verwaschenen Peaks im Frequenzspektrum, da viele und auch hohe Frequenzen nötig sind, um Kanten zu erzeugen. Der



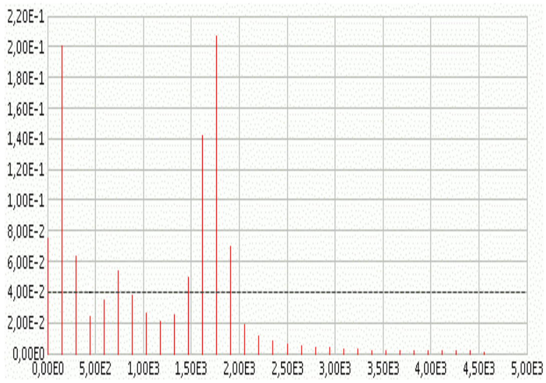
(a) Superposition



(b) Transformierte



(c) In Zeitfenster



(d) Nach Fensterfunktion

Abbildung 4.3.: Abbildung (a) zeigt die Superposition von drei Schwingungen mit den Frequenzen 200 Hz, 1000 Hz und 2000 Hz sowie den Amplituden 0.4, 0.1 und 0.4. Die Transformierte in (b) liefert trennbare Peaks, bei denen lediglich ein leichter Leakage-Effekt zu beobachten ist. Abbildung (c) zeigt die Transformation in einem schmalen Zeitfenster. Der Peak bei 1000 Hz ist kaum noch zu erkennen. In Abbildung (d) wurde vor der Transformation eine Blackman-Harris Gewichtung in dem Zeitfenster vorgenommen, wodurch der Peak bei 1000 Hz wieder wahrzunehmen ist.

Beginn und das Ende einer Wertereihe stellen solche scharfe Kanten dar und führen somit zu Frequenzen im Spektrum, die eigentlich gar nicht auftreten[1]. Daher ist es sinnvoll, die Ränder einer Wertereihe vor Anwendung einer Fourier-Transformation abzuschwächen. Abbildung 4.3 zeigt eine Superposition von harmonischen Schwingungen und ihre Transformationen in verschiedenen Zeitfenstern und den Nutzen einer im Vorfeld angewendeten Fensterfunktion.

Die Anwendung einer Fensterfunktion scheint auf den ersten Blick in jedem Fall sinnvoll zu sein. Allerdings gibt es eine Kehrseite der Medaille, die – wie so oft – im Nutzen begründet liegt: Ist der Anteil hoher Frequenzen an der vorliegenden Schwingung tatsächlich hoch, so wird das Ergebnis verzerrt. Abbildung 4.2(b) auf Seite 37 zeigt diese Problematik für eine Rechteckschwingung. Bei Musikdaten treten solche Schwingungstypen insbesondere bei elektronischer Musik auf.

Zeitliche Fensterfunktionen

Fensterfunktionen können zudem dazu benutzt werden, jüngere Werte stärker zu gewichten. Bei vielen Zeitreihen ist es wünschenswert, jüngere Werte bei der Suche nach Merkmalen stärker zu berücksichtigen. Statt einer Abschwächung an beiden Rändern des betrachteten Fensters, wendet man eine der folgenden Fensterfunktionen an:

$$\begin{aligned} \textbf{Linear:} \quad f_{linear}(i) &= \frac{i}{n} \\ \textbf{Exponentiell:} \quad f_{exp}(i) &= e^{\frac{\frac{i}{n}-1}{\gamma}} \end{aligned}$$

Der Parameter γ steuert den exponentiellen Anstieg im zweiten Fall.

Bark-Skala

Unser Gehör kann akustische Signale nur innerhalb eines bestimmten Frequenz- und Schallpegelbereichs wahrnehmen. Der für Menschen hörbare Frequenzbereich liegt etwa zwischen 20 Hz und 20 kHz. Für die Hörbarkeit ist aber außerdem noch ein gewisser Mindestschalldruck von 20 μPa erforderlich. Dieser Mindestschalldruck entspricht einem Schallpegel von 0 dB.

Dieser Abschnitt beschäftigt sich mit dem Phänomen, dass verschieden hohe Töne mit gleicher Lautstärke nicht als gleich laut empfunden werden. Im Laufe der Evolution hat sich das menschliche Ohr an seine Umwelt angepasst, wodurch zum Überleben wichtige Frequenzen leichter, also lauter, wahrgenommen werden. Dies wird zu einem speziellen Skalierungs-Filter führen, der auf ein Frequenzspektrum von Audiodaten angewendet werden sollte, um diesem Umstand Rechnung zu tragen.

Hörfläche Experimentelle Untersuchungen zum Hörverhalten von Menschen haben ergeben, dass die empfundene Lautstärke eines Tons nicht nur von der Schwingungsamplitude, sondern auch von der Frequenz des Tons abhängt. Abbildung 4.4 auf der nächsten Seite zeigt die Ergebnisse dieser Studien [62]. Auf der x-Achse sind die für den Menschen hörbaren Frequenzen von 20 Hz bis 20 kHz aufgetragen und auf der y-Achse die Schalldruckpegel in Dezibel (dB).

Zeichnet man den Schalldruckpegel, der notwendig ist um einen Ton gerade noch zu hören als Funktion der Frequenz auf, so erhält man die *Ruhehörschwelle*. Leisere Töne der gegebenen Frequenz sind nicht mehr hörbar. Erhöht man den Schalldruck, so wird man ab einem bestimmten

[1] Dieser Effekt wird bei einer Fourier-Transformation innerhalb eines Zeitfensters sogar noch verstärkt, da die Zeitfenster weniger Werte beinhalten als die gesamte Reihe; siehe Abschnitt 6.2.

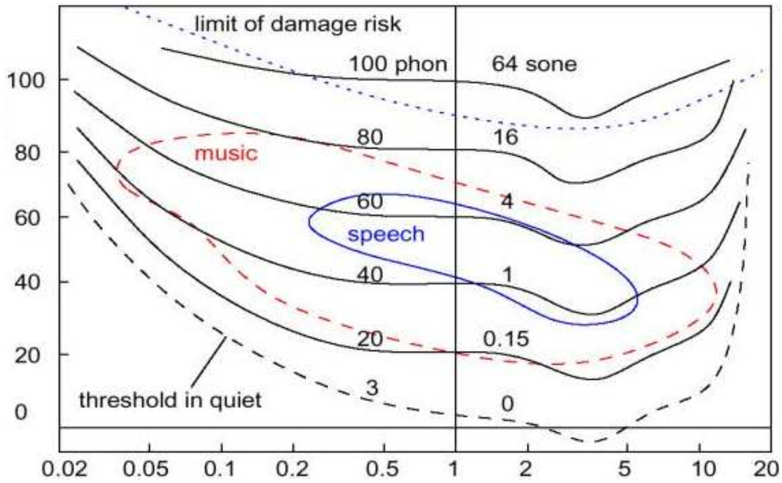


Abbildung 4.4.: Hörfläche des Menschen. Auf der x-Achse sind die hörbaren Frequenzen aufgetragen und auf der y-Achse verschiedene Lautstärken [dB]. Die durchgezogenen Linien kennzeichnen die als gleich laut empfundenen Lautstärken für verschiedene Frequenzen (aus [62]).

Schalldruckpegel beim Hören der Töne Schmerz empfinden. Trägt man diese Kurve ebenfalls in Abhängigkeit der Frequenz auf, so erhält man die *Schmerzschwelle*. Wird die Schmerzschwelle überschritten, so ist mit einer bleibenden Schädigung des Gehörs zu rechnen. Den Bereich zwischen Ruhehörschwelle und Schmerzschwelle bezeichnet man als *Hörfläche*.

Die Ruhehörschwelle ist die unterste einer Reihe von Kurven gleicher Lautstärke, also der *Isophonen*. Sie geben in Abhängigkeit der Frequenz den Schalldruckpegel an, der die jeweils gleiche Lautstärkeempfindung hervorruft wie ein Sinuston der Frequenz 1 kHz und dem zu beschreibenden Schallpegel. Aus dem Verlauf dieser Kurven geht hervor, dass das Gehör nicht für alle Frequenzen die gleiche Empfindlichkeit aufweist. Je weiter eine solche Kurve zu niedrigeren Schallpegeln hin verläuft, desto empfindlicher reagiert das Gehör auf den entsprechenden Frequenzbereich. Interessant ist, dass das menschliche Ohr im Bereich der Sprache sehr sensibel ist. Die Kurven gleicher Lautstärke besitzen im Bereich von 1 bis 5 kHz einen entsprechenden Knick. In diesem Bereich ist nur ein sehr geringer Schallpegel notwendig um eine Hörempfindung hervorzurufen.

Aus diesen Erkenntnissen lassen sich zwei Folgerungen für die Verarbeitung von Audiomerkmalen ableiten:

- Einer Extraktion von Merkmalen aus einem Frequenzspektrum oder weiteren Transformationen sollte eine Adaption des Spektrums an das menschliche Höremfinden vorangehen.

- Da die maximal wahrnehmbare Frequenz bei etwa 20 kHz liegt und nach dem Shannon'schen Abtasttheorem mindestens die doppelte Anzahl von Werten benötigt wird um Maskierungseffekten vorzubeugen, sollten Fourier-Transformationen auf 32768 Werten vorgenommen werden. Mehr Werte bringen kaum weiteren Informationsgewinn. Bei Musikdaten werden häufig jedoch weniger als die hörbaren Frequenzen benutzt, wodurch auch Fourier-Transformationen in kleineren Zeitfenstern sinnvoll sind.

Bark-Filter Die Isophonen demonstrieren eindrucksvoll, dass das Ohr für mittlere Frequenz-Bereiche besonders empfindlich ist und dass Signale im tiefen und sehr hohen Frequenzbereichen nicht gut wahrgenommen werden können. Eine Betrachtung der Frequenzspektren von Liedern zeigt dann auch, dass tiefe Frequenzen besonders laut auftreten, da diese bei der Abmischung stärker berücksichtigt werden müssen. Um das menschliche Ohr zu modellieren und der Anhebung tiefer Frequenzen entgegenzuwirken, sollte eine Gewichtung der Intensitäten in einem Frequenzspektrum in Abhängigkeit von der Frequenz erfolgen. Eine Skalierung nach psychoakustischen Maßstäben geschieht durch Umrechnung in die Einheit Barks[2]:

Definition 4.8 (Bark-Filter) *Sei* $(x_i)_{i\in\{1,\dots,n\}}$ *die gegebene Reihe. Der* BARK-FILTER *skaliert jeden Wert* x_i *der Reihe mit* $Bark_w(i)$ *und bildet eine neue Reihe* $(y_i)_{i\in\{1,\dots,n\}}$ *mit* $y_i = Bark_w(i)\cdot x_i$ *und*

$$Bark_w(i) = 13 \cdot \arctan(0,00076 \cdot index(i)) + 3,5 \cdot \arctan\left(\left(\frac{index(i)}{7500}\right)^2\right)$$

Tiefe Frequenzen in einem Spektrum werden damit reduziert und höhere entsprechend der Hörfläche angehoben, wodurch das Hörempfinden des menschlichen Ohrs modelliert wird.

ERB-Skala Neuere Erkenntnisse erlauben eine akkuratere Umrechnung der Intensitäten [35]. Durch komplexere Messverfahren wurde eine bessere Funktionsanpassung an das menschliche Ohr erreicht. Die Umrechnung eines Frequenzspektrums geschieht analog zum Bark-Filter:

Definition 4.9 (ERB-Filter) *Sei* $(x_i)_{i\in\{1,\dots,n\}}$ *die gegebene Reihe. Der* ERB-FILTER *(equivalent rectangular bandwidth) skaliert jeden Wert* x_i *der Reihe mit* $ERB_w(i)$ *und bildet eine neue Reihe* $(y_i)_{i\in\{1,\dots,n\}}$ *mit* $y_i = ERB_w(i)\cdot x_i$ *und*

$$ERB_w(i) = 21.4 \cdot \log_{10}(0.00437 \cdot index(i) + 1)$$

Dieser Filter erlaubt zudem eine einfache Rücktransformation in die normale Frequenzskala.

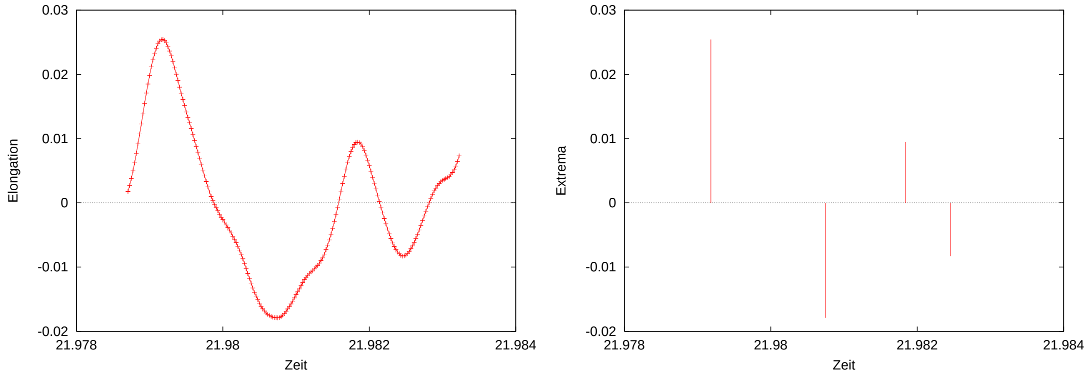
4.6. Extrema-Filter

Als nächstes betrachten wir einen Filter, der die Extrema herausarbeitet und die übrigen Werte der Reihe massiv abschwächt. Ein einfacher Algorithmus zur Findung von Extremwerten [44] liefert wiederholt angewandt besonders herausstechende Werte. Die übrigen Werte sollen nach Anwendung den Wert Null besitzen. Er behält dazu einfach alle Werte, die einen größeren Betrag aufweisen als ihre Nachbarn. Die Anzahl der Nachbarn ist mit Hilfe eines Parameters zu regeln.

Definition 4.10 (Extrema-Filter) *Der* EXTREMA-FILTER *behält lediglich die Höhe und die Position der Extrema einer Reihe, die übrigen Werte erhalten durch die Anwendung den Wert 0.*

Abbildung 4.5 auf der nächsten Seite zeigt einen Ausschnitt eines Musikstückes und die extrahierten lokalen Minima und Maxima nach Anwendung der Extrema-Transformation. Man sieht deutlich die einzelnen Samples, denen ein Extremum der Reihe entspricht. Auf diese Weise ist es möglich, die in Abschnitt 7.3.2 beschriebenen Merkmale zu extrahieren. Als Merkmale können

[2] benannt nach dem deutschen Physiker und Akustiker Heinrich Barkhausen



(a) Musikausschnitt

(b) Nach Extrema-Transformation

Abbildung 4.5.: Abbildung (a) zeigt einen Ausschnitt eines Musikstückes. In Abbildung (b) sieht man deutlich die extrahierten Extrema, die übrigen Werte sind auf Null gesetzt.

die k höchsten Extrema oder auch der durchschnittliche Abstand der Extremwerte berechnet werden. Durch die Einstellung der Anzahl der Nachbarn, welche für jedes Extremum in Betracht gezogen werden, ist eine Anpassung an verschiedene Datensätze möglich.

4.7. Nullstellen-Filter

Analog zu dem Extrema-Filter ist auch die Anwendung eines Filters möglich, welcher die Durchquerung der Null herausarbeitet. Das Merkmal Nulldurchgänge pro Zeiteinheit ist bereits erfolgreich auf Musikdaten angewendet worden.

Definition 4.11 (Nullstellen-Filter) *Sei* $(x_i)_{i \in \{1,\ldots,n\}}$ *die als Eingabe gegebene Reihe. Der* Nullstellen-Filter *bildet eine neue Reihe* $(y_i)_{i \in \{1,\ldots,n\}}$*, wobei für jeden Wert* y_i *gilt:*

$$y_i = \begin{cases} 1 & falls\ sign(x_{i-1}) \neq sign(x_i) \\ 0 & sonst \end{cases}$$

Die Filterung arbeitet also nur die Positionen der Nulldurchgänge heraus, die Merkmale müssen – genau wie beim Extrema-Filter auch – durch Anwendung des Durchschnittsfunktionals extrahiert werden (siehe Abschnitt 7.3.2).

4.8. Frequenzpässe

Frequenzpässe lassen bestimmte Frequenzbereiche durch und andere nicht. Ein *Hochpass* lässt hohe Frequenzen passieren und blockt tiefe Frequenzen ab. Ein *Tiefpass* verfährt analog mit den tiefen Frequenzen und ein *Bandpass* erlaubt nur bestimmte Frequenzbereiche.

In der Praxis verwendet man zur Entwicklung der verschiedenen Frequenzpässe häufig eine Fourier-Transformation. Ein Hochpass befreit das Spektrum von Peaks mit niedrigen Frequenzen und transformiert das geänderte Spektrum wieder in den Zeitraum. Damit sind Frequenzpässe ein weiterer Schritt zu komplexen Filtern, die sich aus der Kombination von anderen Filtern und Basistransformationen bilden lassen. Wir werden sehen, dass die Berechnung der Fourier-Transformation in Zeitfenstern deutlich schneller als auf der gesamten Reihe ist. Daher bietet sich eine Anwendung der Filterung in Zeitfenstern an, deren Ergebnisse hintereinander gehängt die neue Reihe ergibt. In Kapitel 6 behandeln wir Transformationen, die auf Basis von Fensterungen durchgeführt werden, genauer.

KAPITEL 5

Auszeichnungen in Wertereihen

Die in den letzten Kapiteln vorgestellten Transformationen ändern eine Wertereihe; entweder, indem sie die Werte ändern oder sogar die Basisvektoren des Raumes der Reihe. Diese Transformationen dienen häufig dazu, in den Reihen verborgene Informationen sichtbar zu machen und so eine Extraktion von Merkmalen zu ermöglichen. Ein alternativer Ansatz verändert nicht die Reihe, sondern sucht Bereiche innerhalb der Reihe, in denen die Elemente ähnliche Eigenschaften aufweisen. Sind solche Bereiche erst einmal gefunden, kann man die Reihe mit ihnen auszeichnen und anderen Verfahren ermöglichen, dieses neue Wissen zu benutzen. Damit andere Methoden diese Informationen nutzen können, muss die Reihe fest mit den gefundenen Bereichen verbunden werden. Analog zu Auszeichnungssprachen[1] werden die Grenzen der Bereiche gekennzeichnet und benannt:

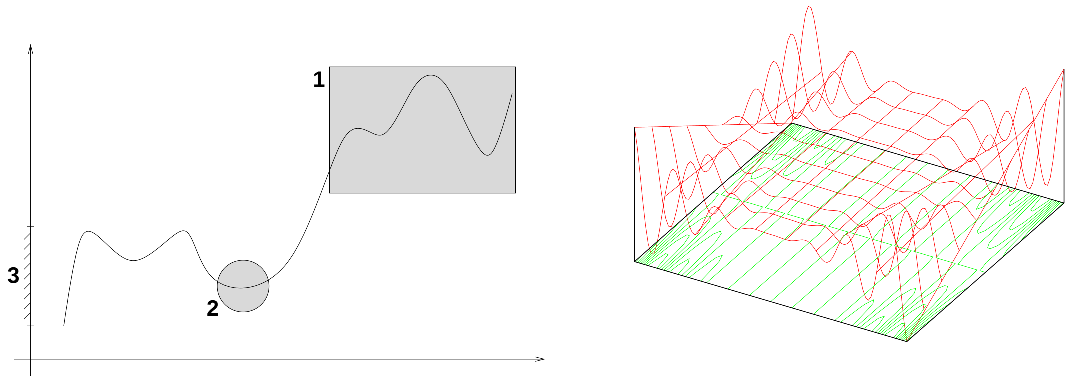
Definition 5.1 (Auszeichnung) *Eine* AUSZEICHNUNG $A : B \rightarrow E$ *weist einem Bereich* B *einer Reihe eine bestimmte Eigenschaft* E *zu.*

Der Bereich B kennzeichnet die Werte der Reihe, für die E zutrifft. Die Eigenschaft E ist hier nicht weiter eingeschränkt. Für die vorliegende Anwendung haben sich feste Konstanten praktisch bewährt, die den Typ des ausgezeichneten Bereichs spezifizieren. Bei den hier vorgestellten Verfahren wird E also stets eine Angabe des Typs sein. Der ausgezeichnete Bereich B kann sich dabei über beliebige viele Dimensionen der Reihe erstrecken. Bei einer 2-dimensionalen Reihe wird es sich entweder um eine Fläche handeln oder um ein Intervall auf einer der beiden Achsen. Für drei oder mehr Dimensionen gibt es zudem noch die Möglichkeit, Volumen innerhalb des Raums der Reihe auszuzeichnen. Abbildung 5.1 auf der nächsten Seite zeigt verschiedene Möglichkeiten der Auszeichnung.

Die Auszeichnung einer Wertereihe kann auch als Auswahl von Daten betrachtet werden. Bei den bisherigen Methoden wurde meist die Reihe als Ganzes betrachtet. Eine Datenreduktion durch Selektion bringt jedoch mehrere Vorteile mit sich. Ist eine Reihe erst einmal ausgezeichnet, so können andere Methoden jeden ausgezeichneten Bereich für sich analysieren oder Merkmale gewinnen aus der Art der Auszeichnung, der relativen Lage zueinander u. ä. Hierzu zwei Beispiele:

Beispiel 5.1 *Musikstücke folgen häufig einem bestimmten Aufbauschema, welches typisch für ein Genre ist. Dieses Schema findet sich in der Indexdimension wieder, bei populärer Musik etwa: Strophe – Refrain – Strophe – Refrain – Strophe – Übergang – Refrain. Findet man die Grenzen der Bestandteile, so ist man in der Lage, jeden Teil einzeln zu analysieren.*

[1] Beispiele für Auszeichnungssprachen sind SGML, XML und LaTeX.



(a) Auszeichnungen 2D

(b) Auszeichnungen 3D

Abbildung 5.1.: Die linke Abbildung zeigt eine Wertereihe mit einer Wertedimension. Der Kasten mit der Nummer 1 zeichnet einen Bereich aus, der durch hohe Werte in der Wertedimension gekennzeichnet ist. Denkbar sind aber auch kreisförmige Bereiche wie bei 2, der die ungefähre Lage eines Minimums wiedergibt. Schließlich kann man auch Bereiche auf den Achsen einer Dimension auszeichnen (Intervalle) wie in 3. Die rechte Abbildung zeigt eine 3-dimensionale Reihe. Der freie Bereich in der Grundebene kennzeichnet die ebene Fläche in der Mitte der Reihe.

Beispiel 5.2 *Bestimmte Frequenzen sind stärker vertreten als andere. Dies hängt zum einen von der Tonart ab, zum anderen jedoch auch von den verwendeten Instrumenten. Bestimmte Bereiche des Frequenzspektrums werden also stark vertreten sein, andere weniger. Diese Frequenzbänder sind charakteristisch für ein Lied und die instrumentelle Besetzung.*

Genau wie bei der Fensterung aus Kapitel 6 steht bei beiden Beispielen nicht nur der Aspekt der Datenreduktion im Vordergrund. Vielmehr geht es darum, zusammenhängende Bereiche ähnlicher Eigenschaften innerhalb der Reihe zu finden. Da Musikdaten 2-dimensionale Reihen sind, bietet sich die Bildung von Intervallen in der Index- und der Wertedimension an. Intervalle dieser Art entsprechen dem Aufbauschema aus Beispiel 5.1 oder den Frequenzbändern aus Beispiel 5.2. In den nächsten Abschnitten werden Methoden vorgestellt, solche Intervalle innerhalb von Reihen zu finden und die Reihe mit den gefundenen Intervallen auszuzeichnen.

5.1. Auszeichnung durch Intervalle

Intervalle kennzeichnen Bereiche innerhalb *einer* Dimension, deren Elemente ähnliche Eigenschaften aufweisen:

Definition 5.2 (Intervall) *Ein* INTERVALL $I : B \rightarrow E$ *ist eine Auszeichnung in einer Dimension. Der Bereich* $B = (d, s, e)$ *ist gegeben durch Angabe der Dimension* d*, des Startwertes* s *und des Endwertes* e *des Intervalls in der Dimension. Die Eigenschaft* $E = (t, \varrho)$ *definiert einen Intervalltyp* t *und eine Dichte* ϱ*.*

Die Dichte ist proportional zur Anzahl der Elemente in dem Intervall und gibt seine Mächtigkeit wieder. Intervalle gleichen Typs sollten ähnliche Eigenschaften aufweisen. Es werden zwei

Methoden vorgestellt, solche Intervalle in Wertereihen zu finden. Die erste Methode basiert auf dem weit verbreiteten *k-means Clustering*. Sie eignet sich bei Musikdaten für das Finden von Intervallen in Wertedimensionen, da ein Clustering in einer Dimension mit äquidistanten Werten nicht durchgeführt werden kann. Die zweite Methode bildet Intervalle inkrementell, indem es für jeden neuen Punkt entscheidet, ob er noch zum aktuellen Intervall gehören soll oder nicht. Je nach Entscheidungsfunktion sind verschiedene Intervallverfahren konstruierbar, von denen zwei vorgestellt werden.

5.1.1. Intervalle mittels Clustering

Der erste Ansatz, ähnliche Werte innerhalb einer Dimension zu finden basiert auf Clustering. Clustering-Verfahren werden häufig dazu benutzt, Anhäufungen innerhalb von Daten oder auch Ausreißer von diesen Ballungen zu finden. Ballungen innerhalb einer Dimension definieren die Grenzen der zu findenden Intervalle. Dieses Verfahren arbeitet prinzipiell sowohl in der Index- als auch in einer Wertedimension. Wenn die Werte innerhalb der betrachteten Dimension jedoch äquidistant auftreten, können keine Ballungen innerhalb dieser Dimension ermittelt werden. Musikdaten sind z. B. sowohl im Zeit- als auch im Frequenzraum äquidistant. Daher wird in dieser Arbeit ein Clustering vorwiegend innerhalb der Wertedimension angewendet.

Das anzuwendende einfache Clustering-Verfahren heißt *k-means* [16]. Zunächst spezifiziert man die Anzahl k der Cluster, die gesucht werden sollen. Diese entspricht der Anzahl der Intervalle, die als Ergebnis entstehen. Dann werden aus den Daten k Punkte zufällig gezogen, die vorerst das Zentrum der Cluster bilden. Die übrigen Punkte werden gemäß des Abstandes in der betreffenden Dimension zu dem Cluster hinzugefügt, zu dessen Zentrum sie den geringsten Abstand haben. Danach wird aus den Punkten jedes Clusters durch Bildung des Durchschnittes ein neues Zentrum berechnet. Alle Werte werden den neuen Zentren zugeordnet und diese solange erneut berechnet, bis die Zentren stabil sind und sich ihre Position nicht mehr ändert:

Algorithmus 5.1 *Sei die Reihe $(x_i)_{i \in \{1,\dots,n\}}$ der Länge n gegeben. Spezifiziere Anzahl der Intervalle k und lege die Intervalldimension fest.*

1. *Wähle k Punkte der Reihe zufällig als Zentroide aus und bestimme ihre Werte c_j innerhalb der festgelegten Dimension.*
2. *Bestimme den Abstand $d(x_i, c_j)$ aller Werte x_i der Reihe zu den Zentroiden. Füge jeden Punkt x_i dem Intervall $\arg\min_j \{d(x_i, c_j) | j \in \{1, \dots, k\}\}$ hinzu.*
3. *Berechne die Zentroide c_j für alle $j \in \{1, \dots, k\}$ neu. Hat sich der Wert eines Zentroides geändert, gehe zu 2.*

Bei diesem Ansatz entstehen zwei Typen von Intervallen: *voll* und *leer*. Volle Intervalle entsprechen den Clustern, der kleinste Wert innerhalb des Clusters entspricht dem Startwert, der größte dem Endwert, der Typ ist *voll* und die Dichte berechnet sich als Quotient

$$\varrho = \frac{\#\ \text{Werte}}{|\text{Endwert} - \text{Startwert}|}$$

Leere Intervalle entsprechen den Zwischenräumen zwischen den Clustern mit den entsprechenden Grenzen und dem Typ *leer*. Die Dichte eines leeren Clusters ist Null. Abbildung 5.2 auf der nächsten Seite zeigt die Intervalle einer Reihe. Volle Intervalle entsprechen den Frequenzbändern des

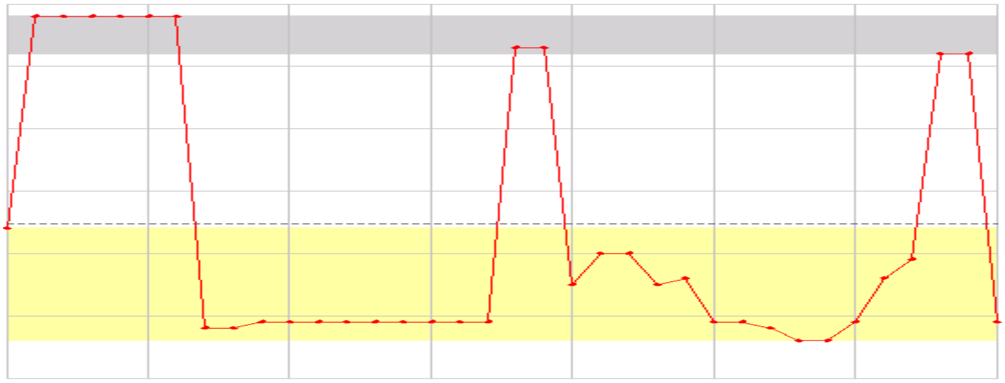


Abbildung 5.2.: Nach einer gefensterten Fourier-Transformation sind die stärksten Frequenzen in jedem Zeitfenster gegen die Zeit aufgetragen. Die Balken markieren die Cluster, die mit k-Means gefunden wurden und damit auch die Grenzen der Intervalle. Der weiße Zwischenraum bildet ein leeres Intervall.

Musikstückes, leere Intervalle entsprechen den Frequenzen, die kaum oder gar nicht im Lied vorkamen. Die Intervalle, die auf diese Weise gefunden werden entsprechen durchaus dem erwarteten Ergebnis. Allerdings leidet der k-Means Clustering Ansatz unter einigen Problemen:

- Die Konvergenz ist nur ein lokales Optimum. Bei der Wahl anderer Startzentroide kann es zu anderen und auch besseren Intervallen kommen. In der Praxis lässt man das Verfahren mehrfach mit unterschiedlichen Anfangsbedingungen starten und liefert die insgesamt beste Intervallaufteilung als Ergebnis.
- Die Anzahl der Intervalle muss vor dem Clustering festgelegt werden. Auch wenn die Festlegung für manche Anwendungen möglich ist, so erschwert es eine automatisierte Intervallfindung erheblich. Ein Ausweg besteht aus der Berechnung der *cluster gaps* und der Bestimmung der optimalen Anzahl [17].
- Pro Iterationsschritt besitzt der Algorithmus einen Aufwand von $O(nk)$, da für alle Daten der Abstand zu den Zentroiden aller Cluster berechnet und der minimale Abstand gefunden werden muss.

Im allgemeinen ist k jedoch viel kleiner als n und nach einigen Iterationen stabilisieren sich die Zentroide, so dass die Laufzeit nicht exponentiell wird. Auf äquidistanten Daten macht es zudem keinen Sinn diesen Intervallfinder in der Indexdimension anzuwenden. Um auf solchen Datensätzen Intervalle in der Indexdimension zu finden, sollte der im nächsten Abschnitt diskutierte inkrementelle Intervallfinder benutzt werden.

5.1.2. Inkrementelle Intervallauszeichnung

Das Finden von Intervallen mittels eines Clustering-Verfahrens setzt voraus, dass zum Zeitpunkt der Intervallbildung bereits alle Werte der Reihe bekannt und zugreifbar sind. Bei Anwendungen, bei denen die Reihe eher einem Strom von Daten entspricht oder Echtzeitanwendungen ist diese Bedingung jedoch nicht gegeben. Der nun vorgestellte Algorithmus zur inkrementellen Intervallfindung besitzt eine günstigere Laufzeit, benötigt keine vorherige Festlegung der Intervallzahl und hat sich als sehr flexibel erwiesen. Es basiert auf der Intervallfindung zur Verarbeitung von Robotersensordaten [36]:

Algorithmus 5.2 *Sei die Reihe* $(x_i)_{i\in\{1,\ldots,n\}}$ *der Länge n gegeben. Spezifiziere Entscheidungsfunktion* f_e *und lege die Intervalldimension fest. Initialisiere Intervallzähler* $t = 1$.

1. *Beginne ein neues Intervall* I_t *und füge den ersten Punkt hinzu.*
2. *Für die übrigen Punkte* x_i *der Reihe führe die folgenden Schritte aus:*
   a) *Wenn* $f_e(I_t, x_i) = 1$, *dann füge* x_i *dem aktuellen Intervall* I_t *hinzu.*
   b) *Sonst beende* I_t *und erhöhe t um 1. Füge* x_i *dem neuen Intervall* I_t *als ersten Wert hinzu.*

Da das Verfahren inkrementell vorgeht, kann es lediglich zur Intervallfindung in der Indexdimension eingesetzt werden. In den Wertedimensionen fehlen die zukünftigen Werte, die jederzeit zwischen bereits vorhandenden Werten eingefügt werden könnten und daher dem bereits abgeschlossenen Intervall noch hinzugefügt werden müssten. Aufgrund der Ordnung der Reihe in der Indexdimension ist das spätere Einfügen von Indexwerten jedoch ausgeschlossen. Der Kern dieses einfachen Ansatzes hängt von der Wahl der Entscheidungsfunktion f_e ab. Sie bekommt als Argumente das aktuelle Intervall I_t und den aktuellen Wert x_i und trifft die Entscheidung, den Wert zu dem aktuellen Intervall hinzuzufügen oder nicht:

Definition 5.3 *Sei I die Menge der Intervalle und W die Menge der Werte der vorliegenden Reihen. Die* ENTSCHEIDUNGSFUNKTION f_e *ist eine Abbildung* $f_e : I \times W \to \{0, 1\}$ *mit einem Boole'schen Bildbereich. Sie legt fest, ob der gegebene Wert in das gegebene Intervall eingefügt wird.*

Die Entscheidungsfunktion bestimmt, nach welchem Kriterium die Ähnlichkeit der Werte innerhalb eines Intervalls definiert ist. Auch die Laufzeit des Verfahrens wird direkt von der Wahl der Entscheidungsfunktion beeinflusst. Da das Verfahren ansonsten streng inkrementell arbeitet, resultiert eine Entscheidungsfunktion mit konstanter Laufzeit in einem Gesamtaufwand von $O(n)$. In den nächsten Abschnitten werden zwei Entscheidungsfunktionen mit konstanter Laufzeit vorgestellt, die sich bei der Behandlung von Musikdaten bewährt haben.

Starke Änderung des Wertes

Die erste Entscheidungsfunktion ist in ähnlicher Form bereits bei der Analyse der Robotersensordaten zum Einsatz gekommen. Das Ergebnis sind Intervalltypen wie „stark steigend – gleich bleibend – steigend – fallend – gleich bleibend – …“. Intervalle dieser Art bilden eine Relation der numerischen Daten mit symbolischen Begriffen. Wir beschränken uns hier auf eine Einteilung verschiedener Grade der Steigung und definieren zunächst die Typen der Intervalle und die Grenzwerte ihrer Steigungen:

stark_steigend: Steigung größer als 3

steigend: Steigung zwischen 1 und 3

gleichbleibend: Steigung zwischen -1 und 1

fallend: Steigung zwischen -1 und -3

stark_fallend: Steigung kleiner als -3

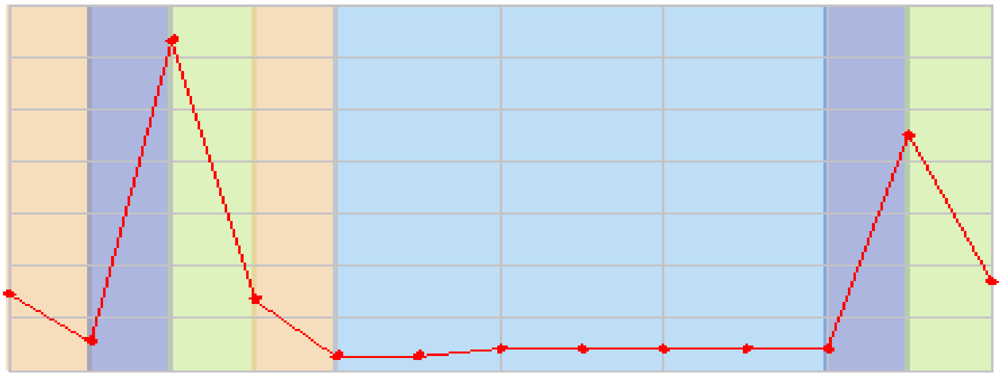


Abbildung 5.3.: Die Intervalle in der Indexdimension dieser Reihe wurden mit Hilfe einer Entscheidungsfunktion gebildet, die die Steigungen der Intervalle beachtet.

Die dazugehörige Entscheidungsfunktion bildet die Steigung m der Verbindungsgerade zwischen dem letzten Punkt des aktuellen Intervalls x_{i-1} und dem aktuell gegebenen Punkt x_i der Reihe $(x_i)_{i\in\{1,\ldots,n\}}$:

$$m = NF \cdot \frac{x_i - x_{i-1}}{index(i) - index(i-1)}$$

Diese Steigung wird normiert auf die Wertebereiche beider Dimensionen mit dem Normierungsfaktor NF:

$$NF = \frac{\max(x_i)_{i\in\{1,\ldots,n\}} - \min(x_i)_{i\in\{1,\ldots,n\}}}{index(n) - index(1)}$$

Wenn die normierte Steigung die Grenzwerte der Steigungen des aktuellen Intervalltyps überschreitet, wird das aktuelle Intervall beendet und ein neues begonnen. Natürlich liefert ein Intervallfinder mit dieser Entscheidungsfunktion eine unterschiedliche Anzahl von Intervallen für jede Instanz. Die Dichte der Intervalle wird analog zu der Dichte der Intervalle bei der k-Means Intervallfindung definiert:

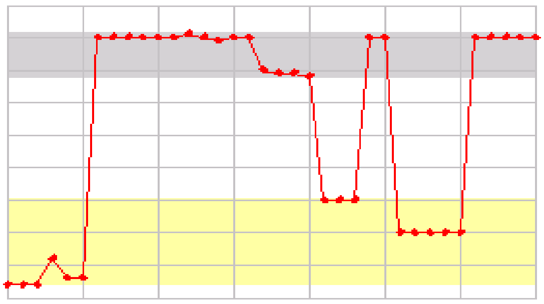
$$\varrho = \frac{\#\ \text{Werte}}{|index(\text{Endwert}) - index(\text{Startwert})|}$$

Abbildung 5.3 zeigt die gefundenen Intervalle einer Reihe. Die unterschiedlichen Farben symbolisieren die verschiedenen Typen „fallend – stark steigend – stark fallend – fallend – gleich bleibend – stark steigend – stark fallend“. Da die Berechnung der Steigung in konstanter Zeit erfolgt, besitzt eine inkrementelle Intervallfindung mit dieser Entscheidungsfunktion eine Laufzeit $O(n)$.

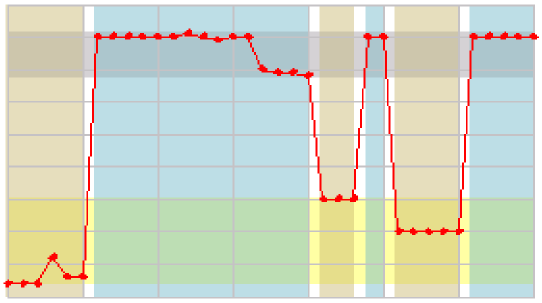
Änderung des Intervalls einer anderen Dimension

Die zweite Entscheidungsfunktion, die hier diskutiert werden soll, kann erst angewendet werden, nachdem bereits Intervalle in einer Wertedimension gefunden wurden. Sie schafft eine Abbildung der Intervalle der Wertedimensionen auf die Indexdimension, indem sie für jeden neuen Wert überprüft, ob er noch in dem gleichen Intervall der Wertedimension liegt wie die übrigen Werte des aktuellen Intervalls. Ist dies nicht der Fall, so wird ein neues Intervall in der Indexdimension begonnen. Liegt der aktuelle Wert jedoch immer noch im gleichen Werteintervall, so wird auch

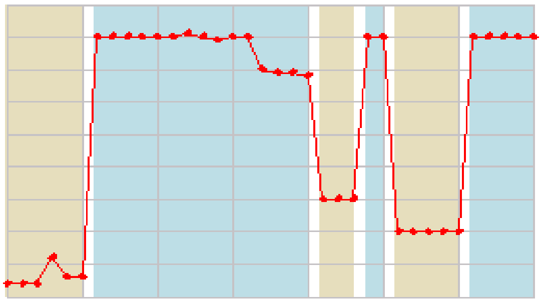
(a) Reihe



(b) Reihe mit Werteintervallen



(c) Intervallbasierte Indexintervalle



(d) Ergebnis

Abbildung 5.4.: Der Prozess der Intervallfindung mit Hilfe einer intervallbasierten Entscheidungsfunktion. Zunächst wird die Wertedimension mit Intervallen ausgezeichnet. Bei jedem Wechsel in dieser Dimension beginnt ein neues Intervall in der Indexdimension.

der jeweilige Indexwert dem aktuellen Indexintervall zugeordnet. Die Typen der Intervalle in der Indexdimension entsprechen den Typen der jeweiligen Intervalle der benutzten Wertedimension. Die Dichte ist jedoch nicht die gleiche wie die Dichte der korrespondierenden Intervalle der Wertedimension, da nicht alle Werte eines Werteintervalls innerhalb desselben Indexintervalls liegen müssen. Daher wird die Dichte der Intervalle genauso definiert wie bei der steigungsbasierten Entscheidungsfunktion.

Abbildung 5.4 zeigt den Prozess des Intervallfindens in der Indexdimension mit Hilfe der Intervalle einer Wertedimension. In Abbildung (a) ist die ursprüngliche Reihe abgebildet, in Abbildung (b) wurde die Reihe in der Wertedimension mit Intervallen ausgezeichnet. Dazu wurde der in Abschnitt 5.1.1 vorgestellte k-Means Intervallfinder benutzt. Abbildung (c) zeigt das Ergebnis einer inkrementellen Intervallfindung mit Hilfe der hier vorgestellten intervallbasierten Entscheidungsfunktion. Immer wenn das Intervall in der Wertedimension wechselt, wird ein neues Intervall in der Indexdimension begonnen. Die letzte Abbildung zeigt die Reihe mit den Indexintervallen als Auszeichnung.

KAPITEL 6

Allgemeine Fensterung

Zur Nachbildung aller bekannten Methoden der Wertereihenanalyse wird ein Konzept benötigt, mit dem Wertereihen abschnittsweise behandelt werden können. Es wird gezeigt, dass hierdurch zudem effizientere Berechnungen ermöglicht werden. Darüberhinaus erlaubt eine verallgemeinerte Sichtweise dieser Fensterung die Einführung neuartiger Extraktionsmethoden.

In Abschnitt 4.1 wurde die Berechnung eines gleitenden Durchschnitts durch die Verschiebung eines Zeitfensters über die Wertereihe und Bildung des Durchschnitts in jedem Fenster definiert. Es drängt sich die Frage auf, ob nicht auch andere Funktionen, außer der Berechnung von Durchschnitten, sinnvolle Ergebnisse liefern. In einem ersten Schritt werden also beliebige Funktionale innerhalb von Fenstern zugelassen:

Definition 6.1 (Fensterung) *Sei eine Wertereihe $(x_i)_{i\in\{1,\ldots,n\}}$ der Länge n gegeben. Eine Transformation heißt* FENSTERUNG, *wenn sie ein Fenster der Breite w mit einer Schrittweite s über die Reihe verschiebt und in jedem Fenster den Wert eines Funktionals F berechnet:*

$$y_j = F((x_i)_{i\in\{j\cdot s,\ldots,j\cdot s+w\}})$$

Die y_j bilden erneut eine Zeitreihe $(y_j)_{j\in\{1,\ldots,(n-w)/s+1\}}$, wobei der Wert eines jeden y_j das Resultat von F in dem betreffenden Fenster ist. F heißt FENSTERFUNKTIONAL.

Ein Funktional ist eine Abbildung eines Funktionenraums auf einen Raum von Zahlen, vergleiche Kapitel 7. Ist F definiert als Durchschnittsfunktion, so stellt eine Fensterung mit dem Fensterfunktional F den Filter zur Bildung eines gleitenden Durchschnittes dar. Berechnet F hingegen das Maximum des Fensters, so erhalten wir als Ergebnis eine Reihe, die für den Verlauf der Zeit die größten Werte enthält.

Beispiel 6.1 *Die Zeitreihe $(x_i)_{i\in\{1,\ldots,n\}}$ beinhaltet die täglichen Verkaufszahlen eines Produktes für den Zeitraum eines Jahres. Interessiert man sich für die kleinsten Verkaufszahlen pro Monat, so genügt die Verwendung des Minimums als Fensterfunktional F, einer Schrittweite $s = 30$ und einer Breite $w = 30$. Das Ergebnis ist erneut eine Zeitreihe, welche für alle zwölf Monate die jeweiligen minimalen Verkaufszahlen enthält. Dies gilt zumindest in Kalendersystemen mit 30 Tagen pro Monat.*

Desweiteren ist es denkbar, vor Berechnung des Funktionals beliebige Transformationen zu erlauben.

Definition 6.2 (Allgemeine Fensterung) *Eine* ALLGEMEINE FENSTERUNG *ist eine Fensterung, die zusätzlich zu dem Fensterfunktional noch beliebig viele Transformationen enthalten darf.*

Abbildung 2.5 auf Seite 14 gibt das Konzept der allgemeinen Fensterung wieder. Prinzipiell wäre die Forderung nach einem Fensterfunktional nicht unbedingt nötig. Es hat sich in der Praxis jedoch herausgestellt, dass bei Verwendung großer Datensätzen, wie z. B. Musikdaten, der Verzicht auf Funktionale die Datenmenge derart erweitern kann, dass diese selbst mit modernen Rechnern nicht mehr bearbeitet werden können. Eine Fensterung mit Schrittweite 1 und lediglich einer Transformation ohne Funktional bläht die Daten um eine weitere Dimension auf. Es gibt zwei Möglichkeiten, wie man einer Inflation der Datenmenge bei der Anwendung einer allgemeinen Fensterung entgegenwirken kann. Durch die Erweiterung auf allgemeine Fensterungen und die gleichzeitige Forderung eines Funktionals wird die Erhöhung der Dimensionszahl vermieden. Welcher Art die Transformationen vor Anwendung des Fensterfunktionals waren – es sind durchaus auch weitere Fensterungen möglich – unterliegt keiner Beschränkung. Damit wird nicht nur die Klasse der gleitenden Durchschnitte und der gefensterten Fourier-Transformation erzeugt, sondern auch eine Vielzahl Faltungen von Filtern und Basistransformationen.

Alternativ kann die Forderung nach einem Funktional entfallen und durch eine Forderung nach einer Filterung ersetzt werden. Um die Datenmenge nicht zu vergrößern, dürfen sich die Fenster zudem nicht überlappen. Auf diese Weise kann eine stückweise Filterung konstruiert werden. In diesem Fall dürfen allerdings nur Filterungen und keine Basistransformationen oder Auszeichner innerhalb jedes Fensters angewendet werden. Die entstehende Reihe hat dann die gleiche Länge und die gleiche Dimensionszahl wie die ursprüngliche Reihe. Die stückweise Filterung wird in Abschnitt 6.4 untersucht.

Zum Schluss soll noch die Anzahl der Fenster bestimmt werden, die bei einer Fensterung mit Schrittweite s und Fensterbreite w entstehen. Diese Anzahl spielt bei der Bestimmung der Laufzeiten eine große Rolle.

Lemma 6.1 *Eine allgemeine Fensterung mit Schrittweite s und Fensterbreite w bildet auf einer Reihe $(x_i)_{i\in\{1,\ldots,n\}}$ der Länge n insgesamt*

$$\frac{n-w}{s}+1$$

Fenster der Breite w, in denen jeweils die inneren Methoden ausgeführt werden.

Beweis: Das letzte Fenster der Länge w, welches noch vollständig innerhalb der Reihe gebildet werden kann, darf spätestens beim Wert $n-w$ starten. Bis dahin sind bereits $\frac{n-w}{s}$ Fenster gebildet worden, alle s Werte eines. Die zusätzliche 1 entspricht dem letzten Fenster.

6.1. Laufzeiten der allgemeinen Fensterung

Für alle Laufzeitbetrachtungen liegt eine Wertereihe $(x_i)_{i\in\{1,\ldots,n\}}$ der Länge n zugrunde. Sie wird durch die Fensterung in Teilstücke der Fensterbreite w aufgeteilt. Bei einer vorgegebenen Schrittweite s beginnt alle s Werte ein neues Fenster. Ist die Schrittweite kleiner als die Fenstergröße, so überlappen sich die Werte innerhalb der Fenster und einige Werte werden bei der Anwendung einer Methode mehrfach benutzt. Die Befürchtung ist, dass mehrfach benutzte Werte zu einer Vergrößerung der Laufzeit im Vergleich zur Ausführung der Methode auf der gesamten Wertereihe führen.

Definition 6.3 (Überlappungsgrad) *Der* ÜBERLAPPUNGSGRAD *einer allgemeinen Fensterung mit Schrittweite s und Fensterbreite w ist definiert als*

$$g = \frac{w}{s}$$

Fensterungen mit einem Überlappungsgrad $g \leq 1$ heißen ÜBERLAPPUNGSFREI.

Alle Fensterungen mit einem Überlappungsgrad $g < 1$ sind zwar möglich, aber unrealistisch. Es werden nicht alle Werte der Reihe für die Fensterung verwendet und somit Informationen verschenkt. Eine überlappungsfreie Fensterung mit $g < 1$ entspricht eher einer Auswahl von Daten (*sampling*) und weniger dem Vorgang einer Fensterung. Wir betrachten nur Fälle, bei denen alle Daten berücksichtigt werden und überlassen die gezielte Auswahl von Daten den darauf spezialisierten Algorithmen.

Bei den übrigen Fensterungen ist die Schrittweite kleiner als die Fensterbreite, es gilt also $s < w$. In diesem Fall ist der Überlappungsgrad größer als 1 und die Werte der Reihe werden mehrfach bearbeitet. Wenn der Überlappungsgrad $g = 1$ ist, werden alle Werte genau einmal betrachtet. Die Schrittweite s ist dann genauso groß wie die Fensterbreite w. Dies ist die günstigste Fensterung, die wir betrachten wollen. Häufig möchte man aber, dass sich die Fenster leicht überlappen. Damit wird versucht, ein glatteres Ergebnis der Fensterung zu erreichen. In der Praxis wird daher häufig ein Überlappungsgrad von 2 gewählt. Abbildung 6.1 zeigt diese Überlappung.

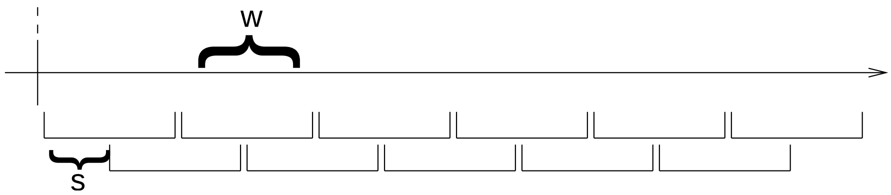


Abbildung 6.1.: Die Abbildung zeigt die Überlappung einer realistischen Fensterung. Die Fensterbreite ist doppelt so groß wie die Schrittweite, wodurch sich ein Überlappungsgrad von 2 ergibt.

Um auch den schlimmsten Fall abschätzen zu können, betrachten wir noch eine Fensterung mit einer Schrittweite $s = 1$. Ziel ist eine möglichst hohe Zahl von Werten, die mehrfach behandelt werden.

Lemma 6.2 *Sei $(x_i)_{i \in \{1,\ldots,n\}}$ eine Wertereihe der Länge n. Eine allgemeine Fensterung mit Schrittweite 1 und Fensterbreite $w = \frac{n}{2}$ besitzt die größte Anzahl mehrfach behandelter Werte.*

Beweis: Eine Fensterung mit Schrittweite 1 kann k Fenster mit einer Fensterbreite von $n - k$ Werten bilden. Die Gesamtzahl der dabei behandelten Werte ist das Produkt $z(k) = k \cdot (n - k)$. Gesucht ist das k, welches dieses Produkt maximiert. Wir bilden die erste Ableitung nach k:

$$z'(k) = n - 2k$$

Zum Finden des Extremums müssen die Nullstellen der Ableitung bestimmt werden:

$$\begin{aligned} & n - 2k && = 0 \\ \Leftrightarrow \quad & 2k && = n \\ \Leftrightarrow \quad & k && = \frac{n}{2} \end{aligned}$$

Die zweite Ableitung ist -2, es handelt sich also wirklich um ein lokales Maximum.

Bei einer gegebenen Schrittweite von $s = 1$ liefert also die Wahl $w = \frac{n}{2}$ die meisten mehrfach zu behandelnden Werte, der Überlappungsgrad ist ebenfalls $g = \frac{n}{2}$. Für innere Methoden mit Laufzeit $O(n)$ ist die Wahl dieses Überlappungsgrades in der Tat der schlimmste Fall, es gibt keine Wahl von Parametern, die die Laufzeit stärker vergrößert. Bei anderen inneren Methoden stellt $g = \frac{n}{2}$ eine untere Schranke für den schlimmsten Überlappungsgrad g dar. Bei Methoden mit hohem Aufwand kann es teurer sein, weniger – aber dafür größere – Fenster zu wählen.

Wir werden die Laufzeitbetrachtungen für allgemeine Fensterungen für verschiedene innere Methoden durchführen. Neben der Berechnung in Abhängigkeit vom Überlappungsgrad g wird exemplarisch der Aufwand einer solchen Fensterung für die Grade 1, 2 und $\frac{n}{2}$ berechnet. Das Vorgehen zur Bestimmung der Laufzeit der Fensterungen wird immer gleich sein. Zunächst wird die Anzahl der Fenster berechnet und diese dann mit dem Aufwand für jedes Fenster multipliziert. Die Anzahl der Fenster für eine Fensterung mit Schrittweite s und Fensterbreite w ist nach Lemma 6.1 gleich $\frac{n-w}{s} + 1$. Mit der Einführung des Überlappungsgrades kann hierfür auch $\frac{n}{s} - g + 1$ geschrieben werden. Wir setzen voraus, dass jede Transformation nur einmal verwendet wird. Werden die Transformationen aus einer Menge mit Zurücklegen gewählt, so kann jede Methode mehrfach verwendet werden und die Anzahl der Transformationen innerhalb einer Fensterung beliebig groß werden. Dieses gilt dann natürlich auch für die Laufzeit. In den nächsten Absätzen werden die Laufzeiten für allgemeine Fensterungen, bei denen die inneren Methoden den Aufwand $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^p)$ und $O(a^n)$ haben, berechnet.

Satz 6.3 *Eine allgemeine Fensterung enthält Methoden mit Laufzeit $O(n)$ und besitzt Überlappungsgrad g. Es gilt: Die Fensterung arbeitet mit Aufwand $O(gn - gw + w)$.*

Beweis: Die Anzahl der Methoden innerhalb jeder Fensterung ist endlich und konstant. Der Aufwand für die Berechnung eines Fensters beträgt also $O(w)$. Für die Ausführung aller k Fenster der allgemeinen Fensterung gilt nach Lemma 6.1 eine Laufzeit von

$$\begin{aligned} k \cdot w &= \left(\frac{n}{s} - g + 1\right) \cdot w \\ &= \frac{n}{s} w - gw + w \end{aligned}$$

und mit $g = \frac{w}{s}$ schließlich

$$= gn - gw + w$$

Für einen Überlappungsgrad $g = 1$ fallen die Terme gw und w weg und es bleibt ein Aufwand von n. Da jeder Wert genau einmal betrachtet wird und die Methoden linearen Aufwand besitzen, ergibt sich auch in der Summe eine lineare Laufzeit. Für eine realistische Fensterung mit Überlappungsgrad 2 ergibt sich ein Aufwand von $2n - w$. Obwohl noch linear, so ist diese Fensterung für alle $w < n$ langsamer als die Anwendung der Methoden auf die komplette Reihe. Bei $g = \frac{n}{2}$ beträgt die Laufzeit $\frac{n^2}{2} - \frac{wn}{2} + w$. Durch den quadratischen Term ist die Laufzeit bei der Fensterung mit maximaler Überlappung um eine Potenz schlechter als die lineare Laufzeit der inneren Methoden.

Satz 6.4 *Eine allgemeine Fensterung enthält Methoden mit Laufzeit $O(n \log n)$ und besitzt Überlappungsgrad g. Es gilt: Die Fensterung arbeitet mit Aufwand $O(gn \log w - gw \log w + w \log w)$.*

Beweis: Die Anzahl der Methoden innerhalb jeder Fensterung ist endlich und konstant. Der Aufwand für die Berechnung eines Fensters beträgt also $O(w \log w)$. Für die Ausführung aller k Fenster der allgemeinen Fensterung gilt nach Lemma 6.1 eine Laufzeit von

$$\begin{aligned} k \cdot w \log w &= \left(\frac{n}{s} - g + 1\right) \cdot w \log w \\ &= \frac{n}{s} w \log w - wg \log w + w \log w \\ &= gn \log w - gw \log w + w \log w \end{aligned}$$

Erneut betrachten wir exemplarisch die drei Überlappungsgrade. Bei $g = 1$ heben sich die beiden letzten Terme auf und es bleibt ein Aufwand von $n \log w$. Es gilt $n \log w < n \log n$, da log streng monoton wachsend und stets $w < n$ ist. Bei der realistischen Fensterung mit $g = 2$ ergibt sich eine Laufzeit von $(2n - w) \log w$. Dies ist erst für $w = n$ genauso groß wie $n \log n$. Sobald also eine Fensterung eine Methode mit Laufzeit $O(n \log n)$ enthält, verringert selbst eine realistische Fensterung mit $g = 2$ den Gesamtaufwand! Nun berechnen wir noch den Aufwand einer teuren Fensterung mit Überlappungsgrad $\frac{n}{2}$ und Schrittweite $s = 1$:

$$\begin{aligned} ng \log w - wg \log w + w \log w &= \frac{n^2}{2} \log \frac{n}{2} - \frac{n^2}{4} \log \frac{n}{2} + \frac{n}{2} \log \frac{n}{2} \\ &= \frac{n^2}{4} \log \frac{n}{2} + \frac{n}{2} \log \frac{n}{2} \\ &= \frac{n^2}{4} \log n - \frac{n^2}{2} + \frac{n}{2} \log n - n \end{aligned}$$

Diese Laufzeit ist durch den dominierenden quadratischen Term $n^2 \log n$ in jedem Fall schlechter als die Anwendung der Methoden auf die gesamte Reihe.

Satz 6.5 *Eine allgemeine Fensterung enthält Methoden mit Laufzeit $O(n^2)$ und besitzt Überlappungsgrad g. Es gilt: Die Fensterung arbeitet mit Aufwand $O(gnw - gw^2 + w^2)$.*

Beweis: Die Anzahl der Methoden innerhalb jeder Fensterung ist endlich und konstant. Der Aufwand für die Berechnung eines Fensters beträgt also $O(w^2)$. Für die Ausführung aller k Fenster der allgemeinen Fensterung gilt nach Lemma 6.1 eine Laufzeit von

$$\begin{aligned} k \cdot w^2 &= \left(\frac{n}{s} - g + 1\right) \cdot w^2 \\ &= \frac{n}{s} w^2 - gw^2 + w^2 \\ &= gnw - gw^2 + w^2 \end{aligned}$$

Auch hier die üblichen Beispiele: Bei $g = 1$ ergibt sich die Laufzeit $n \cdot w$, was für alle $w < n$ kleiner ist als n^2. Für die realistische Fensterung mit $g = 2$ errechnet sich der Aufwand zu $2nw - w^2$, was n^2 erst für $w = n$ erreicht. Wie bereits bei einem Aufwand von $O(n \log n)$ der inneren Methoden, lohnt sich eine realistische Fensterung mit Überlappungsgrad 2 erneut. Im schlimmsten Fall ist der Überlappungsgrad $\frac{n}{2}$. Dann ist der Aufwand

$$\begin{aligned} gnw - gw^2 + w^2 &= n \cdot \left(\frac{n}{2}\right)^2 - \left(\frac{n}{2}\right)^3 + \left(\frac{n}{2}\right)^2 \\ &= \frac{n^3}{4} - \frac{n^3}{8} + \frac{n^2}{4} \\ &= \frac{n^3}{8} + \frac{n^2}{4} \end{aligned}$$

Die Laufzeit ist um eine Potenz schlechter als bei der Anwendung der Methoden auf die gesamte Reihe.

Satz 6.6 *Eine allgemeine Fensterung enthält Methoden mit Laufzeit $O(n^p)$ und besitzt Überlappungsgrad g. Es gilt: Die Fensterung arbeitet mit Aufwand $O(gnw^{p-1} - gw^p + w^p)$.*

Beweis: Die Anzahl der Methoden innerhalb jeder Fensterung ist endlich und konstant. Der Aufwand für die Berechnung eines Fensters beträgt also $O(w^p)$. Für die Ausführung aller k Fenster der allgemeinen Fensterung gilt nach Lemma 6.1 eine Laufzeit von

$$\begin{aligned} k \cdot w^p &= \left(\frac{n}{s} - g + 1\right) \cdot w^p \\ &= \frac{n}{s} w^p - gw^p + w^p \\ &= gnw^{p-1} - gw^p + w^p \end{aligned}$$

Bei $g = 1$ heben sich die Terme mit w^p auf und es ergibt sich die Laufzeit $n \cdot w^{p-1}$, was für alle $w < n$ kleiner ist als n^p. Für die realistische Fensterung errechnet sich der Aufwand zu $2nw^{p-1} - w^p$, was n^p erst für $w = n$ erreicht. Der Aufwand für Überlappungsgrad $\frac{n}{2}$ ist:

$$\begin{aligned} gnw^{p-1} - gw^p + w^p &= n \cdot \frac{n}{2} \cdot \left(\frac{n}{2}\right)^{p-1} - \left(\frac{n}{2}\right)^p + \left(\frac{n}{2}\right)^p \\ &= n \cdot \left(\frac{n}{2}\right)^p - \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \\ &= n \cdot \frac{n^p}{2^p} - \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \\ &= \frac{n^{p+1}}{2^p} - \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \\ &= \frac{2n^{p+1}}{2^{p+1}} - \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \\ &= \frac{n^{p+1}}{2^{p+1}} + \left(\frac{n}{2}\right)^p \\ &= \left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p \end{aligned}$$

Die Laufzeit für diesen Überlappungsgrad ist stets um eine Potenz schlechter als bei der Anwendung der Methoden auf der gesamten Reihe.

Satz 6.7 *Eine allgemeine Fensterung enthält Methoden mit Laufzeit $O(a^n)$ und besitzt Überlappungsgrad g. Es gilt: Die Fensterung arbeitet mit Aufwand $O(\frac{gn}{w} a^w - ga^w + a^w)$.*

Beweis: Die Anzahl der Methoden innerhalb jeder Fensterung ist endlich und konstant. Der Aufwand für die Berechnung eines Fensters beträgt also $O(a^w)$. Für die Ausführung aller k Fenster der allgemeinen Fensterung gilt nach Lemma 6.1 eine Laufzeit von

$$\begin{aligned} k \cdot a^w &= \left(\frac{n}{s} - g + 1\right) \cdot a^w \\ &= \frac{n}{s} a^w - g a^w + a^w \\ &= \frac{gn}{w} a^w - g a^w + a^w \end{aligned}$$

Für Fensterungen mit $g = 1$ heben sich die letzten beiden Terme a^w auf und es ergibt sich die Laufzeit $\frac{n}{w} a^w$. Der exponentielle Faktor überwiegt vor dem linearen, so dass die Laufzeit für alle $w < n$ kleiner ist als a^n. Für die realistische Fensterung beträgt der Aufwand $\left(\frac{2n}{w} - 1\right) a^w$, was a^n erneut erst für $w = n$ erreicht. Der Aufwand bei Überlappungsgrad $\frac{n}{2}$ ist:

$$\begin{aligned} \frac{gn}{w} a^w - g a^w + a^w &= n \cdot \frac{n}{2} \cdot \frac{2}{n} a^{\frac{n}{2}} - \frac{n}{2} a^{\frac{n}{2}} + a^{\frac{n}{2}} \\ &= \left(n - \frac{n}{2} + 1\right) a^{\frac{n}{2}} \\ &= \left(\frac{n}{2} + 1\right) a^{\frac{n}{2}} \end{aligned}$$

Es handelt sich zwar immer noch um eine exponentielle Laufzeit, jedoch hat der Exponent sich halbiert. Für innere Methoden mit exponentiellen Laufzeiten ist also eine Verringerung des Aufwands unabhängig von dem Überlappungsgrad zu erwarten. Von den in dieser Arbeit vorgestellten Transformationen und Funktionalen besitzt jedoch keine Methode eine exponentielle Laufzeit. Es ist aber nicht auszuschließen, dass zukünftige Methoden eine exponentielle Laufzeit aufweisen und entsprechend stark von einer Fensterung profitieren.

Tabelle 6.1 auf der nächsten Seite stellt die Ergebnisse zusammen. Sie gibt die Laufzeiten der allgemeinen Fensterung für verschiedene Überlappungsgrade g und verschiedene innere Methoden wieder. Die Fensterung mit dem geringsten Aufwand besitzt Überlappungsgrad 1. Außer bei linearen inneren Methoden besitzt diese Fensterung für alle Laufzeiten eine kleinere Gesamtlaufzeit. Für die Praxis relevant sind jedoch Fensterungen mit einer Überlappung wie $g = 2$. Es ließ sich zeigen, dass unabhängig von der Laufzeit der inneren Methoden eine solche Fensterung weniger oder genauso viel Aufwand verursacht wie die Anwendung der Methoden auf die gesamten Daten. Der Überlappungsgrad $g = \frac{n}{2}$ liefert eine Abschätzung für den schlimmsten Fall. Hier können die Laufzeiten um eine Klasse in der O-Notation steigen. Der positive Effekt der Fensterung auf die Laufzeit tritt für diesen Überlappungsgrad erst bei exponentiellen Methoden auf. In dieser Arbeit kommen jedoch schlimmstenfalls Methoden mit quadratischer Laufzeit vor. Mit dem in der Praxis bewährten Überlappungsgrad von 2 spart man also stets Laufzeit.

6.2. Beispiel: Stärkste Frequenz in Zeitfenstern

Die Diskussion der allgemeinen Fensterung wird mit einem Beispiel beendet, welches sich auf Musikdaten bewährt hat. Da Musikdaten meist eine Superposition harmonischer Schwingungen sind, bietet sich die Anwendung einer Fourier-Transformation geradezu an. Wendet man diese auf das gesamte Musikstück an, erhält man lediglich die Information, wie intensiv jede Frequenz

Methode	**Laufzeit**	$\mathbf{g = 1}$	$\mathbf{g = 2}$	$\mathbf{(s = 1, w = \frac{n}{2}) \to g = \frac{n}{2}}$
$O(n)$	$gn - gw + w$	n	$2n$	$\frac{n^2}{2}$
$O(n \log n)$	$gn \log w - gw \log w + w \log w$	$n \log w$	$(2n - w) \log w$	$\frac{n^2}{4} \log n - \frac{n^2}{2} + \frac{n}{2} \log n - n$
$O(n^2)$	$gnw - gw^2 + w^2$	nw	$2nw - w^2$	$\frac{n^3}{8} + \frac{n^2}{4}$
$O(n^p)$	$gnw^{p-1} - gw^p + w^p$	nw^{p-1}	$2nw^{p-1} - w^p$	$\left(\frac{n}{2}\right)^{p+1} + \left(\frac{n}{2}\right)^p$
$O(a^n)$	$\frac{gn}{w}a^w - ga^w + a^w$	$\frac{n}{w}a^w$	$(\frac{2n}{w} - 1)a^w$	$(\frac{n}{2} + 1)a^{\frac{n}{2}}$

Tabelle 6.1.: Die Tabelle fasst die Laufzeiten der allgemeinen Fensterung für verschiedene Überlappungsgrade g und verschiedene innere Methoden zusammen. Überlappungsgrad $g = 1$ liegt für Fensterungen mit gleicher Schrittweite s und w vor. Eine realistische Fensterung besitzt $g = 2$. Unabhängig von der Laufzeit der inneren Methoden ist die Laufzeit dieser Fensterung niedriger als der Aufwand der Methoden auf den gesamten Daten. Die letzte Spalte entsteht durch Schrittweite $s = 1$ und $w = \frac{n}{2}$ und damit $g = \frac{n}{2}$. Hier tritt der Effekt der Fensterung erst bei den „teureren“ Methoden auf.

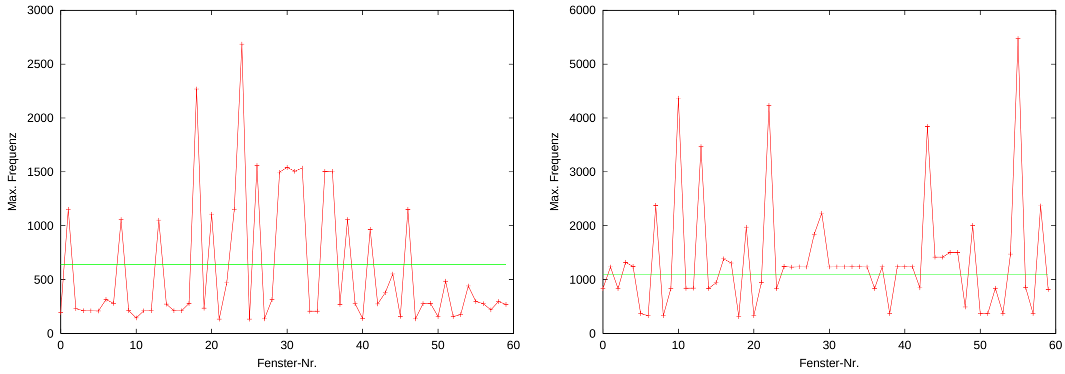
insgesamt auftritt, nicht aber, an welchen Stellen des Liedes dieses geschieht. Daher ist die Berechnung der höchsten Peaks der Fourier-Transformierten naturgemäß unabhängig von zeitlichen Verschiebungen.

Mit diesem Vorgehen verschenkt man jedoch die Information bezüglich der zeitlichen Ordnung. Daher wird ein Zeitfenster mit Schrittweite s und Breite w über die gesamte Zeitreihe verschoben und in jedem Fenster

$$y_j = \max_{index}(FT(\{x_i\}_{i \in \{j \cdot s, \ldots, j \cdot s + w\}})))$$

berechnet. Diese y_j bilden erneut eine Zeitreihe $(y_j)_{j \in \{1, \ldots, n/s-1\}}$, wobei der Wert eines jeden y_j die in dem betreffenden Zeitfenster stärkste Frequenz bezeichnet. Abbildung 6.2 auf der nächsten Seite zeigt das Ergebnis dieser Transformation für ein Klassik- und für ein Popstück. Man sieht deutlich die größere Varianz von Frequenzen bei dem klassischen Stück. Als Merkmale bildet man folglich den Durchschnitt von $(y_j)_{j \in \{1, \ldots, n/s-1\}}$ sowie die Varianz der Reihe.

Die gefensterte Fourier-Transformation zeigt das Zusammenspiel einer Transformation und eines Funktionals innerhalb einer Fensterung. In jedem Zeitfenster wird zunächst eine Fourier-Transformation durchgeführt und dann mit Hilfe des $\max_{index}$ Funktionals aus Abschnitt 7.1.5 das Ergebnis für jedes Fenster bestimmt. Die Werte für alle Fenster bilden dann die in Abbildung 6.2 dargestellten Reihen, aus denen weitere Merkmale extrahiert werden oder die mit Auszeichnungen versehen werden können. Die gefensterte Fourier-Transformation ist also eine Rücktransformation in den Zeit-Raum. Die Varianz gibt zwar die zeitliche Veränderlichkeit an, die beiden Merkmale Durchschnitt und Varianz sind jedoch unabhängig vom genauen Zeitpunkt der auftretenden Ereignisse. Da die Laufzeit einer Fast Fourier Transformation durch $O(n \log n)$ gegeben ist und die Berechnung des Maximums in $O(n)$ möglich ist, ist die Anwendung einer gefensterten Fourier-Transformation mir einem Überlappungsgrad von z. B. 2 zudem noch schneller als die Anwendung der Transformation auf die gesamten Daten.



(a) Pop (b) Klassik

Abbildung 6.2.: Die Abbildungen (a) und (b) zeigen das Ergebnis einer gefensterten Fourier-Transformation. Zu jedem Fenster wird die stärkste Frequenz bestimmt und aufgetragen, die horizontale Linie gibt die durchschnittliche Frequenz an.

6.3. Fensterbildung in Abhängigkeit einer Auszeichnung

Eine Fensterung wird durchgeführt, um Eigenschaften einer Reihe lokal zu berechnen und diese Eigenschaft einer bestimmten Stelle zuzuordnen. Die allgemeine Fensterung bedient sich dabei zweier Parameter: Der Schrittweite und der Fensterbreite. Je kleiner die Schritte, desto mehr Bereiche der Reihe betrachtet man, je größer die Breite der Fenster, desto unschärfer wird der analysierte Bereich. Dabei besteht die Hoffnung, dass sich die Wahl der Parameter im Durchschnitt einer Fensterung als gut erweist und die Häufigkeit und die Menge der betrachteten Daten dazu geeignet sind, die Reihe mittels einer Fensterung zu transformieren. Dabei kann man beliebig falsch liegen! Betrachten wir z. B. eine angeregte Schwingung, deren Anregungsfrequenz sich ungleichmäßig über die Zeit ändert. In drei unterschiedlich langen Perdioden sollen sich sowohl die Amplitude als auch die Frequenz der erregenden Schwingung ändern. Eine Fensterung mit festen Schrittweiten und Fensterbreiten wird niemals korrekte Werte für Amplitude und Frequenz bestimmen können und diese Werte den unterschiedlich langen Bereichen als Merkmal zuweisen können.

Ein einfacher Ausweg besteht darin, flexible Fensterbreiten und Schrittweiten zu erlauben. Jedoch stellt sich immer noch die Frage, wie die Parameter für eine vorliegende Reihe gewählt werden sollen. In Abschnitt 2.3 wurde die Methodengruppe der Auszeichner vorgestellt, die in Kapitel 5 ausführlich behandelt wurden. Mit ihrer Hilfe können wir zusammenhängende Bereiche in der Indexdimension (Intervalle) kennzeichnen und Methoden auf diesen Bereichen anwenden:

Definition 6.4 (Intervall-Fensterung) *Sei $(x_i)_{i\in\{1,\ldots,n\}}$ eine in der Indexdimension mit Intervallen $I_1,\ldots,I_m$ ausgezeichnete Wertereihe der Länge n. Eine* Intervall-Fensterung *ist eine allgemeine Fensterung, deren Schrittweite s und Fensterbreite w nach jedem Fenster angepasst werden durch die Länge des aktuellen und des nächsten Intervalls.*

Es wird also je ein Fenster gebildet für alle m Intervalle. Diese Fenster überlappen sich nicht, d. h. der Überlappungsgrad ist $g = 1$. Das Vorgehen ist analog zu dem einer allgemeinen Fensterung ohne Überlappung. Es gelten also alle Aussagen für Fensterungen mit Überlappungsgrad $g = 1$ in gleicher Weise für die Intervall-Fensterung.

6.4. Stückweise Filterung

An dieser Stelle wird eine weitere Untergruppe der allgemeinen Fensterung behandelt. Die Forderung nach einem Funktional ist dadurch begründet worden, dass die Anwendung einer Transformation eine neue Reihe für jedes Fenster ergibt. Bei einem hohem Überlappungsgrad bilden alle Reihen zusammen eine 3-dimensionale Reihe. Dieser Effekt der Datenvermehrung tritt immer bei einem Überlappungsgrad von $g > 1$ auf und einer Fensterung ohne Funktional. Die entstehenden Reihen sind bei großen Datensätzen wie den Musikdaten jedoch nicht mehr im Hauptspeicher zu bearbeiten. Um trotzdem in den Genuss der Laufzeitverringerung durch die Anwendung einer Fensterung ohne Funktional zu kommen, lassen wir die Forderung nach einem Funktional fallen und verlangen stattdessen einen Überlappungsgrad $g = 1$. Die entstehenden Reihen in jedem Fenster werden genauso aneinandergehängt wie die Ergebnisse der Funktionale aus den Fenstern der allgemeinen Fensterung. Auf dieser Weise wird das „Aufblähen“ um eine Dimension vermieden.

Definition 6.5 (Stückweise Filterung) *Eine allgemeine Fensterung ohne Fensterfunktional mit einem Überlappungsgrad $g = 1$ heißt* STÜCKWEISE FILTERUNG.

Da das Aneinanderhängen von Reihen nach einer Basistransformation nur selten zu sinnvollen Ergebnissen bei einer überlappungsfreien Fensterung führt, betrachten wir als innere Methoden in jedem Fenster hauptsächlich Filterungen; daher auch der Name. Für diese Fensterungen gelten die gleichen Laufzeitbetrachtungen wie für allgemeine Fensterungen mit Überlappungsgrad $g = 1$.

6.4.1. Beispiel: Frequenzfilter

Im letzten Kapitel wurde bereits die Aufgabe eines Frequenzfilters beschrieben. Die Konstruktion eines solchen Filters mit Hilfe der stückweisen Filterung ist nun denkbar einfach: In jedem Fenster wird zunächst eine Fourier-Transformation durchgeführt. Durch die Anwendung eines Funktionsfilters werden die unerwünschten Peaks reduziert und danach mit einer weiteren Fourier-Transformation die Reihe wieder in den Zeit-Raum überführt. Durch die Anwendung der stückweisen Filterung in Zeitfenstern mit einer Größe von mindestens 32.768 Werten wird eine schnellere Frequenzfilterung erreicht als durch die Anwendung der Methoden auf der gesamten Reihe.

Interessant ist, dass die Fourier-Transformation selbst kein Filter ist, aber durch die Kombination der drei Methoden in jedem Fenster ein Filter durchgeführt wird.

6.5. Basistransformation oder Filter?

Es wurde gezeigt, dass durch die allgemeine Fensterung die Anwendung eines gleitenden Durchschnittes simuliert werden kann. In gleicher Weise sind auch weitere Filter denkbar, die mit Hilfe von Fensterung erzeugt werden können. Die stückweise Filterung ist selbst natürlich auch ein Filter. Im Beispiel der gefensterten Fourier-Transformation änderte sich jedoch die Basis der

Wertedimension. Vor der Anwendung waren die Elongationen eines Musikstückes aufgetragen, danach die zu jedem Zeitpunkt stärkste Frequenz.

Eine allgemeine Fensterung umfasst also Transformationen aus *beiden* Gruppen, sowohl aus den Basistransformationen als auch aus der Gruppe der Filter. Es hängt von dem inneren Funktional und den inneren Transformationen ab, ob die Ausgabedimensionen die gleichen sind wie vor Anwendung der Fensterung. Innerhalb jeden Fensters kann mehrfach die Basis gewechselt werden, es kommt lediglich auf die Basis nach der letzten Transformation und auf das abschließende Funktional an.

Die allgemeine Fensterung erlaubt also die Konstruktion vieler unterschiedlicher Transformationen, die sowohl der Gruppe der Basistransformationen als auch den Filterungen zugeordnet werden können. Für effiziente Methoden ist auch eine allgemeine Fensterung effizient. Dadurch und durch die Beschränkung auf bestimmte innere Methoden in Abhängigkeit vom Überlappungsgrad ist eine allgemeine Fensterung gut als Transformation auf großen Datenmengen geeignet.

KAPITEL 7

Funktionale

Nachdem in den letzten Kapiteln die Transformationen von Wertereihen genau untersucht wurden, soll in diesem Kapitel die Grundlage für die zweite große Gruppe der Methoden gebildet werden: die *Funktionale*.

Wertereihen können als Funktion in einem m-dimensionalen Raum beschrieben werden. Für Musikdaten ist dies der $\mathbb{R}^2$, wobei auf der x-Achse der zeitliche Verlauf des Liedes und auf der y-Achse die Elongation zu jedem Zeitpunkt aufgetragen wird. Aus dem Bereich der deskriptiven Statistik ist das Vorgehen bekannt, aus Reihen Kenngrößen wie Durchschnitte oder Standardabweichungen zu berechnen. Wir benötigen also eine Abbildung von Reihen auf reelle Zahlen:

Definition 7.1 *Sei F ein Funktionenraum. Eine Abbildung $f : F \to \mathbb{R}$ heißt* FUNKTIONAL.

In Kapitel 2 wurde für maschinelle Lernverfahren eine Eingabe als Menge von Merkmalsvektoren definiert. Die Merkmale weisen untereinander keine Ordnung mehr auf. Daher kann im Bereich des maschinellen Lernens das Ergebnis eines Funktionals direkt als Merkmal benutzt werden.

Statt $\mathbb{R}$ sind als Bildraum auch beliebige andere Zahlenräume möglich. Aus Kapitel 3 sind bereits einige Funktionale bekannt. Die Norm, aber auch ein Skalarprodukt in einem Funktionenraum sind Funktionale. Im Fall der Musikdaten ist z. B. die Berechnung der mittleren Lautstärke als Funktional aufzufassen, da aus einer Funktion im $\mathbb{R}^2$ (dem Musikstück) eine Zahl – nämlich die Lautstärke, eine Zahl aus $\mathbb{R}$ – berechnet wird. Das Ergebnis des Funktionals entspricht also dem Merkmal „Lautstärke“ eines Musikstückes. Lautet die Klassifikationsaufgabe, laute von leisen Liedern zu unterscheiden, so sollte dieses Merkmal bereits gute Dienste leisten. In anderen Anwendungen, also nicht auf Musikdaten bezogen, besitzt das Funktional eine ganz andere Interpretation (z. B. durchschnittliche Verkaufsmenge) und manchmal gelingt es nur mit viel Aufwand, eine einleuchtende Interpretation des Ergebnisses eines Funktionals zu finden. Dies gilt um so mehr, wenn die Berechnung des Funktionals erst nach mehreren Fensterungen und Basistransformationen stattfindet.

7.1. Statistische Kenngrößen

In den folgenden Abschnitten sollen einige Funktionale definiert werden, die sich im Rahmen der Zeitreihenanalyse und auch für die Anwendung auf Musikdaten bewährt haben. Da viele von ihnen in der Statistik als Kenngrößen zur Charakterisierung von Reihen eingesetzt werden, erfolgt eine Zusammenfassung unter dem Namen *statistische Kenngrößen*.

7.1.1. Durchschnitt

Eine der wesentlichen Größen bei der Analyse von Wertereihen ist der durchschnittliche Wert der gesamten Reihe. Auch wenn die meisten der hier vorgestellten Durchschnittsberechnungen für Musikdaten einen Wert nahe bei Null ergeben, ist ihre Bedeutung im Rahmen der Analyse von Wertereihen so groß, dass die bekanntesten Durchschnitte und ihre Rechenvorschriften hier vorgestellt werden. Bei Musikdaten findet vor allem der absolute Durchschnitt zur Berechnung der mittleren Lautstärke und die Berechnung von Durchschnitt und Varianz nach unterschiedlichen Transformationen Anwendung.

Definition 7.2 (Arithmetisches Mittel) *Das* ARITHMETISCHE MITTEL *einer gegebenen Reihe* $(x_i)_{i\in\{1,\dots,n\}}$ *der Länge* n *berechnet sich wie folgt:*

$$d_{arithm}((x_i)_{i\in\{1,\dots,n\}}) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Der Arithmetische Mittelwert gibt die Lage des Schwerpunkts der Verteilung an. Er errechnet sich aus der Summe der Werte, dividiert durch die Anzahl der Werte. Durch die Anwendung einer Filtertransformation vor der Berechnung des arithmetisches Mittels wird ein gewichtetes Mittel berechnet. Bei Zeitreihen ist es so einfach möglich, jüngere Werte gegenüber ältere Werte stärker zu gewichten. Auch für die anderen hier vorgestellten Mittel wird eine Gewichtung auf diese Weise durchgeführt.

Definition 7.3 (Quadratisches Mittel) *Das* QUADRATISCHE MITTEL *einer gegebenen Reihe* $(x_i)_{i\in\{1,\dots,n\}}$ *der Länge* n *berechnet sich wie folgt:*

$$d_{quad}((x_i)_{i\in\{1,\dots,n\}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$$

Das Quadratische Mittel gewichtet die Werte stärker, die weit von Null entfernt sind. Eine Anwendung ist die Berechnung des Effektivwertes abgetasteter Signale, z. B. von Spannungen oder Strömen. Bei Audiodaten ist das quadratische Mittel auch bekannt als *root mean square amplitude* und ein Maß für die Lautstärke.

Definition 7.4 (Geometrisches Mittel) *Das* GEOMETRISCHE MITTEL *einer gegebenen Reihe* $(x_i)_{i\in\{1,\dots,n\}}$ *der Länge* n *berechnet sich wie folgt:*

$$d_{geo}((x_i)_{i\in\{1,\dots,n\}}) = \left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}}$$

Das Geometrische Mittel wird verwendet, wenn Durchschnitte von Verhältniszahlen berechnet werden sollen. Dies kommt besonders bei Wachstums- und Zerfallsprozessen vor. Dazu werden alle x_i miteinander multipliziert und daraus die n-te Wurzel gezogen. Das Geometrische Mittel sollte nur verwendet werden, wenn alle Werte der Reihe größer als Null sind.

Da die Berechnung von Produkten und das Ziehen der n-ten Wurzel eine große Rechenkomplexität bedeutet, berechnet eine günstigere Implementierung des geometrischen Mittels folgendes:

$$d_{geo}((x_i)_{i\in\{1,\dots,n\}}) = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \log x_i\right)$$

Eine weitere Anwendung findet das geometrische Mittel bei der Berechnung typischer Audiomerkmale (siehe Abschnitt 7.6).

Definition 7.5 (Harmonisches Mittel) *Das* HARMONISCHE MITTEL *einer gegebenen Reihe* $(x_i)_{i\in\{1,\ldots,n\}}$ *der Länge n berechnet sich wie folgt:*

$$d_{harm}((x_i)_{i\in\{1,\ldots,n\}}) = \frac{n}{\sum\limits_{i=1}^{n} \frac{1}{x_i}}$$

Das Harmonische Mittel entspricht in der Berechnung dem Arithmetischen Mittel. Die zu mittelnden Größen sind jedoch die Kehrwerte der Reihenelemente. Eine bekannte Anwendung ist die Berechnung der Durchschnittsgeschwindigkeit, wenn jeweils gleiche Strecken mit verschiedenen Geschwindigkeiten durchfahren werden.

Definition 7.6 (Median) *Der* MEDIAN *ist der Wert einer Reihe, von dem aus gesehen höchstens die Hälfte der Reihe einen kleineren Wert besitzt und zugleich höchstens die Hälfte der Reihe einen größeren Wert besitzt. Es handelt sich also um den Wert in der Mitte einer sortierten Liste. Ergeben sich zwei Medianwerte, wird deren Mittelwert genommen.*

Der Median hat zuweilen eine größere Aussagekraft als die anderen Mittel. Erstens verschieben einige extreme Werte am Rande der sortierten Liste den Wert des Median nicht und zum anderen wird zumindest für Reihen ungerader Länge ein tatsächlich vorkommener Wert der Reihe geliefert und nicht ein berechneter virtueller Wert. Für zahlreiche Anwendungen ist es wichtig, dass das Mittel auch tatsächlich als Wert vorhanden ist.

Definition 7.7 (Absolutes Mittel) *Das* ABSOLUTE MITTEL *einer Wertereihe* $(x_i)_{i\in\{1,\ldots,n\}}$ *der Länge n berechnet sich wie folgt:*

$$d_{abs}((x_i)_{i\in\{1,\ldots,n\}}) = \frac{1}{n}\sum_{i=1}^{n}|x_i|$$

Die durchschnittliche Auslenkung einer Lautsprechermembran ist ein Grad für die mittlere Lautstärke. Da die Elongation jedoch um den Wert 0 schwankt und wir eine negative Auslenkung als genauso stark empfinden wie eine positive, berechnen wir statt des arithmetischen Mittels das absolute Mittel als weiteres Maß für die Lautstärke eines Musikstückes.

Da zur Berechnung des Median alle Werte der Reihe sortiert werden müssen, ist ein Aufwand von $O(n \log n)$ nötig. Für die Berechnung der übrigen Mittel genügt eine Laufzeit von $O(n)$, da die Werte einmal durchlaufen werden müssen.

Eine Abhandlung, welcher Durchschnitt für welche Anwendung am besten ist, findet man in [56]. Für Musikdaten gilt, dass die Elongation um den Nullpunkt schwankt und die Berechnung aller Durchschnitte daher Null ergibt. Jedoch liefern Varianz und die Berechnung des absolute Mittels interessante Informationen über ein Musikstück. Dies gilt insbesondere nach der Anwendung einer der bereits vorgestellten Transformationen.

7.1.2. Varianz und Standardabweichung

Alleine durch Kenntnis der Durchschnittswerte und der Varianz von zwei Reihen sind Aussagen und Vergleiche über die Reihen möglich. Einfache Vergleiche wie „Reihe (a) hat durchschnittlich größere Werte, dafür schwanken die Werte von Reihe (b) mehr“ können erste Hinweise auf die zugrundeliegende Struktur der Daten und inhärente Muster liefern.

Definition 7.8 (Varianz) *Sei $(x_i)_{i\in\{1,\ldots,n\}}$ eine Wertereihe und d der ermittelte Durchschnitt. Die* Varianz σ^2 *der Reihe berechnet sich wie folgt:*

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(d - x_i)^2$$

Eine weitere häufig benutzte Größe ist die *Standardabweichung* σ, die durch Ziehen der Wurzel aus der Varianz entsteht, also

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d - x_i)^2}$$

Varianz und Standardabweichung können ebenfalls in $O(n)$ berechnet werden. Es hat sich gezeigt, dass die Varianz unterschiedlichster Größen bei der Behandlung von Musikdaten eine große Rolle spielt. Sie gibt die Veränderlichkeit der Daten wieder, ohne selbst von den genauen Zeitpunkten der Veränderungen abzuhängen.

7.1.3. Zentroid

Die bisher behandelten Durchschnitte, insbesondere aber das arithmetische Mittel, repräsentieren den Schwerpunkt der Reihe in der y-Achse, also der Dimension der Werte. Häufig interessiert man sich auch für die Stelle auf der x-Achse, die die häufigsten Werte repräsentiert. Dies ist häufig bei Reihen der Fall, welche die Werte einer Verteilung beinhalten.

Definition 7.9 (Zentroid) *Ein Funktional, welches den mit den Werten einer Reihe gewichteten Mittelwert der Indexvariablen liefert, heißt* Zentroidfunktional *und wird wie folgt berechnet:*

$$c = \frac{\sum\limits_{i=1}^{n} x_i \cdot index(i)}{\sum\limits_{i=1}^{n} x_i}$$

Abbildung 7.1 auf der nächsten Seite zeigt die Wertereihe eines Chromatographen. Die waagerechte Linie kennzeichnet den Wert des arithmetischen Mittels, die senkrechte den Wert des Zentroidfunktionals. Der Schnittpunkt beider Linien entspricht dem Schwerpunkt des 2-dimensionalen Graphen. Die Berechnung des Zentroiden geschieht mit Aufwand $O(n)$.

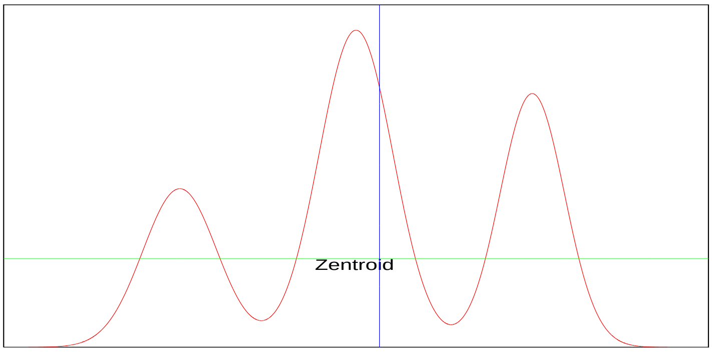


Abbildung 7.1.: Die Wertereihe eines Chromatographen. Die waagerechte Linie gibt den Wert des arithmetischen Mittels wieder, die senkrechte das Ergebnis des Zentroidfunktionals. Der Schnittpunkt entspricht dem Zentroiden des Graphen.

7.1.4. Amplitude

Amplituden sind bei Schwingungsvorgängen interessant. Sie kennzeichnen die größte Auslenkung, die das System in jeder Richtung vollzieht. Bei Musikdaten entspricht die Amplitude der maximal erreichten Lautstärke eines Liedes. Die Elongation hingegen ist die momentane Auslenkung einer Schwingung.

Definition 7.10 (Amplitude) *Die* Amplitude *entspricht dem größten Betrag eines Wertes der Reihe, also:*

$$amp((x_i)_{i\in\{1,\ldots,n\}}) = \max\{|x_i| \mid i \in 1,\ldots,n\}$$

7.1.5. Minima und Maxima

Für viele Anwendungen ist es sinnvoll, den minimalen oder maximalen Wert einer Wertereihe zu kennen. Der maximale Wert eines Frequenzspektrums entspricht der Intensität der stärksten Frequenz. In einer Reihe von Verkaufswerten etwa gibt das Maximum den stärksten Abverkauf, das Minimum die geringste Verkaufszahl wieder. In Kombination mit dem Durchschnitt und der Varianz bekommt man durch die Kenntnis der Grenzen einer Reihe eine ungefähre Vorstellung von ihrer räumlichen Entwicklung.

Über die zwei Funktionale min und max hinaus ist oft die Stelle der Indexwerte der betreffenden Werte gewünscht. Im Beispiel des Frequenzspektrums interessiert man sich über die Intensität hinaus meist auch noch für die Größe der stärksten Frequenz. Bei den Verkaufsdaten ist der genaue Zeitpunkt der minimalen und maximalen Verkäufe sicher interessant. Daher werden zusätzlich noch die Funktionale

$$\min_{index}((x_i)_{i\in\{1,\ldots,n\}}) = index(\arg\min_i((x_i)_{i\in\{1,\ldots,n\}}))$$

und analog

$$\max_{index}((x_i)_{i\in\{1,\ldots,n\}}) = index(\arg\max_i((x_i)_{i\in\{1,\ldots,n\}}))$$

definiert, die die Stelle des Minimums und des Maximums in der Indexdimension liefern.

7.1.6. Länge der Wertereihe

Einige Hörer bevorzugen lange Stücke epischen Ausmaßes, andere hingegen eher kurze Lieder. Populärmusik, welche im Radio gesendet werden soll, darf eine Länge von 3.30 Minuten nicht überschreiten. Für viele Hörer ist die Länge des Musikstückes ein wichtiges Kriterium. Und auch bei der Genreklassifikation scheint die Länge eine große Rolle zu spielen: Klassische Arien mit einer Länge von einer halben Stunde sind nichts ungewöhnliches, Rockstücke dieser Länge allerdings schon!

Ein weiteres und besonders einfaches Funktional liefert also die Länge n der gegebenen Wertereihe. Bei Musikstücken entspricht n der Zahl der sample points des Liedes.

7.2. Intervall-Merkmale

In Kapitel 5 wurden Methoden vorgestellt, Intervalle innerhalb der verschiedenen Dimensionen von Wertereihen zu finden. Ein Intervall besteht aus den vier Größen Startwert, Endwert, Typ und Dichte. Sind diese Intervalle einmal gefunden und wurde die Wertereihe mit ihnen ausgezeichnet, so werden Funktionale benötigt, die die Eigenschaften der Intervalle als Merkmale liefern.

7.2.1. Einfaches Intervallfunktional

Die wenigsten Lernverfahren und Ähnlichkeitsmaße können mit Intervallen arbeiten. Zur Berechnung von Maßen wie dem euklidischen Abstand zwischen zwei Reihen können die vier Größen eines Intervalls jedoch herangezogen werden: Zwei Reihen sind sich um so ähnlicher, je kleiner der Abstand zwischen den Werten ihrer Intervalle ist.

Definition 7.11 *Ein Funktional, welches für jedes Intervall einer Reihe die Werte für Start, Ende, Typ und Dichte liefert, heißt* EINFACHES INTERVALLFUNKTIONAL.

Bei k Intervallen in den Wertedimensionen ergeben sich also $4k$ Zahlen als Ergebnis. Da das Ergebnis des einfachen Intervallfunktionals als Merkmale für Lernverfahren bzw. als Basis für ein Ähnlichkeitsmaß benutzt werden kann, kommen für die Anwendung dieses Funktionals lediglich Intervalle in den Wertedimensionen in Frage. Es wurde gezeigt, dass die gleiche Anzahl von Intervallen in der Indexdimension mit den in Kapitel 5 vorgestellten Verfahren nicht gesichert werden kann. Um weitere Merkmale aus den Intervallen einer Reihe zu gewinnen bietet sich die Anwendung einer Intervall-Transformation (siehe Abschnitt 3.5) oder das im nächsten Abschnitt vorgestellte statistische Intervallfunktional an.

7.2.2. Statistisches Intervallfunktional

Das nun vorgestellte Funktional bestimmt statistische Kenngrößen aus den vorliegenden Intervallen. Da die Berechnung dieser Werte nicht von der Anzahl der Intervalle abhängt, gelingt die Anwendung sowohl in den Wertedimensionen als auch in der Indexdimension. Die Berechnung wird dabei auf allen Intervallen einer Dimension durchgeführt.

Definition 7.12 *Ein Funktional, welches für alle Intervalle einer Dimension die Werte*

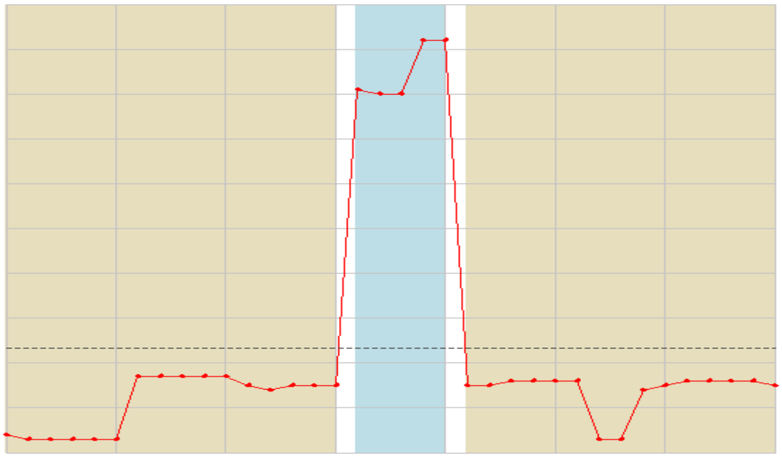


Abbildung 7.2.: Eine Reihe, die mit Intervallen in der Indexdimension ausgezeichnet wurde. Das statistische Intervallfunktional liefert die Werte: 3 Intervalle, eine durchschnittliche Länge von $12\frac{1}{3}$ und einen Typ von $1\frac{1}{3}$.

- *Anzahl der Intervalle*
- *durchschnittliche Länge der Intervalle und die Varianz der Längen*
- *durchschnittlicher Typ und die Varianz der Typen*

berechnet, heißt STATISTISCHES INTERVALLFUNKTIONAL.

Abbildung 7.2 zeigt eine mit Intervallen in der Indexdimension ausgezeichnete Reihe. Das statistische Intervallfunktional liefert für diese Reihe die Anzahl von 3 Intervallen mit einer durchschnittlichen Länge von $12\frac{1}{3}$ und einem Typendurchschnitt von $1\frac{1}{3}$. Die durchschnittliche Länge der Intervalle und der durchschnittliche Typ können zudem mit der Dichte der Intervalle sowohl proportional als auch reziprok gewichtet werden. Dadurch wird ermöglicht, dass Intervalle mit besonders vielen Werten einen größeren Einfluss auf das Ergebnis haben. Andererseits können wahlweise auch die schwachen Intervalle stärker gewichtet werden, damit z. B. der Typ schwacher (seltener) Intervalle stärker berücksichtigt wird. Dies könnten gerade die interessantesten Intervalle sein. Bei Musikdaten ist häufig zu beobachten, dass Hörer Lieder nur aufgrund einer einzigen, durchaus kurzen, Stelle bevorzugen.

7.3. Funktionscharakteristika

Da ein Funktional eine Abbildung vom Raum der Funktionen auf die reellen Zahlen ist, liegt es nahe, auch Extrema zu berechnen. Der Bildbereich enthält die Stellen der Reihe, an denen ein Extremum liegt. Zwischen je zwei Extrema liegt ein Wendepunkt. Auf eine genauere Bestimmung der Wendepunkte wird allerdings verzichtet.

Es werden zwei Verfahren vorgestellt, die sich für verschiedene Anwendungen unterschiedlich gut eignen: das *Peak-Funktional* und das Extrahieren der Werte nach einer *Extrematransformation*.

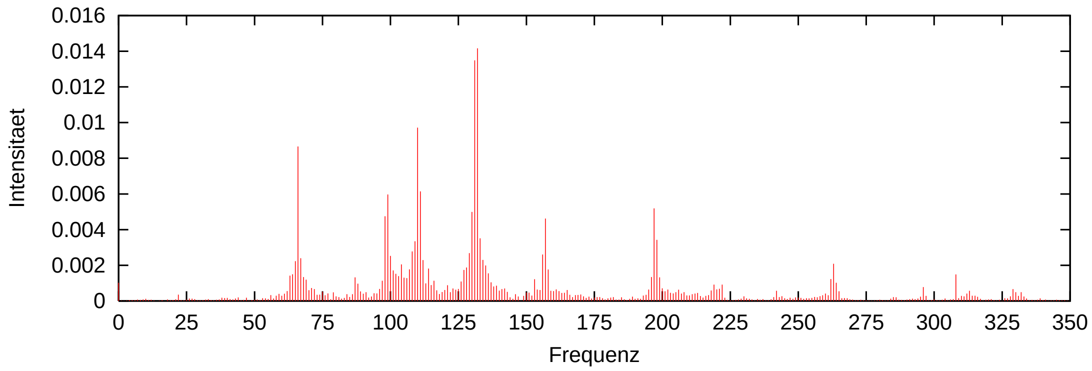


Abbildung 7.3.: Das Frequenzspektrum eines Liedes (vergleiche Abschnitt 3.3). Die x-Achse entspricht den Frequenzen des Liedes, die y-Achse den Intensitäten, mit denen diese Frequenzen auftreten.

Peak	Intensität	Frequenz (Hz)	Breite (Hz)
1	0.0142	132	27
2	0.0097	110	11
3	0.0087	66	72
4	0.0060	99	17
5	0.0052	197	63

Tabelle 7.1.: Die Werte der fünf höchsten Peaks des Frequenzspektrums in Abbildung 7.3. Durch eine Anpassung des Schwellwertes erreicht man ein für die Anwendung optimiertes Ergebnis.

7.3.1. Finden von Peaks

Das Finden von Peaks, also einzelnen ausgeprägten Hügeln innerhalb einer Reihe, ist vor allem bei Verteilungen oder Spektren sinnvoll. Ein Peak kann durch seine Lage in der Indexdimension, seine Höhe bzw. Amplitude in der Wertedimension und seine Breite charakterisiert werden.

Definition 7.13 (k-Peaks-Funktional) *Das k-Peaks-Funktional liefert für eine Wertereihe $(x_i)_{i\in\{1,\dots,n\}}$ die Stelle, die Höhe und die Breite der k höchsten Peaks.*

Die k höchsten Peaks einer Reihe können mit einem Divide-and-Conquer Ansatz gefunden werden. Dieser sucht zunächst das Maximum und bestimmt damit den höchsten Peak. Sodann werden rekursiv die Peaks links und rechts bestimmt. Die Werte des aktuellen Peaks werden dabei ausgeschlossen. Zum aktuellen Peak gehören alle benachbarten Werte, die einen Schwellwert nicht überschreiten. Die Höhe, die Stelle und die Breite der k stärksten Peaks werden dann als Merkmale bereitgestellt. Damit ergeben sich für alle Instanzen einheitlich $3k$ Merkmale.

Die höchsten Peaks eines Frequenzspektrums entsprechen den stärksten Frequenzen eines Liedes. Diese können durch besonders häufige Töne oder auch besonders häufige Oberschwingungen der Instrumente entstanden sein. Sie sind charakteristisch für die Klangfarbe eines Musikstückes und eignen sich daher als Merkmal für Musikstücke. Abbildung 7.3 zeigt einen Ausschnitt aus dem Frequenzspektrum eines Liedes, welches mit der in Abschnitt 3.3 beschriebenen Fourier-Transformation erzeugt wurde. Tabelle 7.1 fasst das Ergebnis eines 5-Peak-Funktionals zusammen. Die fünf höchsten Peaks des Spektrums entsprechen denen, die auch ein Betrachter ad hoc auswählen würde. Eine Verbesserung des Ergebnisses kann durch eine Anpassung des Schwellwertes erzielt werden. So werden weitere oder weniger Werte in jeden Peak aufgenommen. Mit dem Peak-Funktional können natürlich auch die Peaks von anderen Spektren oder Verteilungen identifiziert werden.

Das Aufteilen der Reihe kann maximal $\log n$ mal vorgenommen werden, nach jeder Aufteilung werden insgesamt alle n Werte der Reihe durchlaufen, um in den Hälften nach dem Maximum zu suchen. Wie auch bei anderen Divide-and-Conquer Ansätzen ergibt sich also eine Gesamtlaufzeit von $O(n \log n)$.

7.3.2. Finden von Extrema

Während das Peak-Funktional für das Finden von Peaks in Verteilungen und Spektren optimiert wurde, können mit Hilfe des Extrema-Filters beliebige Minima und Maxima extrahiert werden (vergleiche Abschnitt 4.6). Indem man die Extrema mittels einer Transformation herausarbeitet, behält man sowohl die Informationen über Höhe und Auftreten der Extrema als auch ihre Ordnung innerhalb der Reihe. Dadurch ist es möglich, über das Finden der Extrema hinaus auch weitere Merkmale zu extrahieren.

Betrachten wir noch einmal Abbildung 4.5 auf Seite 42. Die Werte der Extrema blieben erhalten, die übrigen Werte wurden auf Null gesetzt. Zunächst ist es möglich, ein sehr einfaches Funktional zu definieren, welches die deutlichsten Maxima und Minima liefert:

Definition 7.14 (k-Extrema-Funktional) *Das k-Extrema-Funktional liefert für eine Reihe* $(x_i)_{i \in \{1,\ldots,n\}}$ *die Stelle und die Höhe der k Extrema mit dem höchsten Betrag.*

Dieses Funktional ist eng verwandt mit dem k-Peaks-Funktional, jedoch liefert es auch Minima. Da in einer periodischen Reihe wie Musikdaten keine Peaks im oben beschriebenen Sinn auftreten, können auch keine Breiten extrahiert werden.

Nach Anwendung des Extrema-Filters ist es darüberhinaus möglich, Differenzen zwischen den Extrema sowohl bezüglich ihrer Werte als auch ihrer Position zu bestimmen. Die Differenzen zwischen den Extrema eines Liedes ergeben einen ersten Hinweis auf die Geschwindigkeit.

Definition 7.15 (Durchschnittlicher Abstand der Extrema) *Sei* $E_x = \{x_i | x_i \neq 0\}$ *die Menge der Extrema der Zeitreihe* $(x_i)_{i \in \{1,\ldots,n\}}$*, also derjenigen Werte* x_i*, die nach Anwendung des Extrema-Filters nicht 0 sind. O.B.d.A. seien diese in der Reihenfolge ihrer ursprünglichen Indizes mit* $e_1, \ldots, e_m$ *benannt. Der* DURCHSCHNITTLICHE ABSTAND DER EXTREMA *ist dann*

$$AVG_{diff}(E_x) = \frac{1}{|E_x|} \sum_{i=2}^{|E_x|} (index(e_i) - index(e_{i-1}))$$

Die Varianz der Differenzen errechnet sich kanonisch:

$$VAR_{diff}(E_x) = \frac{1}{|E_x|} \sum_{i=2}^{|E_x|} ((index(e_i) - index(e_{i-1})) - AVG_{diff}(E_x))^2$$

Im Rahmen der Musikdaten kann diese Varianz als Maß für den Abwechslungsreichtum benutzt werden: Sehr gleichförmige Musik besitzt eine sehr kleine Varianz der Differenzen.

7.4. Steigung einer Regressionsgeraden

Ist eine Punktmenge in einem 2-dimensionalen Raum gegeben, so kann man versuchen, diese Punkte durch eine Ausgleichsgerade zu beschreiben. Ist der funktionale Zusammenhang zwischen den Dimensionen von linearer Natur und Rauschen und Messfehler relativ klein, so liegen die Punkte auf oder nahe bei der Gerade. Jeder Wert der Wertereihe zusammen mit seiner Stelle in der Indexdimension bildet einen Punkt in einem 2-dimensionalen Raum. Für eine Reihe $(x_i)_{i\in\{1,\dots,n\}}$ der Länge n erhalten wir also n Punkte $(index(i), x_i)$ für $i \in \{1,\dots,n\}$. Gesucht ist ein funktionaler Zusammenhang, beschränkt auf die Funktionsklasse der linearen Funktionen, d. h. wir suchen zwei Parameter m und b mit

$$\hat{x}_i = b + m \cdot index(i)$$

so dass der quadratische Fehler

$$Q(m,b) = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2$$

minimiert wird. Durch partielle Ableitung der Funktion Q nach den Parametern m und b ermitteln wir die Werte dieser Parameter für ein minimales Q. Dazu setzen wir $\hat{x}_i$ ein und stellen die Gleichung um:

$$\begin{aligned}
Q(m,b) &= \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 \\
&= \sum_{i=1}^{n} (x_i - (b + m \cdot index(i)))^2 \\
&= \sum_{i=1}^{n} (x_i^2 - 2bx_i - 2mx_i \cdot index(i) + 2mb \cdot index(i) + m^2 \cdot index^2(i) + b^2) \\
&= \sum_{i=1}^{n} x_i^2 - 2b\sum_{i=1}^{n} x_i - 2m\sum_{i=1}^{n} x_i \cdot index(i) + 2mb\sum_{i=1}^{n} index(i) \\
&\quad + m^2 \sum_{i=1}^{n} index^2(i) + nb^2
\end{aligned} \tag{7.1}$$

Gleichung 7.1 leiten wir zunächst nach b ab und erhalten

$$\frac{\partial Q(m,b)}{\partial b} = -2\sum_{i=1}^{n} x_i + 2m\sum_{i=1}^{n} index(i) + 2nb$$

Um Q zu minimieren muss die erste Ableitung gleich 0 sein. Wir setzen das Ergebnis der Ableitung gleich 0 und lösen nach b auf:

$$\begin{aligned}
& 0 = -2\sum_{i=1}^{n} x_i + 2m\sum_{i=1}^{n} index(i) + 2nb \\
\Leftrightarrow \quad & 2nb = 2\sum_{i=1}^{n} x_i - 2m\sum_{i=1}^{n} index(i) \\
\Leftrightarrow \quad & b = \frac{1}{n}\sum_{i=1}^{n} x_i - m \cdot \frac{1}{n}\sum_{i=1}^{n} index(i)
\end{aligned} \tag{7.2}$$

Der y-Achsenabschnitt b der Regressionsgeraden kann also berechnet werden durch die Subtraktion von m multipliziert mit dem arithmetischen Mittel der Indexwerte der Reihe von dem arithmetischen Mittel der Reihenwerte. Um die Steigung m zu bestimmen, leiten wir Gleichung 7.1 nach m ab und erhalten:

$$\frac{\partial Q(m,b)}{\partial m} = -2\sum_{i=1}^{n}(x_i \cdot index(i)) + 2m\sum_{i=1}^{n} index^2(i) + 2b\sum_{i=1}^{n} index(i)$$

Wir setzen das Ergebnis für b in die Gleichung ein und setzen sie ebenfalls gleich 0, bevor wir nach m auflösen:

$$\begin{aligned}
& 0 = -2\sum_{i=1}^{n}(x_i \cdot index(i)) + 2m\sum_{i=1}^{n} index^2(i) \\
& \qquad +2\left(\frac{1}{n}\sum_{i=1}^{n} x_i - m\cdot\frac{1}{n}\sum_{i=1}^{n} index(i)\right)\cdot\sum_{i=1}^{n} index(i) \\
\Leftrightarrow\quad & 0 = -2\sum_{i=1}^{n}(x_i \cdot index(i)) + 2m\sum_{i=1}^{n} index^2(i) \\
& \qquad +\frac{2}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} index(i) - \frac{2}{n}\cdot m\cdot\sum_{i=1}^{n} index(i)\sum_{i=1}^{n} index(i) \\
\Leftrightarrow\quad & \frac{1}{n}\cdot m\cdot\sum_{i=1}^{n} index(i)\sum_{i=1}^{n} index(i) - m\sum_{i=1}^{n} index^2(i) = \\
& = \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} index(i) - \sum_{i=1}^{n}(x_i \cdot index(i)) \\
\Leftrightarrow\quad & m = \frac{\frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} index(i)}{n} - \sum_{i=1}^{n}(x_i \cdot index(i))}{\frac{\sum_{i=1}^{n} index(i)\sum_{i=1}^{n} index(i)}{n} - \sum_{i=1}^{n} index^2(i)}
\end{aligned} \tag{7.3}$$

Mit Gleichung 7.3 lässt sich die Steigung m nur aus den Werten x_i bzw. aus den Indexwerten $index(i)$ einer Reihe $(x_i)_{i\in\{1,\ldots,n\}}$ berechnen. Mit dieser Steigung m und Gleichung 7.2 kann dann der y-Achsenabschnitt b der Regressionsgerade berechnet werden. Zusätzlich wird als Merkmal auch die Abweichung (*Diskrepanz*) d der Werte von der berechneten Regressionsgeraden ermittelt. Mit Hilfe der linearen Regression, die auch auf mehr als 2 Dimensionen erweitert werden kann, ist die Bildung eines Funktionals möglich:

Definition 7.16 (Regressions-Funktional) *Ein Funktional, welches zu einer gegebenen Reihe $(x_i)_{i\in\{1,\ldots,n\}}$ eine lineare Regressionsgerade bestimmt und Steigung m, y-Achsenabschnitt b und die Diskrepanz d als Merkmale liefert, heißt lineares* REGRESSIONS-FUNKTIONAL.

Ein Regressions-Funktional kann bei vielen Wertereihen zur Bestimmung eines Trends eingesetzt werden. Bei Musikdaten, die ja keinen Trend besitzen, wird das Funktional eher nach Anwendung von Transformationen benutzt: Die Steigung der Extrema im Frequenzspektrum liefert ein Merkmal, welches auch als *Spektrale Neigung* bekannt ist und die Berechnung der Steigung nach einer Fensterung repräsentiert den Verlauf des Fensterfunktionals innerhalb des Musikstückes. Berechnet man in jedem Fenster z. B. die durchschnittliche Lautstärke, so liefert die Steigung einen Hinweis, ob das Lied stets lauter oder leiser wird oder etwa gleich bleibt.

7.5. Funktional-Kombination

Häufig können weitere Merkmale aus bereits bekannten Merkmalen gewonnen werden durch einfache Kombinationen der bekannten Merkmale mittels mathematischer Operatoren [45]. Im nächsten Abschnitt werden einige Merkmale diskutiert, die als Quotient der Werte bereits bekannter Funktionale definiert sind.

Es wird sich später für die Automatisierung noch als günstig erweisen, die Kombination von Funktionalen oder die Anwendung eines mathematischen Operators auf einem Funktional als Baum zu repräsentieren:

Definition 7.17 (Kombinations-Funktional) *Die Anwendung eines mathematischen Operators auf andere Funktionale heißt* KOMBINATIONS-FUNKTIONAL. *Der Operator ist Wurzel eines Baumes, die Operation wird auf die Kinder angewendet.*

Diese Definition erlaubt auch weitere Kombinations-Funktionale als Kinder und einen rekursiven Aufbau komplexer Funktionale. Je nach Operator hat ein Kombinations-Funktional ein oder zwei Kinder. Operationen wie Sinus oder Logarithmus besitzen nur ein Kind, arithmetische Operationen wie Addition oder Multiplikation besitzen jedoch zwei Kinder. Die möglichen Operationen auf einem Wert w sind $\log w, \sin w$ und die Bildung des reziproken Wertes $\frac{1}{w}$. Operationen auf zwei Werten w_1 und w_2 umfassen $w_1 + w_2, w_1 - w_2, w_1 \cdot w_2$ und $\frac{w_1}{w_2}$.

7.6. Audiomerkmale

Bisher wurde bewusst auf Merkmale verzichtet, die aus dem Bereich der psychoakustischen Forschung stammen. Das Ziel war, die bekannten und gut untersuchten Methoden der Zeitreihenanalyse und ihre Kombination zur Erzeugung der Merkmale zu verwenden. An dieser Stelle sollen jedoch noch einige Merkmale definiert werden, die ihren Ursprung in der digitalen Sprachverarbeitung haben und bereits auf Audiodaten angewendet wurden [21]. Allen ist gemein, dass sie auf einfache Weise durch eine Kombination der bisher beschriebenen Methoden konstruiert werden können.

Definition 7.18 (Spectral Flatness Measure) *Ein Funktional, welches aus einem Spektrum* $(x_i)_{i \in \{1,\ldots,n\}}$ *das Verhältnis zwischen geometrischem und arithmetischem Mittel bestimmt, heißt* SPECTRAL FLATNESS MEASURE-*Funktional (SFM).*

Eine Realisierung kann mit Hilfe des im vorigen Abschnitt definierten Kombinations-Funktionals mit dem geometrischen und arithmetischen Mittel geschehen. Die Anwendung des SFM-Funktionals geschieht – wie der Name bereits andeutet – normalerweise auf einem Spektrum. Das geometrische Mittel ist klein, sobald einer der Werte der Reihe klein ist. Einzelne hohe Peaks in einem Frequenzspektrum liefern trotzdem ein kleines geometrisches Mittel, da auch sehr kleine Werte neben den Peaks vorkommen. Ein tonales Frequenzspektrum resultiert daher in einem SFM-Wert nahe bei 0. Besitzt das Spektrum jedoch keine einzelnen Peaks sondern durchweg hohe Werte, ergibt sich ein Wert nahe 1. Das Spectral Flatness Measure Funktional kann also gut zur Erkennung von Rauschen eingesetzt werden. Da Musikdaten jedoch meist tonal sind und in praktisch jedem Musikstück eine Frequenz gar nicht vorhanden ist, ergibt dieses Merkmal für Musikdaten stets 0. Daher ist es kaum für andere Klassifikationsaufgaben als die Unterscheidung zwischen Rauschen und tonalen Signalen geeignet.

Definition 7.19 (Spectral Crest Factor) *Ein Funktional, welches aus einer gegebenen Reihe* $(x_i)_{i\in\{1,\ldots,n\}}$ *das Verhältnis zwischen Maximum und arithmetischem Mittel bestimmt, heißt* SPECTRAL CREST FACTOR-*Funktional (SCF).*

Erneut ist die Realisierung durch die Verwendung des Kombinations-Funktionals mit den betreffenden Funktionalen einfach. Der spektrale Spitzenfaktor besitzt einen hohen Wert, wenn sich das Maximum des Spektrums weit vom Durchschnitt abhebt. Dieses Funktional misst also, wie klar einzelne Signale herauskommen. Anders als das Spectral Flatness Measure bildet dieses Funktional auch für Klassifikationsaufgaben wie die Genreunterscheidung ein sinnvolles Merkmal.

Auch typische Transformationen, wie z. B. die Berechnung der *real cepstral coefficients* (RCC) sind auf einfache Weise durch die bis jetzt behandelten Methoden zu realisieren[1]. Für die Anwendung der RCC führt man in einer Fensterung die inverse Fourier-Transformation auf den Logarithmus des Frequenzspektrums des Fensters aus. Die nahe verwandte *mel-frequency cepstral coefficients* (MFCC) Transformation gewichtet die Frequenzen vor Anwendung der inversen Fourier-Transformation noch nach psychoakustischen Gesichtspunkten (siehe Abschnitt 4.5.2) [13]. Die Methoden der Zeitreihenanalyse und die vorgestellte Systematisierung eignen sich also gut dazu, bekannte Transformationen und Merkmalsextraktionen aus Audiodaten zu realisieren. Durch die allgemeine Fensterung können zudem eine Vielzahl weiterer Transformationen und Merkmalsextraktoren synthetisiert werden.

[1] Der Begriff *cepstral* ist ein Kunstwort, das durch Vertauschung der Konsonanten s und c in dem Wort *spectral* entsteht. Analog spricht man auch vom *cepstrum*.

KAPITEL 8

Automatisierte Merkmalsextraktion mittels Genetischer Programmierung

Bisher erfolgte eine Systematisierung aller Methoden, mit deren Hilfe wir Merkmale aus Wertereihen für Klassifikationsaufgaben extrahieren können. Diese Methoden wurden in den letzten Kapiteln ausführlich vorgestellt. Durch diese Systematisierung wird es Benutzern ermöglicht, einfach und schnell Ketten zusammenzustellen, um Merkmale aus den vorliegenden Daten zu extrahieren. In diesem Kapitel soll ein Verfahren entwickelt werden, mit dem die Extraktion von Merkmalen automatisiert durchgeführt werden kann. Der Benutzer soll lediglich die Lernaufgabe und die zugrundeliegenden Daten definieren. Durch die Strukturierung des Raums der Methoden wird eine automatische Suche nach der optimalen Kette zur Merkmalsextraktion ermöglicht. Dabei sollen die Methoden so ausgewählt und miteinander kombiniert werden, dass ein Merkmalsvektor generiert wird, der sich möglichst gut zur Bearbeitung der gegebenen Lernaufgabe eignet. Das in dieser Arbeit vorgestellte Verfahren zur automatisierten Merkmalsextraktion aus umfangreichen Wertereihen wird mit Hilfe von *genetischer Programmierung* [23] realisiert.

8.1. Evolutionäre Algorithmen

Alle evolutionären Algorithmen, zu denen auch die genetische Programmierung zählt, verwalten Suchpunkte im zu durchsuchenden Raum [2]. Im Falle der genetischen Algorithmen oder der Evolutionsstrategien, bei denen der Suchraum meist durch den $\mathbb{R}^m$ oder den $\{0,1\}^m$ gegeben ist, entspricht ein Suchpunkt x einem Vektor in diesem Raum.

Definition 8.1 *Evolutionäre Algorithmen versuchen in einem* SUCHRAUM *den optimalen Suchpunkt für die Lösung einer Aufgabe zu finden. Die Suchpunkte, die ein evolutionärer Algorithmus verwaltet, nennt man* INDIVIDUEN*; verwaltet er mehrere gleichzeitig, so nennt man diese eine* POPULATION*.*

Bei einem evolutionären Algorithmus wird eine Population über eine große Zahl von Generationen verwaltet. In jeder Generation kann es dabei zu mehreren Operationen kommen. Ähnlich dem biologischen Vorbild wird bei einer *Mutation* ein Suchpunkt verändert. Es entsteht durch eine „fehlerhafte" Replikation aus einem Suchpunkt, dem Elter, ein Kind. Dabei können prinzipiell beliebige Kinder entstehen, auch solche, die dem Elter kaum noch ähnlich sind. Allerdings sollten Mutationen zu ähnlichen Kindern eine größere Wahrscheinlichkeit besitzen als solche zu unähnlichen Kindern. Um diese wünschenswerte Eigenschaft der Mutation zu formalisieren, muss auf dem Suchraum eine Metrik d gegeben sein. Für einen Elter x und das durch Mutation entstandene Kind X sollte gelten: $d(x, x_1) < d(x, x_2) \Rightarrow Prob(X = x_1) > Prob(X = x_2)$.

Definition 8.2 (Mutationen) *Unter* MUTATIONEN *verstehen wir alle Operationen, die aus einem Individuum ein anderes ableiten.*

Bei der biologischen *Kreuzung* (*Crossover*) tauschen zwei Elternteile Informationen aus. Die Kinder stellen eine Mischung der Informationen der Eltern dar. Eine sinnvolle Forderung an die Metrik und den Suchraum besteht darin, dass ein Kind x_k, welches aus den Eltern x_1 und x_2 entsteht, nicht weiter von den Eltern entfernt ist als diese voneinander, formal: $d(x_1, x_k) \leq d(x_1, x_2)$ und $d(x_2, x_k) \leq d(x_1, x_2)$.

Definition 8.3 (Crossover) *Unter* KREUZUNGEN *oder* CROSSOVER *verstehen wir alle Operationen, die aus zwei oder mehreren Individuen ein anderes ableiten.*

Ohne einen strukturierten Suchraum ist keine Auswahl geeigneter Mutation- und Kreuzungsoperationen möglich. Eine im $\{0,1\}^m$ häufig angewendete Mutation ist das Kippen einzelner Stellen des Bitvektors, jede Stelle mit einer Wahrscheinlichkeit von $\frac{1}{n}$. Damit sind auch sehr unähnliche Kinder möglich, aber im Erwartungswert kippt nur eine Stelle. Für die Evolutionstrategien im $\mathbb{R}^m$ ändert man die einzelnen Elemente des Zahlenvektors durch Addition oder Multiplikation mit einer normalverteilten Größe. Erneut sind ähnliche Kinder wahrscheinlicher als unähnliche. Das gibt einen Hinweis auf die Mutationen in dem Raum der Wertereihenmethoden: Die Änderung einer Methode sollte in einer Methode des gleichen Typs resultieren.

Ein evolutionäres Verfahren bewegt sich durch Mutationen und Crossover durch den Suchraum und bewertet jeden Suchpunkt, der dabei erzeugt wird, mittels einer *Fitnessfunktion*. Sie definiert die Aufgabe, die mittels des Verfahrens gelöst werden soll. Anhand der Fitness werden die Individuen ausgewählt, die für Kreuzungen oder Mutationen benutzt werden. Sie bestimmt auch, welche Individuen eine Generation überleben und ein weiteres mal zur Bildung neuer Suchpunkte beitragen dürfen. Meist ist es sinnvoll, dass die Wahrscheinlichkeit zum Überleben um so höher ist, je fitter das Individuum ist (*survival of the fittest*).

Definition 8.4 (Selektion) SELEKTION *bezeichnet eine Klasse von Verfahren, die zur Auswahl von Individuen benutzt werden. Häufig gilt, dass Individuen mit einer höheren Fitness mit einer größeren Wahrscheinlichkeit selektiert werden.*

Nachdem die Bausteine eines evolutionären Algorithmus und die Fitnessfunktion festgelegt wurden, arbeitet er solange, bis eine Abbruchbedingung erfüllt ist. Die Lösung ist gut genug, es ist eine vorgegebene Zeit verstrichen oder es gab in den letzten Generationen keine Verbesserungen mehr sind mögliche Abbruchbedingungen. Abbildung 8.1 auf der nächsten Seite zeigt schematisch die Arbeitsweise genetischer Algorithmen.

8.1.1. Genetische Programmierung

Die genetische Programmierung besitzt viele Gemeinsamkeiten mit den bisher betrachteten Evolutionsstrategien oder genetischen Algorithmen. Während letztere einfach strukturierte Räume wie den $\mathbb{R}^m$ oder $\{0,1\}^m$ durchsuchen, ist das Ziel der genetischen Programmierung, endliche Automaten zu erzeugen, die sich gut an die Umwelt anpassen. Wie in der Lerntheorie erhält die genetische Programmierung Beispiele mit der richtigen Antwort und versucht, einen Automaten

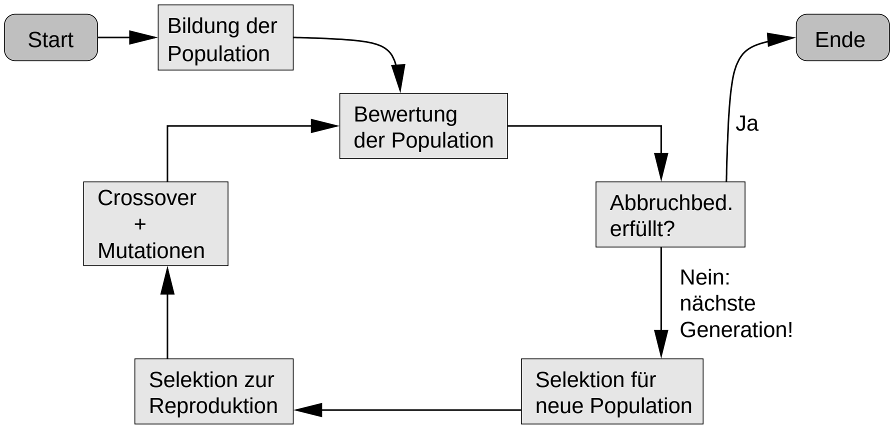


Abbildung 8.1.: Der Ablauf eines evolutionären Algorithmus ist kreisförmig. Zu Beginn wird eine Population aus zufällig generierten Individuen gebildet und bewertet. Solange die Abbruchbedingung nicht erfüllt ist, wird in jeder Generation eine Selektion der Individuen sowie Mutation und Kreuzung der Individuen in Abhängigkeit von ihrer Fitness durchgeführt.

zu erzeugen, der möglichst klein ist und die korrekten Antworten vorhersagt. Nach *Occam's Razor* Theorem werden kleine Automaten dabei bevorzugt. Dieses ist ein äußerst ambitioniertes Ziel, da mit Hilfe von endlichen Automaten unendlich viele Programme simuliert werden können.

Der Suchraum besteht also aus einer Menge von syntaktisch korrekten Programmen, deren Fitness daran gemessen wird, wie gut sie die gegebene Aufgabe erfüllen. Dabei wird nicht gefordert, dass beliebige Programme in beliebigen Sprachen erzeugt werden können. Es bietet sich an, für die Programme Baumdarstellungen zu verwenden; etwa Variablen und Konstanten an den Blättern und mathematische Operationen und Funktionen an den inneren Knoten. Es ist darauf zu achten, dass innere Knoten die korrekte Anzahl von Kindern besitzen. Ein Knoten mit der Sinusfunktion darf nur ein Kind, ein Plusknoten muss jedoch zwei Kinder besitzen. Das Finden der korrekten Zielfunktion ist eine häufige Anwendung der genetischen Programmierung. Wir gehen jedoch einen Schritt weiter und erlauben als Knoten zusätzlich komplexere Operationen: Die Methoden des Wertereihenanalyse.

8.2. Suchraum der Methoden

Die behandelten Methoden der Wertereihenanalyse bilden einen Suchraum, in dem genetische Algorithmen arbeiten können. Gesucht ist eine optimale Auswahl und Kombination von Methoden zur Merkmalsextraktion für das maschinelle Lernen. Ausgehend von den Grundelementen „Transformation“ und „Funktional“ sowie der Möglichkeit, Ketten aus diesen zu bilden und Elemente dieser Gruppen in Fensterungen zu verschachteln und auf diese Weise zu Methodenbäumen zu kommen, kann ein Raum über alle möglichen Kombinationen aufgebaut werden. Zunächst ist es denkbar, jede Methode für sich mit einer Wertereihe zu benutzen. Allerdings werden die Methoden mit dem Ziel angewandt, Merkmale aus Reihen zu extrahieren. Für den resultieren Merkmalsvektor wird also gefordert, dass er nur aus einzelnen, ungeordneten Zahlen besteht. Damit fallen als zulässiges Element des Suchraums alle Methodenkombinationen weg, die kein Funktional enthalten. Wir definieren:

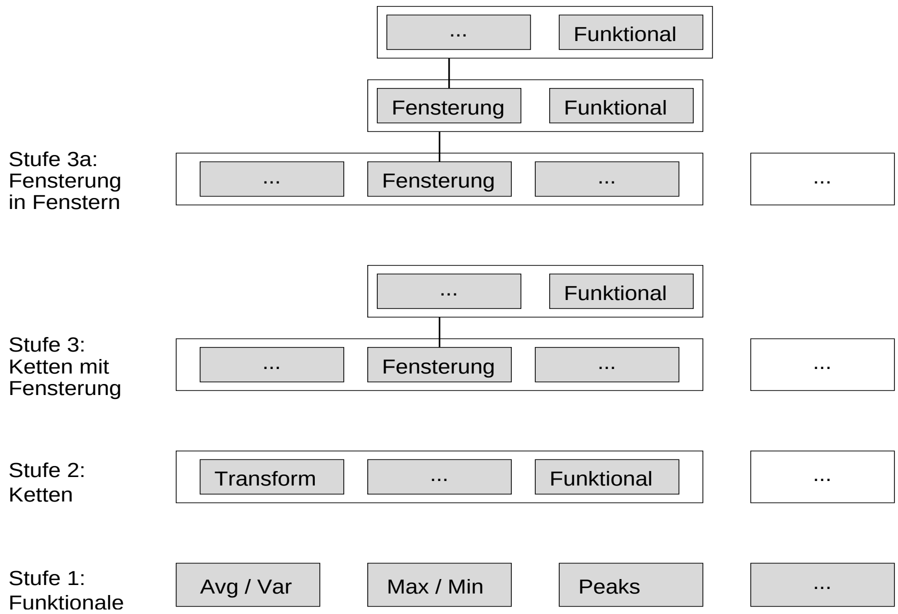


Abbildung 8.2.: Der Suchraum der Methoden. Die Elemente sind Methodenbäume (ohne Wurzel dargestellt). Auf der untersten Stufe befinden sich Bäume, die lediglich ein Funktional als Kind besitzen. Die nächste Stufe erlaubt Ketten mit einem Endfunktional, in Stufe 3 sind auch Fensterungen erlaubt. Die höheren Stufen entstehen durch Schachtelung von Fensterungen bis zu einer maximalen Tiefe von $n-1$.

Definition 8.5 (Kette) *Eine* Kette *besteht aus beliebig vielen Transformationen und einem Funktional am Ende, dem sogenannten* Endfunktional.

Dies reduziert die Menge der Grundelemente zunächst auf alle Funktionale, d. h. Ketten, die nur ein Funktional aber keine Transformationen enthalten. Die nächste Stufe der möglichen Elemente sind Ketten, die mit einem Funktional enden, vorher aber Transformationen anwenden. Die dritte, und bereits letzte Stufe sind Ketten, die Fensterungen enthalten. Fensterungen sind ebenfalls Transformationen, aber sie erweitern den Raum um eine weitere Ebene, da sie beliebige Transformationen enthalten können. Um einheitliche Begriffe verwenden zu können, wird eine „künstliche Fensterung“ definiert. Diese wendet die inneren Methoden nur auf ein einziges Fenster an, welches die komplette Reihe enthält:

Definition 8.6 (Null-Fensterung) *Sei n die Länge der gegebenen Wertereihe. Eine* Null-Fensterung *ist eine Fensterung, welche eine Fensterbreite n und Schrittweite* 1 *besitzt.*

Die inneren Methoden der Null-Fensterung werden also nur einmal auf die vollständige Reihe angewendet. Mit Hilfe der allgemeinen Fensterung, der Null-Fensterung und der Kette kann die Programmstruktur unserer Vorverarbeitung definiert werden:

Definition 8.7 (Methodenbaum) *Ein* Methodenbaum *ist eine Fensterung. Ihre Kinder bilden eine Kette. Enthält die Kette eine Fensterung, so ist diese Wurzel eines neuen Methodenbaums.*

Die Wurzel eines Methodenbaums ist also eine Null-Fensterung. Damit werden die Methoden der Kinder-Kette auf die vollständige Reihe angewendet. Innere Knoten, also innere Bäume, werden durch eine normale Fensterung gebildet. Dann werden die inneren Methoden für jedes Fenster der Fensterung einmal abgearbeitet. Mit jeder weiteren verschachtelten Fensterung vergrößert sich der Suchraum um eine Ebene. Da die betrachteten Eingabereihen jedoch endlich sind, ist auch die Anzahl an zusätzlichen Ebenen, die entstehen können, endlich. Jede Fensterung hat eine maximale Fenstergröße von $n-1$, d.h. die Anzahl der zusätzlichen Ebenen ist durch n beschränkt.

Abbildung 8.2 auf der vorherigen Seite zeigt die ersten Stufen des Suchraums. Im folgenden soll die Größe des Suchraums abgeschätzt werden. Sei T_0 die Anzahl der verwendeten Transformationen und Auszeichner, allerdings ohne die Fensterung. Sei weiter F die Anzahl der Funktionale und n die Länge der Eingabewertereihe. Die Anzahl der Methoden auf Stufe 1 ist F. Betrachten wir als nächstes die Anzahl von Methoden, die sich auf Stufe 2 ergeben, also durch die Bildung von Ketten, die Transformationen und am Ende ein Endfunktional enthalten. Die Auswahl der Transformationen entspricht einer geordneten Ziehung ohne Zurücklegen, es gibt also

$$T_0 \cdot (T_0 - 1) \cdot \ldots \cdot (T_0 - k + 1) = \frac{T_0!}{(T_0 - k)!}$$

Möglichkeiten, eine Folge von Transformationen der Länge k zu bilden. Wir summieren alle möglichen Längen k auf und erhalten

$$\sum_{k=0}^{T_0} \frac{T_0!}{(T_0 - k)!}$$

verschiedene Folgen von Transformationen. Diese Anzahl muss noch mit der Anzahl der Funktionale multipliziert werden, da jede Kette zur Merkmalsgenerierung mit einem Funktional endet:

$$K_0 = F \cdot \sum_{k=0}^{T_0} \frac{T_0!}{(T_0 - k)!} \tag{8.1}$$

Gleichung 8.1 fasst die Anzahl der Kombinationen für Stufe 1 und 2 zusammen, da für $k = 0$, also einer Kette ohne Transformationen, die Summe 1 ergibt. Der nächste Schritt auf Stufe 3 ist nun sehr einfach. Wir überlegen uns zunächst, wieviele verschiedene Fensterungen existieren, die sich durch Benutzen von Transformationen und einem Funktional bilden lassen. Dabei lassen wir zunächst noch keine inneren Fensterungen zu. Wir können direkt das Ergebnis von Gleichung 8.1 benutzen. Es gibt so viele verschiedene Fensterungen, wie es verschiedene Ketten von Transformationen und einem Endfunktional gibt. Jede Fensterung ist für sich ebenfalls eine Transformation und wir definieren eine neue Menge von Transformationen, die aus den übrigen T_0 Transformationen und allen möglichen Fensterungen besteht. Ihre Anzahl ist

$$T_1 = T_0 + K_0 = T_0 + F \cdot \sum_{k=0}^{T_0} \frac{T_0!}{(T_0 - k)!}$$

Damit haben wir Stufe drei erreicht, also die Anzahl aller Transformationen und Fensterungen, aus denen sich Methodenketten bilden lassen.

Als nächstes erlauben wir auch geschachtelte Fensterung und erreichen damit die Stufen über Stufe 3. Wir betrachten noch einmal die Gleichung für T_1 und stellen fest, dass dies der Anzahl der Transformationenketten entspricht, inklusive aller Möglichkeiten für Fensterungen ohne weitere innere Fensterungen. Wir nennen dies die Anzahl der echten Transformationen der Ebene 0.

Wollen wir diese Anzahl von Transformationsmöglichkeiten nun innerhalb einer Fensterung erlauben, also eine neue Schachtelungsebene beginnen, so können wir T_1 dazu benutzen, die Anzahl der Fensterungen ohne und mit *einer* inneren Fensterung zu berechnen:

$$T_2 = T_0 + K_1 = T_0 + F \cdot \sum_{k=0}^{T_1} \frac{T_1!}{(T_1 - k)!}$$

K_1 entspricht der Anzahl aller Fensterungen, mit und ohne maximale einer inneren Fensterung bis zur Schachtelungsebene 1. Dieses rekursive Prinzip setzt sich allgemein fort, es gilt:

$$\begin{aligned} T_3 &= T_0 + K_2 \\ T_4 &= T_0 + K_3 \\ &\vdots \\ T_i &= T_0 + K_{i-1} \end{aligned}$$

und auch

$$\begin{aligned} K_2 &= F \cdot \sum_{k=0}^{T_2} \frac{T_2!}{(T_2 - k)!} \\ K_3 &= F \cdot \sum_{k=0}^{T_3} \frac{T_3!}{(T_3 - k)!} \\ &\vdots \\ K_i &= F \cdot \sum_{k=0}^{T_i} \frac{T_i!}{(T_i - k)!} \end{aligned}$$

Bisher haben wir nur die Anzahlen der Transformationen berechnet, aus denen sich Methodenketten bilden lassen. Wir haben also den Pool aller Transformationen und Fensterungen aufgebaut und kennen die Anzahl aller echten Transformationen – also inklusive den Fensterungen – für eine bestimmte Schachtelungstiefe i. Da jede Kette mit einem Funktional endet, ist die Anzahl der Ketten insgesamt:

$$F \cdot \sum_{k=0}^{T_i} \frac{T_i!}{(T_i - k)!}$$

Wir haben oben bereits begründet, dass spätestens nach einer Schachtelungstiefe von $n-1$ bei einer Reihe von n Werten nicht mehr genug Werte vorhanden sind, um eine weitere Fensterung durchzuführen. Für endliche Reihen endet also der Schachtelungsprozess mit einer maximalen Tiefe von $n-1$:

$$F \cdot \sum_{k=0}^{T_{n-1}} \frac{T_{n-1}!}{(T_{n-1} - k)!} \tag{8.2}$$

Gleichung 8.2 liefert also die Anzahl aller Ketten, ohne Transformationen, mit Transformationen und mit verschachtelten Fensterungen bis zur maximalen Tiefe von $n-1$. Ungewöhnlich ist, dass die Größe des Suchraums nicht nur von der Anzahl der Methoden sondern auch von der Länge der Eingabe abhängt. Daher ist diese Anzahl auch eine obere Schranke für die Anzahl

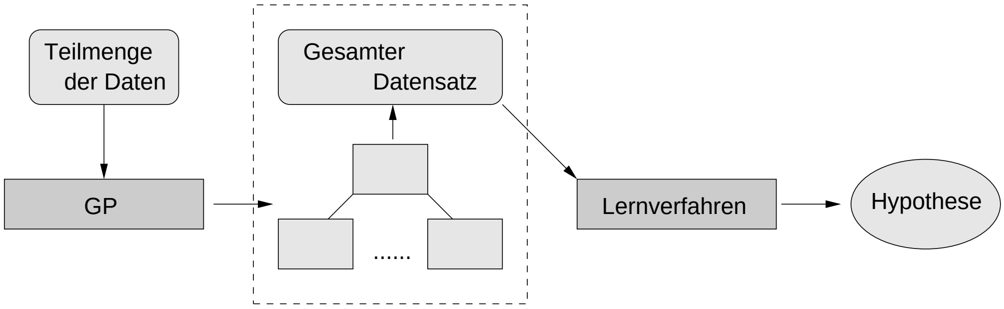


Abbildung 8.3.: Der Ablauf der automatisierten Merkmalsextraktion wird in mehreren Phasen vollzogen. Zunächst wird ein optimaler Methodenbaum auf einer Teilmenge der Daten trainiert der dann auf alle Daten angewendet wird. Die zweite Trainingsphase auf den Merkmalsvektoren aller Instanzen liefert die gesuchte Hypothese.

der Elemente im Suchraum. Kleinere Fenstergrößen sorgen dafür, dass in jeder Ebene mehr Werte der Reihe verloren gehen. Zum Vergleich: Der Suchraum genetischer Algorithmen im $\{0,1\}^m$ hat immerhin eine Größe von 2^m, der Suchraum für Evolutionsstrategien im $\mathbb{R}^m$ ist sogar überabzählbar unendlich.

Die beschriebene Vorgehensweise zur Merkmalsextraktion führt allerdings nur zur einem Merkmal bzw. zu einigen wenigen Merkmalen, die das Endfunktional der Wurzel-Kette liefert. Um diese Anzahl zu vergrößern, kann man die genetische Programmierung mehrfach anwenden oder die in Abschnitt 8.4.2 beschriebene Verzweigung bemühen. Eine weitere Variante erlaubt das Wählen der Transformationen mit Zurücklegen, also mehrfache Anwendung der gleichen Transformation innerhalb einer Kette. Für diese – durchaus sinnvolle – Variante gibt die Berechnung nur eine untere Schranke für den Suchraum der Methoden an. Das gleiche gilt für das baumartige Kombinations-Funktional. Durch beliebige Kombinationen von anderen Funktionalen mit Hilfe mathematischer Operationen wird der Suchraum der Methoden ebenfalls unendlich groß.

8.3. Doppelt Trainieren hält besser

Die automatisierte Merkmalsextraktion kann in mehrere Phasen unterteilt werden. Ziel ist ein optimaler Methodenbaum, der Merkmalsvektoren aus den Daten liefert, mit denen ein Lernverfahren eine möglichst gute Hypothese finden kann (*Training*). Da die genetische Programmierung ebenfalls als Lernverfahren betrachtet werden kann, findet die Merkmalsextraktion in mehreren Trainingsphasen statt:

1. Trainieren der Merkmalsextraktion (GP) auf einer Teilmenge der Daten. Dabei inneres Training zur Fitnessbestimmung (siehe Abschnitt 8.4.5).
2. Anwendung des optimalen Methodenbaums auf allen Daten.
3. Trainieren der Hypothese mit dem Lernverfahren aus den Merkmalsvektoren aller Instanzen.

Abbildung 8.3 zeigt den Ablauf der automatisierten Merkmalsextraktion. Auf einer Teilmenge der Daten wird mit Hilfe der genetischen Programmierung ein optimaler Methodenbaum zur Merkmalsextraktion gesucht. Betrachtet man die genetische Programmierung auch als Lernverfahren, kann man sagen, dass man die Merkmalsextraktion aus den Daten trainiert. Das Ergebnis

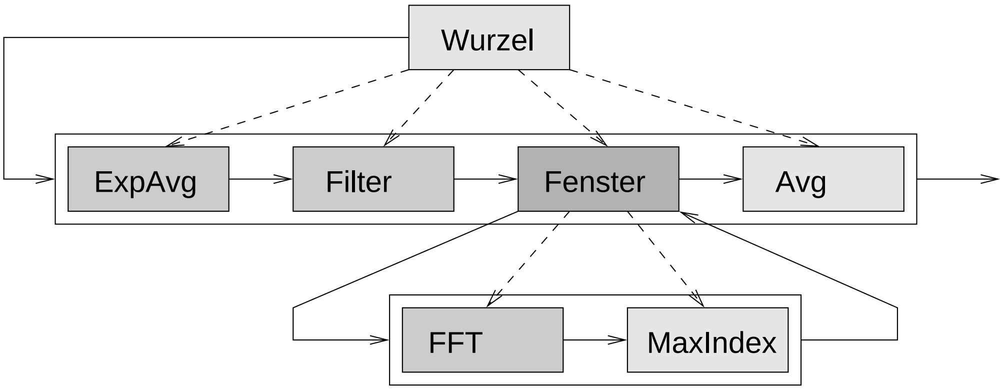


Abbildung 8.4.: Die Wurzel kennzeichnet das Individuum, also den Suchpunkt im Suchraum der Methoden. Der Methodenbaum besitzt vier Kinder, die in einer Kette abgearbeitet werden. Das Endfunktional dieser Kette liefert das gesuchte Merkmal. Das dritte Kind ist eine Fensterung und vergrößert die Tiefe des Baums um 1. Die gestrichelten Pfeile geben die Verschachtelung der Methoden innerhalb des Methodenbaums wieder, die durchgezogenen Pfeile den Datenfluss durch den Baum.

ist ein optimaler Methodenbaum der auf dem gesamten Datensatz angewendet wird (gestrichelter Kasten). Der Methodenbaum wird auf jede Instanz angewendet und damit eine Transformation der Repräsentationssprache von L_E zu L'_E durchgeführt. Dieser neue Datensatz wird benutzt, damit ein Lernverfahren eine Hypothese finden kann. Dies ist die zweite Trainingsphase in diesem Ablauf. Das Resultat kann dann auf ungesehene Instanzen angewendet werden, die ihrerseits mit Hilfe des Methodenbaums in einen Merkmalsvektor transformiert wurden.

8.4. Programmierung im Raum der Methoden

Ein Element des Suchraums, also ein Methodenbaum, zeigt Abbildung 8.4. Der dargestellte Baum ist ein Programm welches als Eingabe eine Wertereihe erhält und als Ausgabe einen Vektor von Merkmalen liefert. Die Knoten stellen aber nicht mathematische Operatoren und Funktionen dar, sondern die in den letzten Kapiteln vorgestellten Methoden. Die Abarbeitung geschieht wie bei einer Tiefensuche, die Geschwister werden jeweils von links nach rechts durchlaufen. Eine Gruppe von Geschwistern eines Methodenbaumes bildet eine Kette von Methoden.

Die Ausnahme in der Reihenfolge der Abarbeitung bildet lediglich eine Fensterung: Ihre Kinder werden für jedes Fenster einmal abgearbeitet. Ein Knoten des Baumes, der eine Fensterung darstellt, ist ein neuer Startpunkt für die Tiefensuche. Mit dem Begriff der oben eingeführten Null-Fensterung als Wurzel der Methodenbäume, gilt sogar allgemein, dass die Kinder der Fensterung für jedes Fenster einmal ausgeführt werden und die Traversierung ansonsten wie bei einer Tiefensuche durchgeführt wird. Bei der Erstellung eines solchen „Programms“ und bei Veränderungen durch Mutationen muss darauf geachtet werden, dass die folgenden Forderungen erfüllt sind:

1. Die Wurzel enthält mindestens ein Funktional.

2. Jede Fensterung mit einem Überlappungsgrad $g > 1$ enthält als letztes Kind ein Funktional.

```
<operator name="Wurzel" class="ValueSeriesPreprocessing">
  <operator name="Kette 1" class="OperatorChain">
    <operator name="ExpMA" class="ExponentialMovingAverage" />
    <operator name="Filter" class="FilterTransformation" />
    <operator name="Fensterung" class="Windowing">
      <parameter key="step_size"   value="4096"/>
      <parameter key="window_size" value="8192"/>
      <operator name="Kette 2" class="OperatorChain">
        <operator name="FFT" class="FastFourierTransform" />
        <operator name="MaxIndex" class="MaxIndexPoint" />
      </operator>
    </operator>
    <operator name="Avg" class="AverageFunction" />
  </operator>
</operator>
```

Abbildung 8.5.: Die XML-Repräsentation des Methodenbaums aus Abbildung 8.4. Die beiden Ketten, die den Ablauf des Programms steuern sind genauso gut zu erkennen wie die Eltern-Kinder-Hierarchie.

Beide Punkte implizieren, dass leere Bäume und Fensterungen verboten sind, da stets mindestens ein Funktional enthalten ist. Anstelle der Funktionale können auch beliebige Methoden gefordert werden, dies hat jedoch drastische Konsequenzen auf den Speicherverbrauch und die Laufzeit der Verfahren. Die Forderung nach einem Funktional kann daher nur bei überlappungsfreien Fensterungen entfallen und durch die Forderung nach einem Filter ersetzt werden. Dieses Vorgehen ist als stückweise Filterung aus Kapitel 6 bereits bekannt.

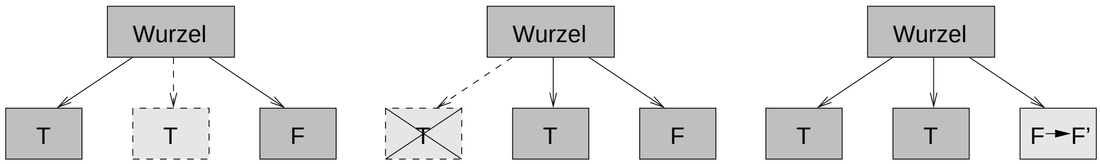
8.4.1. Methoden-Programmiersprache

Die Programmiersprache definiert die Repräsentation der Programme. Es wurde bereits festgelegt, dass die Programme durch Methodenbäume gegeben sind. Eine Sprache, die sich sehr gut zur Repräsentation von Bäumen eignet, ist XML (*eXtensible Markup Language*). Sie ist auch die Grundlage der in Kapitel 9 vorgestellten Umgebung YALE, mit der alle Experimente durchgeführt werden. Abbildung 8.5 zeigt, wie der in Abbildung 8.4 vorgestellte Methodenbaum durch XML beschrieben wird. YALE kümmert sich um den korrekten Datenfluss und darum, dass ein korrekter Aufbau der Vorverarbeitungsketten gewährleistet ist.

Die Wurzel, hier eine Methode vom Typ *ValueSeriesPreprocessing* entspricht einer Null-Fensterung. Die inneren Operatoren sind die Kinder der Wurzel, umrandet von *Kette 1*. Die Fensterung stellt einen inneren Knoten dar, deren Kinder ebenfalls von einer Kette umschlossen sind. Sie definiert den Startpunkt für einen neuen Methodenbaum. Die Parameter wie Schrittweite und Fensterbreite der Fensterung können einfach durch definierte Schlüsselwörter angegeben werden. Die XML-Beschreibung des Programms zur Merkmalsextraktion entspricht damit genau dem abgebildeten Methodenbaum.

8.4.2. Mutationen

Unter Mutationen versteht man alle Operationen, die aus einem Suchpunkt oder Individuum ein anderes ableiten. Dass die verwendeten Mutationen ein Individuum nicht zu stark verändern



(a) Generierung (b) Löschen (c) Wechsel

Abbildung 8.6.: Die linke Abbildung demonstriert das Einfügen einer weiteren Transformation vor dem Endfunktional. In der mittleren Abbildung wird eine Transformation gelöscht. Es ist darauf zu achten, dass kein zwingend benötigtes Endfunktional gelöscht wird. Die letzte Abbildung schließlich zeigt den Wechsel eines Funktionals F zu einem Funktional F'.

sollten bzw. dass die Wahrscheinlichkeit für starke Veränderungen kleiner ist als die für kleine Veränderungen, wurde bereits gefordert. Es werden hauptsächlich drei Mutationen verwendet. Zwei davon ändern die Struktur des Baums, sie bauen neue Methoden an passenden Stellen in den Baum ein oder löschen Methoden an Stellen, an denen dies erlaubt ist. Die dritte Mutation ändert eine Methode innerhalb der Klasse der Methode.

Generierungs-Mutation: Diese Mutation erzeugt zufällig eine neue Methode und fügt sie an einer passenden Stelle ein. Diese wird ebenfalls zufällig ermittelt. Dabei werden die folgenden Bedingungen beachtet:

1. Transformationen werden vor dem Endfunktional eingefügt.
2. Ketten müssen genau ein Funktional enthalten.

Die zweite Bedingung kann für die Kinder einer Null-Fensterung abgeschwächt werden. Der Methodenbaum liefert auf diese Weise mehr als ein Merkmal gleichzeitig, zum Beispiel Durchschnitt, Varianz und Maximum. Überlappungsfreie Fensterungen können ebenfalls auf ein Funktional verzichten. Dies entspricht der stückweisen Filterung aus Abschnitt 6.4.

Löschungs-Mutation: Diese Mutation löscht eine zufällig ausgewählte Methode aus dem Methodenbaum. Dabei wird sichergestellt, dass nur zulässige Methoden gelöscht werden. Alle Fensterungen mit Überlappungsgrad $g > 1$ müssen ihr Endfunktional behalten, insbesondere auch die Null-Fensterung an der Wurzel.

Wechsel-Mutation: Die Wechsel-Mutation nimmt keine strukturelle Änderung an dem Individuum vor. Sie wählt eine Methode zufällig aus und tauscht sie gegen eine andere Methode der gleichen Klasse aus. Sollte die neue Methode eine Fensterung mit Überlappungsgrad $g > 1$ sein, so enthält diese zwingend ein Endfunktional.

Unter den Bedingungen der Generierungs-Mutation wird auch eine zufällige Startpopulation einer gewünschten Größe erzeugt. Abbildung 8.6 zeigt schematisch die Arbeitsweise der drei Mutationen.

Es sind zwei weitere Mutationen möglich, die den Raum der Methoden jedoch drastisch vergrößern können. Viele Methoden der Wertereihenanalyse besitzen Parameter. Bei einer Fensterung sind dies z. B. die Schrittweite und Fensterbreite, bei einer Zustandsraumrekonstruktion die Anzahl der Dimensionen und die Verzögerung. Eine weitere Mutation, die *Parameter-Mutation*, ändert die Parameter je nach Typ des Parameters mit unterschiedlichen Strategien. Bei Zahlen addiert die Mutation ähnlich einer Evolutionsstrategie eine normalverteilte Größe und binäre

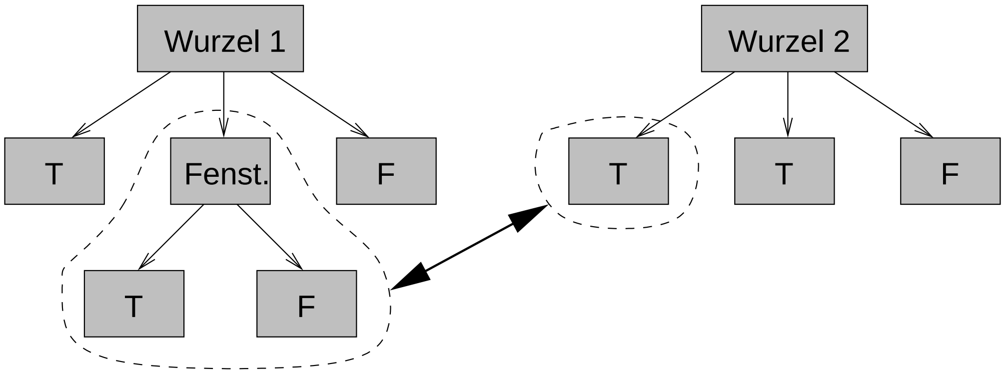


Abbildung 8.7.: Crossover wird realisiert durch Vertauschen von gleichartigen Methoden der beiden gewählten Individuen. Ist eine der Methoden eine Fensterung, so wird der gesamte zugehörige Teilbaum verschoben.

Parameter bzw. kategorische Parameter werden geflippt bzw. zufällig ausgewählt. Durch die Vergrößerung des Suchraums erhöht sich auch die erwartete Optimierungszeit entsprechend.

Die zweite optionale Mutation ist nützlich, wenn mehr als ein Merkmal extrahiert werden soll. Bei den bisher beschriebenen Methodenbäumen liefert lediglich das Endfunktional der Wurzel das extrahierte Merkmal. Eine *Verzweigungsmutation* kann in den Methodenbaum einen Knoten einbauen, an dem sich die weitere Vorverarbeitung verzweigt. Die zu diesem Zeitpunkt vorliegende Reihe wird kopiert und getrennt in verschiedenen Bäumen verarbeitet, für jeden Ast der Verzweigung in einem neuen Methodenbaum. Dies erhöht den Speicheraufwand und kann daher nur auf kleinen Wertereihen angewendet werden. Die Vergrößerung des Suchraums fällt dagegen nicht so stark ins Gewicht. Sie entspricht lediglich der Vergrößerung, die auch beim Hinzufügen einer weiteren Transformation entstehen würde.

8.4.3. Crossover

Die Kreuzung zweier Individuen ist nun denkbar einfach. Nach der Selektion der Eltern wird die Kreuzung durch das Umhängen von passenden Teilbäumen realisiert. Es wird zufällig eine Methode des ersten Elter ausgewählt und geprüft, ob der zweite Elter ebenfalls eine Methode dieses Typs besitzt. Falls ja, so werden die Methoden der beiden Eltern ausgetauscht. Handelt es sich bei den zufällig gewählten Methoden um die Wurzeln von Teilbäumen, also um Fensterungen oder ein Kombinations-Funktional, so werden die vollständigen Teilbäume ausgetauscht. Abbildung 8.7 demonstriert das Umhängen der Teilbäume.

8.4.4. Selektion

Die *Selektion* dient zum einen dazu, Eltern für Mutationen und Crossover zu wählen, andererseits dazu, Individuen für die nächste Generation zu wählen. Es gibt eine ganze Reihe von Selektionsverfahren, die alle gemeinsam haben, dass Individuen mit einer höheren Fitness bevorzugt selektiert werden. In dieser Arbeit wird ein Selektionsverfahren namens *Roulette Wheel* benutzt. Jedes Individuum erhält von den 360° eines Roulette Rades den Anteil, der der Fitness des Individuums entspricht. Je fitter das Individuum, desto größer ist der Anteil an der Scheibe und desto wahrscheinlicher ist es, dass das Individuum durch „Drehen“ des Rades selektiert wird. Optional ist es möglich, dass Individuum mit der höchsten Fitness in jedem Fall zu wählen (*elitist selection*).

8.4.5. Fitness

Wie bereits erwähnt wurde, wird bei der genetischen Programmierung jedes Individuum danach bewertet, wie gut es die vorliegende Aufgabe löst. Die Aufgabe ist die Extraktion von Merkmalen, auf deren Basis Lernverfahren eine gute Performanz liefern. Für die Bewertung einer von einem Lernverfahren gelernten Hypothese gibt es vielfältige Möglichkeiten. Wenn es eine unabhängige Menge von Testdaten gibt, so kann das gelernte Modell auf diesen angewendet werden und so die erreichte Performanz bestimmt werden. Da Daten aber generell knapp sind, kann man auch einen Teil der Daten zurückhalten, auf den übriggebliebenen Daten lernen und die Performanz auf den zurückgehaltenen Daten messen. Wiederholt man dieses Vorgehen für k zufällig zurückgehaltene Teilmengen und mittelt das Ergebnis über alle Teilmengen, so nennt man diese Performanzmessung *k-fache Kreuzvalidierung*.

Eine Kreuzvalidierung liefert also eine recht gute Abschätzung, wie gut sich die gelernte Hypothese auf unbekannten Daten verhält. Je nach gewählten Performanzmaß, für Klassifikationen sind üblich *Accuracy*, *Precision* oder *Recall* kann eine Berechnung der Fitness vorgenommen werden. Ein Merkmalsvektor, und damit auch der zugehörige Methodenbaum, ist um so fitter, je besser das jeweilige Performanzmaß ist. Der Nachteil der Kreuzvalidierung liegt in der großen Laufzeit. Praktisch fällt diese jedoch kaum ins Gewicht, da die Extraktion der Merkmale zumindest bei umfangreichen Wertereihen deutlich länger dauert als das k-fache Lernen auf den im Vergleich kurzen Merkmalsvektoren.

8.5. Diskussion und Aufwandsabschätzungen

Evolutionäre Algorithmen sind eine faszinierende Möglichkeit, Probleme zu lösen. Nicht zuletzt deshalb ist es in den vergangenen Jahrzehnten zu Missverständnissen gekommen, zu welcher Leistung diese Verfahren fähig sind. Ihr Vorteil liegt unumstritten darin, dass sie auf eine große Klasse von Problemen – den Optimierungsproblemen – schnell anzuwenden sind und in vielen Fällen ausreichend gute Ergebnisse liefern. Allerdings liefern auf das Problem spezialisierte Algorithmen häufig bessere Resultate oder arbeiten in kürzerer Zeit. Es gilt daher immer: Wenn Wissen über das Problem vorhanden ist, sollte es auch eingesetzt werden. Hintergrundwissen erlaubt fast immer eine effizientere Lösung des Problems. Der Vorteil evolutionärer Algorithmen ist dann groß, wenn man kaum etwas oder gar nichts über das Problem weiß.

Die hier vorgestellte Systematik der Methoden ist eine Form von Hintergrundwissen. Sie erlaubt bestimmte Kombinationen und verbietet andere. Der Rahmen ist allgemein genug, um auch neue Methoden zu integrieren. Wenn noch mehr Wissen über die konkrete Aufgabe vorhanden ist, so erlaubt dieser Rahmen eine Zusammenstellung der Extraktionsmethoden von Menschenhand und unterstützt einen Benutzer bei der Auswahl geeigneter Methoden. Es wäre unsinnig, das Wissen über Musik nicht einzusetzen und z. B. darauf zu verzichten, bereits bewährte Merkmale aus dem Frequenzspektrum zu generieren. Besitzt der Anwender keine weitere Information über die vorliegenden Daten, so besteht immer noch die Möglichkeit, gute Methodenbäume zur Merkmalsextraktion automatisch zu entwickeln. Dies trifft auch zu, wenn der Benutzer kein ausreichendes Interesse oder Zeit hat, eine spezielle Lösung für die vorliegenden Daten zu entwickeln.

Man erkauft sich diesen Verzicht auf das eigene Denken allerdings mit Laufzeit. Genetische Programmierung braucht seine Zeit, da über eine große Anzahl von Generationen der Suchraum nur langsam durchschritten wird, bis das Optimum innerhalb des Suchraums gefunden werden kann. Zunächst betrachten wir den Aufwand eines Methodenbaums auf einer Wertereihe der Länge n. Da die Fensterungen dabei ineinandergeschachtelt werden können, macht es nun keinen Sinn mehr, für alle Fensterungen die gleiche Fensterbreite w zu verwenden. Nach der ersten Fensterung hat die Reihe, die von den Kindern der Fensterung bearbeitet wird, nur noch Größe w und eine weitere Fensterung ist nicht mehr möglich. Stattdessen führen wir jetzt basierend auf der allgemeinen Fensterung aus Kapitel 6 das Konzept der *dynamischen Fensterung* ein. Für einen gegebenen Überlappungsgrad g sowie der vorliegenden Reihe der Länge n werden Fensterbreite und Schrittweite angepasst.

Definition 8.8 (Dynamische Fensterung) *Sei $(x_i)_{i\in\{1,\ldots,n\}}$ die zu bearbeitende Wertereihe der Länge n und $d \in \{2,\ldots,\frac{n}{2}\}$. Eine Fensterung mit Überlappungsgrad g, Fensterbreite $w = \frac{n}{d}$ und Schrittweite $s = \frac{n}{gd}$ heißt* DYNAMISCHE FENSTERUNG.

Im Gegensatz zu der Fensterung mit starren Fensterbreiten kann eine dynamische Fensterung in der Praxis gut eingesetzt werden. Das folgende Lemma liefert die maximale Tiefe eines Methodenbaumes mit dynamischen Fensterungen:

Lemma 8.1 *Sei eine Wertereihe $(x_i)_{i\in\{1,\ldots,n\}}$ der Länge n gegeben. Ein Methodenbaum mit dynamischen Fensterungen besitzt eine maximale Tiefe von $\log_d n - 1$.*

Beweis: Eine dynamische Fensterung unterteilt die Wertereihe in Fenster der Breite $\frac{n}{d}$. Diese Fensterbreite hängt sowohl von der Länge der Wertereihe ab, auf die die Fensterung angewendet werden soll als auch von dem Parameter d. Besitzt diese Fensterung als Kind eine weitere Fensterung, so stehen dieser nur noch $\frac{n}{d}$ Werte zur Verfügung. Auf diesen Werten können erneut Fenster gebildet werden, jedes Fenster hat die Länge $\frac{n}{d^2}$. Dieses Vorgehen kann maximal $\log_d n - 1$ mal wiederholt werden, bis die Fenster die Länge

$$\frac{n}{d^{\log_d n - 1}} = \frac{n}{\frac{d^{\log_d n}}{d}} = \frac{n \cdot d}{n} = d$$

erreichen. Eine weitere Fensterung würde die Länge der Eingabe für die Kinder auf 1 reduzieren.

8.5.1. Methodenbäume mit dynamischen Fensterungen

Nachdem die maximale Tiefe eines Methodenbaumes berechnet werden kann, wird nun die Laufzeit der Methodenbäume mit dynamischen Fensterungen auf einer Reihe der Länge n betrachtet. Die Laufzeit der inneren Methoden auf einer Reihe der Länge n ist durch die Funktion $L(n)$ gegeben. Für eine dynamische Fensterung ergibt sich der gesamte Aufwand nach Lemma 6.1 durch Multiplikation der Anzahl der Fenster mit dem Aufwand pro Fenster:

$$\begin{aligned}
&\left(\frac{n}{s} - g + 1\right) \cdot L\left(\frac{n}{d}\right) \\
=&\left(\frac{ngd}{n} - g + 1\right) \cdot L\left(\frac{n}{d}\right) \\
=&(gd - g + 1) \cdot L\left(\frac{n}{d}\right) \\
=&(g(d-1) + 1) \cdot L\left(\frac{n}{d}\right)
\end{aligned}$$

Fügen wir dieser Fensterung als Kind eine weitere Fensterung hinzu, so ersetzen wir in dieser Gleichung $L(\frac{n}{d})$ durch

$$(g(d-1)+1) \cdot L\left(\frac{n}{d^2}\right)$$

und erhalten

$$(g(d-1)+1)^2 \cdot L\left(\frac{n}{d^2}\right)$$

Wir iterieren dieses Vorgehen und erhalten bei der i-ten Ebene des Methodenbaums den Aufwand

$$(g(d-1)+1)^i \cdot L\left(\frac{n}{d^i}\right)$$

Nach Lemma 8.1 ist die Tiefe des Methodenbaums durch $\log_d n - 1$ beschränkt. Diese maximale Tiefe setzen wir für i ein und erhalten schließlich:

$$\begin{aligned}
& (g(d-1)+1)^{\log_d n-1} \cdot L\left(\frac{n}{d^{\log_d n-1}}\right) \\
= \quad & (g(d-1)+1)^{\log_d n-1} \cdot L\left(\frac{n \cdot d}{n}\right) \\
= \quad & (g(d-1)+1)^{\log_d n-1} \cdot L(d) \qquad (8.3)
\end{aligned}$$

Als nächstes betrachten wir exemplarisch die Laufzeiten der Fensterungen, die wir auch schon in Abschnitt 6.1 genauer betrachtet haben. Wir benutzen die realistischen Überlappungsgrade $g = 1$ sowie $g = 2$ und berechnen den Aufwand für $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^p)$ und $O(a^n)$ als mögliche Laufzeitfunktionen L der inneren Methoden:

$g = 1$: Diese Fensterung, bei der die Fenster direkt aneinanderstoßen, bildet nach Gleichung 8.3 Methodenbäume mit einer Laufzeit von

$$d^{\log_d n-1} \cdot L(d) = \frac{n}{d} \cdot L(d)$$

Besitzen alle inneren Methoden Laufzeit $O(n)$, so ist auch L linear und der Gesamtaufwand ist $\frac{n \cdot d}{d} = n$. Unabhängig von der Wahl des Parameters d besitzt ein Methodenbaum aus linear arbeitenden Methoden und dynamischen Fensterungen mit $g = 1$ also auch insgesamt eine lineare Laufzeit. Für innere Methoden mit Laufzeit $O(n \log n)$ beträgt der Gesamtaufwand des Methodenbaums $\frac{n \cdot d}{d} \log d = n \log d$. Bei quadratischem Aufwand der inneren Methoden $n \cdot d$, analog dazu $n \cdot d^{p-1}$ für innere Methoden mit Aufwand $O(n^p)$ und schließlich ein Gesamtaufwand von $\frac{n}{d} a^d$ für einen Methodenbaum, dessen innere Methoden einen exponentiellen Aufwand von $O(a^n)$ besitzen. Da $d \leq \frac{n}{2}$ gilt, ist ein Methodenbaum aus dynamischen Fensterungen mit $g = 1$ nie teurer als die Anwendung der Methoden auf der gesamten Reihe.

$g = 2$: Bei dieser realistischen Fensterung überlappen sich die Fenster halb. Ein Methodenbaum mit solchen Fensterungen besitzt nach Gleichung 8.3 eine Laufzeit von

$$(2d-1)^{\log_d n-1} \cdot L(d) = \frac{1}{2d-1}(2d-1)^{\log_d n} \cdot L(d)$$

Für $d = 2$ z. B. ist die Laufzeit gegeben durch

$$\frac{1}{3} \cdot 3^{\log_2 n} \cdot L(d) = \frac{1}{3} \cdot 3^{\frac{\log_3 n}{\log_3 2}} \cdot L(d) = \frac{1}{3} \cdot \left(3^{\log_3 n}\right)^{\frac{1}{\log_3 2}} \cdot L(d) = \frac{1}{3} \cdot n^{\frac{1}{\log_3 2}} \cdot L(d) \approx \frac{1}{3} \cdot n^{1{,}585} \cdot L(d)$$

Auch für Methodenbäume mit diesen Fensterungen betrachten wir den Gesamtaufwand, exemplarisch für $d = 2$. Besitzen alle inneren Methoden Laufzeit $O(n)$, so ist auch L

linear und der Gesamtaufwand ist $\frac{2}{3} \cdot n^{1,585}$. Wir sehen, dass unabhängig von der Wahl des Parameters d ein Methodenbaum aus linear arbeitenden Methoden und dynamischen Fensterungen mit $g = 2$ insgesamt keine lineare Laufzeit mehr besitzt. Für innere Methoden mit Laufzeit $O(n \log n)$ beträgt der Gesamtaufwand des Methodenbaums ebenfalls $\frac{1}{3} \cdot n^{1,585} \cdot 2 \log 2 = \frac{2}{3} \cdot n^{1,585}$. Bei quadratischem Aufwand der inneren Methoden ergibt sich $\frac{4}{3} \cdot n^{1,585}$, analog dazu $\frac{2^p}{3} \cdot n^{1,585}$ für innere Methoden mit Aufwand $O(n^p)$ und schließlich ein Gesamtaufwand von $\frac{a^d}{3} \cdot n^{1,585}$ für einen Methodenbaum, dessen innere Methoden einen exponentiellen Aufwand $O(a^n)$ besitzen.

Für dynamische Fensterungen mit einem realistischen Überlappungsgrad von $g = 2$ kann also garantiert werden, dass der Aufwand eines Methodenbaums bestehend aus nicht exponentiell arbeitenden Methoden auch insgesamt nicht exponentiell wird. Da alle Methoden in dieser Arbeit im schlimmsten Fall Laufzeit $O(n^2)$ besitzen, betrachten wir zum Abschluss noch einmal allgemein die Laufzeit im schlimmsten Fall für einen Methodenbaum in Abhängigkeit von d und g für eine Laufzeit der Methoden von $O(n^2)$. Wir starten mit Gleichung 8.3 und setzen für $L(d)$ die quadratische Laufzeit d^2 ein:

$$(g(d-1)+1)^{\log_d n - 1} \cdot d^2 = \frac{d^2}{g(d-1)+1}(g(d-1)+1)^{\log_d n}$$

Nach Anwendung der Logarithmus- und Potenzgesetze erhalten wir

$$\begin{aligned}
\frac{d^2}{g(d-1)+1}(g(d-1)+1)^{\log_d n} &= \frac{d^2}{g(d-1)+1}(g(d-1)+1)^{\frac{\log_{g(d-1)+1} n}{\log_{g(d-1)+1} d}} \\
&= \frac{d^2}{g(d-1)+1}\left((g(d-1)+1)^{\log_{g(d-1)+1} n}\right)^{\frac{1}{\log_{g(d-1)+1} d}} \\
&= \frac{d^2}{g(d-1)+1} \cdot n^{\frac{1}{\log_{g(d-1)+1} d}} \\
&= \frac{d^2}{g(d-1)+1} \cdot n^{\log_d g(d-1)+1}
\end{aligned}$$

Das Einsetzen der Werte $g = 2$ und $d = 2$ liefert das Ergebnis der beispielhaften Analyse von oben. In dieser allgemeinen Form ist deutlich zu sehen, dass für die in dieser Arbeit vorgestellten Methoden unter keiner Bedingung eine exponentielle Laufzeit der Methodenbäume in der Länge der Reihe entstehen kann. Eine Bestimmung des maximalen Wertes für d bei gegebener Länge n und Überlappungsgrad g sollte einer automatisierten Erstellung von Methodenbäumen vorangehen. Dies kann einfach durch Umstellung der letzten Gleichung erreicht werden.

Es bleibt erneut festzustellen, dass auch bei einem realistischen Überlappungsgrad von $g = 2$ eine Laufzeitersparnis durch Fensterungen in Methodenbäumen auftreten kann. Genau wie bei der Laufzeitanalyse der allgemeinen Fensterung tritt der Effekt um so stärker auf, je teurer die Anwendung der Methode auf der gesamten Reihe ist.

8.5.2. Starke Überlappung in nicht-dynamischen Fensterungen

Im vergangenen Abschnitt wurde die Einführung der dynamischen Fensterung damit motiviert, dass eine vernünftige Schachtelung sonst nicht machbar ist. Dabei wurde der Überlappungsgrad festgehalten und sowohl die Fensterbreite als auch die Schrittweite durch einen Faktor angepasst. Nun wird gezeigt, dass diese Definition der dynamischen Fensterung sinnvoll war.

Die Methodenbäume in diesem Abschnitt benutzen keine dynamischen Fensterbreiten, sondern legen die Schrittweite s fest und benutzen für die Fensterung einen vorgegebenen Teil der Werte. Sei $s = 1$ und $w = g = n - 1$ für jede dieser Fensterungen. Selbst wenn ansonsten nur Methoden mit linearer Laufzeit zugelassen werden, können mit diesen Fensterungen Methodenbäume mit exponentieller Laufzeit konstruiert werden.

Jede Fensterung erzeugt zwei Fenster mit einer Breite $n - 1$. Es bleiben also für die Kinder in jedem Fenster $n - 1$ Werte zur Bearbeitung. Es können erneut Fensterungen mit den gleichen Parametern als Kinder benutzt werden, die dann wiederum zwei Fenster erzeugen; diesmal mit der Länge $n - 2$. Dieses Vorgehen kann iteriert werden:

$$\begin{gathered} 2 \cdot O\left(\tfrac{n}{2}\right) \\ 4 \cdot O\left(\tfrac{n}{4}\right) \\ 8 \cdot O\left(\tfrac{n}{8}\right) \\ \vdots \\ 2^i \cdot O\left(\tfrac{n}{2^i}\right) \end{gathered}$$

Nach $n - 2$ Ebenen im Methodenbaum bleiben nur noch zwei Werte in jedem Fenster übrig und eine weitere Fensterung ist nicht möglich. Damit ist der Aufwand insgesamt:

$$2^{n-1} \cdot 2 = 2^n$$

Selbst für lineare innere Methoden ergibt sich für diese Art der Fensterung eine exponentielle Laufzeit in der Länge der Reihen. Die Wahl der Definition der dynamischen Fensterung hat sich also als sinnvoll erwiesen.

8.5.3. Erwartete Zahl von Generationen

Der Gesamtaufwand für die automatisierte Merkmalsextraktion mittels genetischer Programmierung berechnet sich als das Produkt aus dem Aufwand pro Generation und der Zahl der benötigten Generationen. Durch die Kenntnis der Laufzeit, die jeder Methodenbaum besitzt kann der Aufwand für eine Generation berechnen werden. Um eine Aussage über die Laufzeit der gesamten Automatisierung machen zu können, muss die zentrale Frage der theoretischen Analyse von evolutionären Algorithmen beantwortet werden: Wie viele Generation braucht das Verfahren, bis es das Optimum gefunden hat? Wie groß ist die Wahrscheinlichkeit, dass es in einem lokalen Optimum stecken bleibt? Dazu wäre es nötig, die Wahrscheinlichkeit zu kennen, in ein lokales Optimum zu geraten und auch wieder heraus. Die Beantwortung dieser Fragen gelingt kaum bei leichteren Problemen und einfacheren evolutionären Algorithmen wie den $(1 + 1)$EA und ist Bestandteil aktueller Forschung[11]. Eine vollständige Analyse der erwarteten Optimierungszeit der Automatisierung der Merkmalsextraktion aus Wertereihen steht also noch aus. Gerade Kreuzungen machen die Analyse schwer, da sie größere Abhängigkeiten der Individuen erzeugen als Mutationen. Auch die generierende und löschende Mutation und die ständige Veränderung der Größe der Suchpunkte machen eine Analyse nahezu unmöglich. Eine Abschätzung, nach wie vielen Generationen das Verfahren das Optimum findet, ist bei einem komplexen Verfahren wie der vorgestellten Automatisierung durch genetische Programmierung daher noch nicht zu leisten.

Dies hat weitreichende Konsequenzen für die Anwendbarkeit des Verfahrens. Für die Merkmalsextraktion mittels eines Methodenbaumes konnte nachgewiesen werden, dass diese effizient arbeitet. Damit gelingt eine praktische Umsetzung der Extraktion selbst in Echtzeit. Für Lernaufgaben wie die Klassifikation nach Genre kann eine automatisierte Merkmalsextraktion einmalig vor der Anwendung durchgeführt werden und zur Laufzeit werden lediglich die festgelegten Merkmale extrahiert. Für eine Klassifikation der Benutzerpräferenz gelingt dieses Vorgehen jedoch nicht. Für Einzelklassifikationen dieser Art, die vom Benutzer abhängen, müsste eine Extraktion der Merkmale für jeden Benutzer einzeln durchgeführt werden. Da eine polynomielle Anzahl von Generationen nicht garantiert werden kann, ist eine automatisierte Echtzeit-Extraktion der Merkmale kaum möglich. Verschärft wird das Problem dadurch, dass sich Benutzerpräferenzen mit der Zeit oder mit unterschiedlichen Stimmungen ändern können (*concept drift*) und eine ständige Anpassung nötig wäre.

KAPITEL 9

Experimente

Die vorgestellte Systematik der Analysemethoden erlaubte die Entwicklung eines Verfahrens zur automatischen Merkmalsextraktion aus Wertereihen. Die dabei verwendeten Methoden wurden in den letzten Kapiteln beschrieben. Dabei konnte für alle Methoden eine effiziente Laufzeit nachgewiesen werden. Außerdem konnte gezeigt werden, dass die Methodenbäume zur Merkmalsextraktion ebenfalls mit polynomieller Laufzeit arbeiten, wenn alle benutzten Methoden eine polynomielle Laufzeit aufweisen. Im Folgenden werden Experimente beschrieben, bei denen aus umfangreichen Zeitreihen (Audiodaten) Merkmale automatisch extrahiert und je nach Klassifikationsaufgabe die besten selektiert wurden.

9.1. Die Experimentierumgebung Yale

Die Implementierung der dargestellten Methoden erfolgte in einem generischen Framework zur Behandlung von Wertereihen wie in [39] gefordert. Als Basis wurde die Entwicklungsumgebung YALE [12, 33] benutzt, mit der auch alle Experimente durchgeführt wurden. Die Methoden zur Wertereihenanalyse sind als Plugin verfügbar. Installationshinweise finden sich im Anhang C, Details zur Implementierung und die wesentlichen Schnittstellen werden im Anhang A beschrieben.

YALE wurde aus zwei Gründen entwickelt: Zum einen sollten typische Anwendungen des Maschinellen Lernens schnell durchführbar sein, zum anderen sollten neue Methoden leicht entwickelt und integriert werden können. Damit ist eine Anpassung an das vorliegende Problem in kurzer Zeit möglich. Probleme im Bereich des maschinellen Lernens umfassen komplexe Vorverarbeitungs- und Lernschritte. Die Arbeitsweise einer Entwicklungsumgebung sollte daher durch komplexe, geschachtelte Operatorketten definierbar sein. Sie muss eine transparente Datenverwaltung aufweisen und komfortable Parameterjustierungen erlauben. Außerdem muss sie flexibel, erweiterbar und darüberhinaus leicht zu bedienen sein. YALE erfüllt alle Bedingungen und kann sowohl mittels einer intuitiven graphischen Oberfläche als auch im Kommandozeilenmodus benutzt werden.

Die Operatoren in YALE definieren sich durch ihre Ein- und Ausgabe sowie durch ihre Funktion und können baumartig ineinandergeschachtelt werden. Damit entsprechen YALE Operatoren direkt den in dieser Arbeit vorgestellten Methoden. Zusätzlich stehen zahlreiche Operatoren zum Lernen, zur Performanzevaluierung oder Vorverarbeitungsschritte wie Merkmalsselektion bereit.

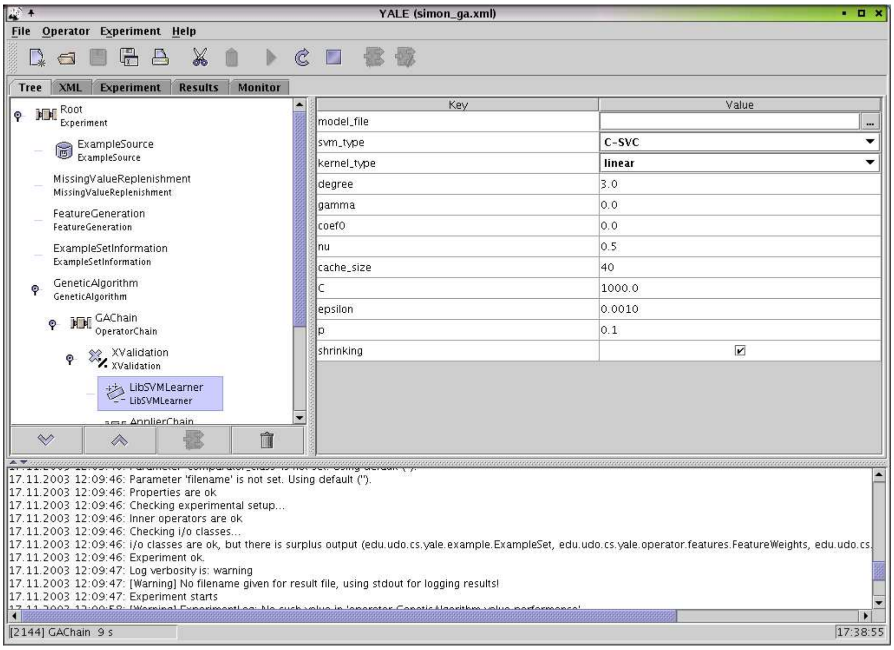


Abbildung 9.1.: Die Experimentierumgebung YALE ist leicht zu bedienen und ebenso leicht zu erweitern. Durch das Konzept der Operatorbäume ist die Integration der in dieser Arbeit vorgestellten Methoden einfach möglich.

Durch die Definition der Ein- und Ausgaben sowie der verpflichtenden Parameter für alle Operatoren können automatische Validitätsüberprüfungen vorgenommen werden. Abbildung 9.1 zeigt einen Operatorbaum und die Parametereinstellung in YALE.

9.1.1. Zusätzliche Operatoren zur Merkmalsextraktion aus Wertereihen

Alle in dieser Arbeit vorgestellten Methoden sind als Operatoren in YALE verfügbar. Transformationen und Funktionale bilden jeweils die Gruppen „Transformations“ und „Functions“. Bei den Transformationen gibt es neben den Untergruppen „Basis“, „Filter“ und „Markup“ auch den Operator für die allgemeine Fensterung, welcher mit entsprechender Parametereinstellung auch als dynamische Fensterung eingesetzt werden kann. Eine ausführliche englischsprachige Beschreibung aller Operatoren findet sich in [32].

Im Folgenden sind einige Operatoren beschrieben, die in Kombination mit den dargestellten Methoden zur Merkmalsextraktion aus Wertereihen benutzt werden können:

Branching: Operatorkette, die die weitere Vorverarbeitung der Reihe aufteilt. Diese fertigt Kopien der aktuellen Reihe an und führt die Merkmalsextraktion der Kinder auf je einer Kopie aus. Das Ergebnis ist die Ergebnisreihe des letzten Kindes und die Vereinigung aller extrahierten Merkmale.

Value Series Preprocessing: Operatorkette, die als Eingabe eine Menge von Wertereihen erwartet. Diese führt auf jeder Wertereihe die Merkmalsextraktion der Kinder durch und liefert die Menge der generierten Merkmalsvektoren als Ergebnis. Der Operator kann nur angewendet werden, wenn alle betrachteten Reihen gleichzeitig in den Hauptspeicher passen und dient als einfaches Bindeglied zu den üblichen Beispielmengen von YALE.

Music Value Series Preprocessing: Wie *Value Series Preprocessing*. Jedoch wird zu jedem Zeitpunkt nur ein Musikstück in den Speicher geladen, aus dem die Merkmale extrahiert werden. Ähnliche Operatoren müssen für andere umfangreiche Wertereihen implementiert werden.

Value Series Genetic Programming: Der in Kapitel 8 vorgestellte Ansatz zur automatischen Merkmalsextraktion aus Wertereihen. Ein innerer Operator muss ein Performanzmaß als Fitness liefern. Das Ergebnis ist der Methodenbaum (Operatorbaum), der die besten Merkmale extrahiert und die mit diesen Merkmalen erzielte Performanz.

Sinus Generator: Operator, der eine Superposition von harmonischen Schwingungen erzeugt. Amplituden und Frequenzen können als Parameter definiert werden. Auf diese Weise können schnell einfache Testwertereihen erzeugt werden.

9.1.2. Visualisierung

Ziel des Maschinelles Lernens ist, den Beispielen inhärente Informationen und Muster zu entlocken und eine Hypothese zu generieren, die die Anwendung auf neue und ungesehene Daten erlaubt. Eine genaue Kenntnis der Daten erlaubt eine sinnvolle Auswahl der Vorverarbeitungsschritte, Lernverfahren und Parameter. Daher geht dem eigentlichen Lernen häufig eine Inspektion der Daten voraus.

Die Bedeutung der Dateninspektion ist um so größer, je komplexer der betrachtete Datenraum ist. Auf diese Weise können lohnende Transformationen und Funktionen bestimmt werden und auch Nicht-Experten werden in die Lage versetzt, trennende Merkmale für Klassifikationsaufgaben zu benennen. Daher wurde ein weiterer Operator zur Visualisierung von Wertereihen eingeführt. Dieser stellt zweidimensionale Reihen in dem aktuellen Raum mit allen Auszeichnungen dar. Zusätzlich werden die bereits extrahierten Merkmale tabellarisch angezeigt. Bei multivariaten Reihen können die Wertedimensionen, die angezeigt werden sollen, aus einer Liste ausgewählt werden. In Abbildung 9.2 auf der nächsten Seite ist die Visualisierung einer Wertereihe, die mit Intervallen ausgezeichnet wurde, dargestellt. Deutlich zu sehen sind auch die Anzeige des Durchschnittwertes und die Gerade einer linearen Regression.

Alternativ ist es möglich, die momentanen Werte in ein von Gnuplot lesbares Format auszugeben. Auf diese Weise wird auch die Darstellung mehrdimensionaler Reihen erlaubt. Die Merkmalsextraktion aus umfangreichen Wertereihen wird durch eine gute Visualisierung erst ermöglicht. Für die Zukunft ist also eine Schnittstellendefinition wünschenswert, die mehreren Systemen die Visualisierung der Daten erlaubt. Sie soll sowohl die große Anzahl bereits existierender Visualisierungen als auch externe Darstellungsprogramme unterstützen. Basierend auf dieser Schnittstellendefinition können weitere Visualisierungsoperatoren entwickelt werden.

9.1.3. Merkmalsselektion

Yale besitzt eine große Zahl von Operatoren, die sich zur Merkmalsselektion einsetzen lassen. Die Selektion von einzelnen Merkmalen kann, genau wie die Extraktion neuer Merkmale oder die Generierung aus bereits vorhandenen Merkmalen die Performanz eines Lernverfahrens steigern. Hängt die zu lernende Zielfunktion z. B. von den Merkmalen a_1, a_2 und a_3 ab, nicht jedoch von a_4 und a_5, so kann der Verzicht auf die beiden letztgenannten dazu beitragen, dass das benutzte Verfahren sich nicht durch zufälliges Rauschen „verwirren“ lässt.

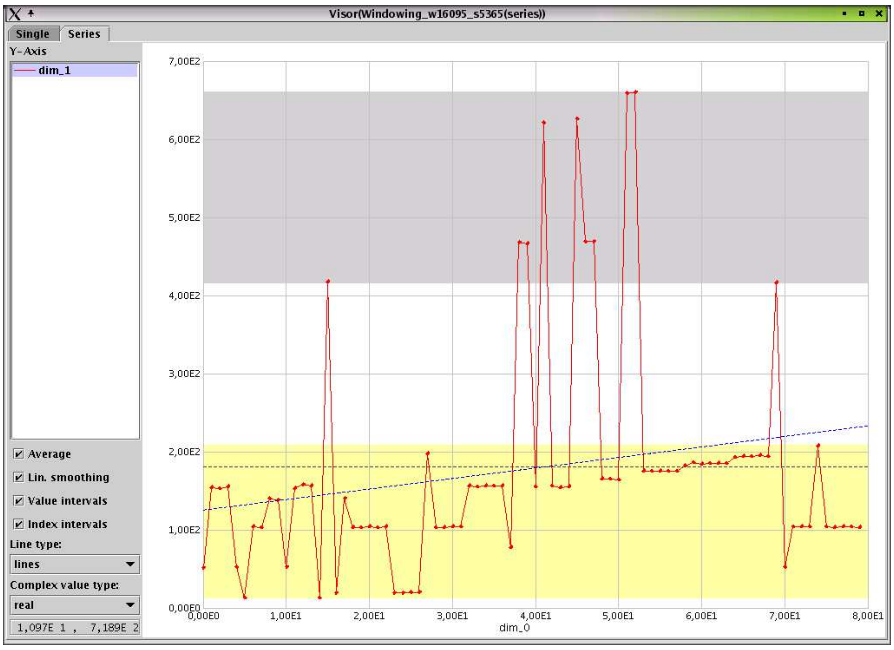


Abbildung 9.2.: Die Visualisierung von Wertereihen ist von großer Bedeutung für die Merkmalsextraktion. Die dargestellte Wertereihe wurde mit Intervallen in der Wertedimension ausgezeichnet. Die waagerechte Linie kennzeichnet den Durchschnittswert, die diagonale Gerade entspricht einer linearen Regressionsgleichung.

Alle Selektionsverfahren hängen von der gestellten Lernaufgabe ab. Die untersuchten Teilmengen des Merkmalssatzes werden anhand der Lernaufgabe evaluiert. Je höher die Performanz des Lernverfahrens ist, desto besser eignet sich die gerade betrachtete Teilmenge der Merkmale für die Lösung der Aufgabe. YALE besitzt unter anderem die folgenden Selektionsverfahren:

Forward Selection: Man evaluiert alle Merkmale einzeln und startet mit dem besten. Nun wird solange ein weiteres Merkmal hinzugewählt, bis die Performanz nicht weiter steigt.

Backward Elimination: Hier startet man mit allen Merkmalen und streicht solange jeweils eins, bis die Performanz nicht weiter steigt.

Genetischer Algorithmus: Teilmengen der Merkmale bilden die Individuen. Bei Mutationen werden Merkmale zufällig an- und ausgeschaltet; Kreuzungen werden durch Austausch der gewählten Merkmale realisiert.

Die ersten beiden Verfahren leiden darunter, nicht in die Zukunft schauen zu können. So brechen sie ab, sobald die Performanz nicht mehr steigt. Es kann aber sein, dass die Hinzunahme bzw. die Streichung von mehr als einem Merkmal die Performanz doch weiter erhöht. Parametrisierbare Varianten dieser einfachen Selektionsverfahren tragen diesem Umstand Rechnung und erlauben eine vorübergehende Fortsetzung der Selektion auch bei einer Verschlechterung der Evaluationsgüte. Auch genetische Algorithmen zur Merkmalsselektion können in einem lokalen Optimum stecken bleiben. Es gelingt ihnen jedoch mit einer gewissen Wahrscheinlichkeit, sich daraus zu befreien. Daher wurden in dieser Arbeit genetische Algorithmen mit einer hohen Generationenzahl zur Merkmalsselektion verwendet. Abbildung 9.3 auf der nächsten Seite zeigt den typischen Treppenverlauf der Performanz während einer Merkmalsselektion mit einem genetischen Algorithmus. Es war zu beobachten, dass die Verwendung von Kreuzungen keinen Gewinn brachte, so

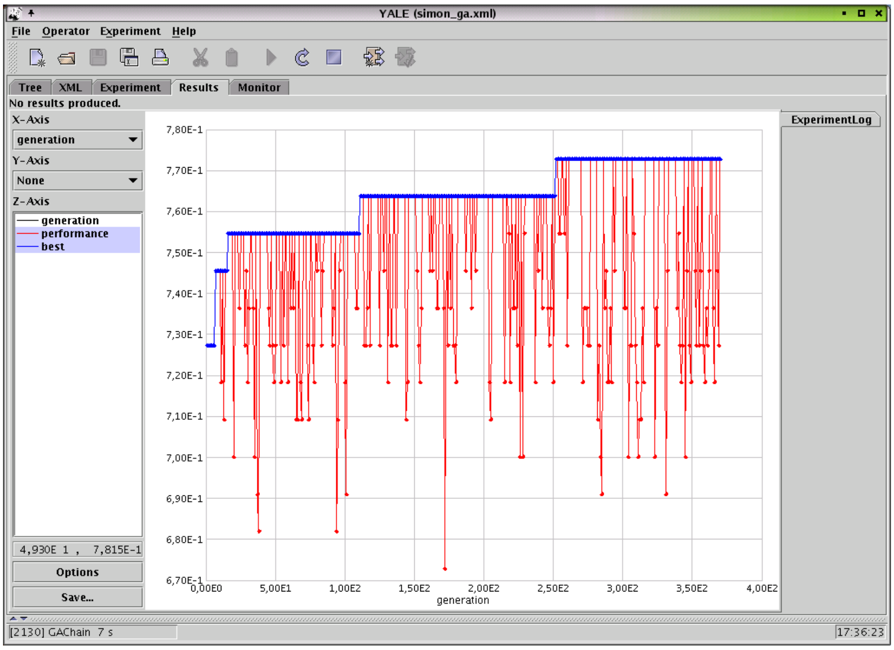


Abbildung 9.3.: YALE kann zur Laufzeit verschiedene Größen anzeigen. Die Abbildung zeigt die Performanz in Abhängigkeit von der Generation der Merkmalsselektion. Die Treppen stehen für die insgesamt beste Performanz bis zur jeweiligen Generation und die zackige Kurve für die aktuell beste Performanz.

dass auf einen schnelleren und theoretisch besser untersuchten (1+1)EA zurückgegriffen werden konnte.

Die Bedeutung der Merkmalsselektion steigt weiter, wenn Lernverfahren verwendet werden, deren Hypothese nicht direkt Rückschlüsse auf die Daten zulässt. In Kombination mit einer Support Vector Machine (SVM), die als Hypothese lediglich die Menge der Stützvektoren liefert, kann eine Merkmalsselektion immerhin die wichtigsten Merkmale zur Lösung der gegebenen Klassifikationsaufgabe nennen.

9.1.4. Merkmalsgewichtung

Bei einer Auswahl der besten Merkmale zur Bearbeitung einer konkreten Klassifikationsaufgabe werden die Merkmale entweder an- oder ausgeschaltet. Eine weitergehende Gewichtung der Merkmale kann die Performanz einfacher Lernverfahren wie k nächste Nachbaren (k-NN) erhöhen und bildet die Grundlage für Ähnlichkeitsmaße zwischen den Reihen.

Im Rahmen dieser Arbeit wurden hierzu drei Verfahren entwickelt und als Operatoren in YALE zur Verfügung gestellt. Alle drei basieren auf den weiter oben vorgestellten Verfahren zur Merkmalsselektion:

Forward Weighting: Das Vorgehen arbeitet analog zur Forward Selection. Statt neue Merkmale einfach einzuschalten, wird ihnen in diskreten Schritten immer mehr Gewicht zugeteilt.

Backward Weighting: Das Vorgehen arbeitet analog zur Backward Elimination. Statt weitere Merkmale einfach auszuschalten, wird ihnen in diskreten Schritten immer weniger Gewicht zugeteilt.

Evolutionäre Suche: Randomisiertes Verfahren zur Optimierung der Merkmalsgewichte. Ähnlich wie bei dem genetischen Algorithmus zur Selektion bestehen die Individuen aus unterschiedlichen Gewichtungsvektoren $w \in \mathbb{R}^m$ für m Merkmale. Mutationen werden durch Addition mit einer normalverteilten Größe realisiert.

Um eine sinnvolle Gewichtung zu erreichen und ein gutes Abstandsmaß im euklidischen Raum der Merkmale ermitteln zu können, sollten die Ausprägungen der Merkmale normiert werden. Eine Normierung auf das Intervall $[0, 1]$ wurde ohnehin stets vor dem eigentlichen Lernschritt vorgenommen. Dies ist besonders wichtig bei ähnlichkeitsbasierten Lernverfahren wie k-NN, da nur so eine überproportionale Berücksichtigung von Merkmalen vermieden werden kann. Andernfalls hätte die Länge der Lieder als Merkmal mit dem größten Wertebereich für alle Klassifikationsaufgaben den stärksten Einfluss.

9.2. Beatles vs. Bach: Genreklassifikation

Die Merkmalsextraktion aus umfangreichen Wertereihen wird anhand verschiedener Klassifikationsaufgaben für Audiodaten illustriert. Ist eine Kenntnis der Domäne vorhanden, kann mit Hilfe von Yale einfach ein Methodenbaum zusammengestellt werden, der die gewünschten Merkmale extrahiert. Andernfalls benutzt man das in Kapitel 8 vorgestellte Verfahren zur automatisierten Merkmalsextraktion. Das Ergebnis stellt in beiden Fällen ein oder mehrere Methodenbäume dar, die einen Satz von Merkmalen liefern. Die im folgenden benutzte Gesamtmenge von Merkmalen und ihre Extraktion wird ausführlich in [31] und in den Kapiteln 3 bis 7 beschrieben. Sie ist in Tabelle 9.2 auf Seite 104 vollständig aufgezählt.

Die Fähigkeit, mit diesem Merkmalssatz unterschiedliche Lernaufgaben lösen zu können, soll anhand verschiedener Klassifikationsaufgaben demonstriert werden. In diesem Abschnitt sollen zunächst unterschiedliche Genreklassifikationen bearbeitet werden. Dabei werden drei sich in ihrer Schwierigkeit unterscheidende Klassifikationsaufgaben gestellt. Im Einzelnen sind dies:

- KLASSIK / POP: Einfache Klassifikation der Genres Klassik und Pop-Musik. Für jede Klasse standen 100 Lieder im Ogg Vorbis Format zur Verfügung.
- TECHNO / POP: Schwierige Klassifikation der Genres Techno und Pop-Musik. Für jede Klasse standen 80 Lieder im Ogg Vorbis Format zur Verfügung.
- HIPHOP / POP: Schwierige Klassifikation der Genres Hip Hop und Pop-Musik. Für jede Klasse standen 105 Lieder im MP3 Format mit einer Kodierung von 128 kbit/s zur Verfügung.

Die Unterscheidung zwischen Klassik und Pop wird als relativ einfache Lernaufgabe angesehen. Menschlichen Hörern gelingt auch anhand sehr kurzer Ausschnitte eine sehr zuverlässige Klassifizierung. Die beiden übrigen Genreklassifizierungen weisen dagegen einen erheblich höheren Schwierigkeitsgrad auf. Der Datensatz HIPHOP / POP besitzt eine weitere Besonderheit: Obwohl die Gesamtzahl der Beispiele höher ist als bei dem zweiten Szenario, entstammen die Lieder nur einigen wenigen Alben. Für die Lernaufgabe TECHNO / POP wurde hingegen ein möglichst breiter Durchschnitt aller Spielarten der beiden Genres gewählt.

Als erstes wurden die Merkmalsvektoren aus den Datensätzen extrahiert. Mit einer genetischen Merkmalsselektion wurde die beste Teilmenge zur Lösung der jeweiligen Klassifikationsaufgabe

	KLASSIK/POP	**TECHNO/POP**	**HIPHOP/POP**
Accuracy	100%	92,89%	82,38%
Precision	100%	94,00%	84,15%
Recall	100%	92,64%	78,44%
Fehler	0%	7,11%	17,62%

Tabelle 9.1.: Die Tabelle zeigt die Performanzmaße Accuracy, Precision, Recall und den Klassifikationsfehler für die drei unterschiedlichen Klassifikationsaufgaben. Die Werte wurden bestimmt mit einer 10-fachen Kreuzvalidierung, als Lernverfahren wurde eine Support Vector Machine verwendet.

bestimmt. Für diese Merkmale wurden mit einer 10-fachen Kreuzvalidierung die Performanzmaße Accuracy, Precision und Recall abgeschätzt [25]. Der Klassifikationsfehler ist definiert als die Differenz zwischen 1 und dem Wert für Accuracy. Die Wahl des Lernverfahrens hatte keinen signifikanten Einfluss auf die Güte des Lernergebnisses. Es kamen k nächste Nachbarn (k-NN), Naive Bayes [60], der Entscheidungsbaumlerner C4.5 [43] und eine Support Vector Machine (SVM) [22, 47] zum Einsatz. Alle Verfahren lieferten vergleichbare Ergebnisse, die hier vorgestellten Ergebnisse wurden mit einer SVM mit linearer Kernelfunktion ermittelt.

Tabelle 9.1 zeigt die Werte für Accuracy, Precision und Recall für alle Klassifikationsaufgaben. Die Unterscheidung der Genres KLASSIK / POP gelingt fehlerfrei. Diese Lernaufgabe kann also als gelöst betrachtet werden. Die wesentlich schwierigeren Klassifikationsaufgaben TECHNO / POP bzw. HIPHOP / POP sind mit einer Fehlerrate von 7,11% bzw. 17,62% möglich. Das schlechtere Ergebnis für den dritten Datensatz ist auf die geringe Anzahl unterschiedlicher Beispiele für beide Klassen zurückzuführen. Die Merkmale geben hauptsächlich Klangcharakteristiken der Musikstücke wieder. Diese sind jedoch für die Stücke eines Albums insgesamt sehr ähnlich, wodurch die Merkmalsvektoren eines Albums Anhäufungen im Merkmalsraum bilden. Diese Ballungen tragen nicht im gleichen Ausmaß zur Positionierung der Hyperebene einer SVM bei wie es ein im Merkmalsraum weiter verteilter Datensatz könnte. Diese Beobachtung gilt ebenfalls für einen der Benutzerdatensätze, wie sie im nächsten Abschnitt verwendet werden.

Über die Güte der Vorhersage hinaus ist natürlich der verwendete Merkmalssatz sehr interessant. Tabelle 9.2 auf der nächsten Seite zeigt alle Merkmale und wie oft sie für die jeweilige Klassifikationsaufgabe selektiert wurden. Der optimale Merkmalssatz zur Bearbeitung der Klassifikationsaufgabe mit einer linearen SVM ist jeweils durch die fettgedruckten Zahlen angegeben. Die meisten der Merkmale des optimalen Merkmalssatzes wurden auch insgesamt besonders häufig ausgewählt. Im Fall KLASSIK / POP wurden z. B. 19 der 22 Merkmale des optimalen Satzes bei mehr als 50% der Selektionsexperimente ausgewählt. Es existiert kein Merkmal, das für alle Datensätze entfernt wurde. Es wird deutlich, dass die Merkmale einen unterschiedlichen Stellenwert für die verschiedenen Genreklassifikationen mit einer Support Vector Machine haben. Daher sollte eine Merkmalsselektion aus dem Satz aller Merkmale vor der Hypothesenbildung erfolgen, um den für die konkrete Lernaufgabe optimalen Merkmalssatz zu bestimmen. Dies gilt insbesonders für den Fall, dass der Vorgang der Merkmalsextraktion nicht automatisiert verlief und der Methodenbaum zur Merkmalsextraktion von Hand konstruiert wurde.

Es fällt auf, dass für alle Datensätze ungefähr 20 Merkmale mit dem genetischen Algorithmus selektiert wurden. Mit dieser Anzahl ist eine SVM in der Lage, die ihr gestellte Lernaufgabe möglichst gut zu lösen. Die Stützvektoren eines SVM Modells eignen sich jedoch kaum dazu, tiefere Einsichten in die Struktur der Daten zu erhalten. Um einen Vergleichswert zu bestimmen und gleichzeitig eine Einsicht zu bekommen, welche Merkmale sich besonders gut zur Trennung

Merkmal	Seite	KLASSIK / POP	TECHNO / POP	HIPHOP / POP
Länge	70	45%	**45%**	85%
Lautstärke (RMS)	66	**80%**	**50%**	75%
Lautstärke (RMS), Varianz	66	**45%**	**45%**	**70%**
Lautstärke (absolut)	67	**60%**	55%	**95%**
Tempo	23	70%	65%	**70%**
Autokorrelation, Varianz	23	45%	**95%**	**40%**
Extremwertdifferenz	73	50%	55%	45%
Extremwertdifferenz, Varianz	73	**55%**	**50%**	**50%**
Nullstellendifferenz	42	**55%**	**75%**	**45%**
Nullstellendifferenz, Varianz	42	55%	**75%**	**75%**
Winkel Zustandsraum	30	**85%**	**65%**	50%
Winkel Zustandsraum, Varianz	30	**85%**	**50%**	**65%**
Abstände Zustandsraum	31	45%	35%	**50%**
Abstände Zustandsraum, Varianz	31	**50%**	30%	**65%**
1. Peak: Frequenz	72	**50%**	40%	35%
1. Peak: Amplitude	72	**80%**	**55%**	35%
1. Peak: Breite	72	35%	**60%**	**40%**
2. Peak: Frequenz	72	**65%**	**35%**	40%
2. Peak: Amplitude	72	**60%**	**55%**	**55%**
2. Peak: Breite	72	50%	50%	45%
3. Peak: Frequenz	72	55%	50%	40%
3. Peak: Amplitude	72	55%	45%	**55%**
3. Peak: Breite	72	75%	45%	35%
4. Peak: Frequenz	72	35%	50%	**70%**
4. Peak: Amplitude	72	60%	50%	65%
4. Peak: Breite	72	60%	55%	60%
5. Peak: Frequenz	72	55%	**55%**	40%
5. Peak: Amplitude	72	**75%**	50%	**45%**
5. Peak: Breite	72	**55%**	35%	**65%**
Steigung Spektrum	75	**45%**	50%	55%
y-Achsenabschnitt Spektrum	75	**70%**	50%	**65%**
Diskrepanz Spektrum	75	**45%**	**35%**	**40%**
Quotient Maximum / Mittel	77	**65%**	**30%**	40%
Gefenstert max. Frequenz	59	**75%**	45%	35%
Gefenstert max. Frequenz, Varianz	59	45%	65%	**50%**
1. Frequenzband Start	70	55%	**65%**	**40%**
1. Frequenzband Ende	70	50%	45%	55%
1. Frequenzband Dichte	70	**75%**	50%	50%
2. Frequenzband Start	70	40%	**55%**	**45%**
2. Frequenzband Ende	70	45%	65%	50%
2. Frequenzband Dichte	70	**55%**	55%	**70%**

Tabelle 9.2.: Die Spalten der Tabelle geben die Anzahl der Selektionsexperimente an, bei denen das Merkmal ausgewählt wurde. Fettgedruckte Zahlen stehen für die Merkmale, die für den optimalen Merkmalssatz selektiert wurden. In der Spalte „Seite" ist die Seitenzahl innerhalb dieser Arbeit angegeben. Die Merkmalsgruppen entsprechen den Merkmalen im Zeitraum, im Zustandsraum und im Frequenzraum.

der verschiedenen Genres eignen, wird ein einfacher Ein-Regel-Lerner [18] auf die Daten angewendet. Dieser generiert eine einzige Regel für ein einzelnes Merkmal a_i vom Typ: „Falls $a_i < v_1$, dann ist die Klasse c_1; sonst falls $a_i < v_2$, dann ist die Klasse c_2, sonst ...“ für aufsteigende Werte v_j. Es war festzustellen, dass eine solche einfache Regel, die nur ein einziges Merkmal berücksichtigt, bereits einen großen Teil der Beispiele gut trennen kann. Für KLASSIK / POP trennt die RMS-Lautstärke bereits 187 der 200 Beispiele in die beiden Klassen. Dies entspricht einer Accuracy von 93%. Bei TECHNO / POP ist der Startwert des zweiten Frequenzbandes das beste Merkmal für eine einzelne Regel. Es teilt 122 der 160 Beispiele in die richtige Klasse ein, was einer Accuracy von 76% entspricht. Für den HIPHOP / POP Datensatz schließlich hat sich die Frequenz des dritten Peaks als bestes Merkmal erwiesen. Mit ihm lassen sich 158 der 210 Beispiele korrekt klassifizieren (75%). Die zugehörige Regel ist allerdings äußerst komplex und lässt eine starke Überspezialisierung erwarten.

Diese Ergebnisse stellen jedoch lediglich einen Vergleichswert dar, den aufwendigere Lernverfahren mit Zugriff auf mehr Merkmale in jedem Fall schlagen sollten. Um weitere Einblicke in die Qualität einzelner Merkmale zu erhalten, wurden zudem Entscheidungsbäume mit dem Verfahren C4.5 generiert und untersucht. Die Wurzel des Entscheidungsbaumes für den Datensatz KLASSIK / POP ist die absolute Lautstärke, die Wurzel für TECHNO / POP ist der Durchschnitt der Winkel im Phasenraum und die Wurzel für HIPHOP / POP schließlich ist die Länge der Lieder. Diese Merkmale wurden auch für die das Lernen mit einer SVM relativ häufig gewählt, vergleiche Tabelle 9.2 auf der vorherigen Seite.

Zumindest für zwei der drei Fälle gilt, dass einfache Merkmale die Wurzel des Entscheidungsbaumes bilden und komplexere Merkmale erst in den tieferen Ebenen der Bäume zu finden sind. Dies macht die Notwendigkeit auch weniger komplexer Merkmale deutlich, anhand derer Ausprägungen sich bereits eine grobe Trennung vollziehen lässt.

9.3. Klassifikation von Benutzerpräferenzen

Für vier unterschiedliche Benutzer sollte gelernt werden, welche Lieder der Benutzer mag und welche nicht. Die Benutzer waren in der Lage, zwischen 50 und 80 positive und ebenso viele negative Beispiele zu nennen. Das Vorgehen entspricht prinzipiell dem der Genreklassifizierung. Neben einer Abschätzung der erreichbaren Güte des Lernergebnisses sind für einen Benutzer natürlich auch die selektierten Merkmale sehr interessant. Da die selektierten Merkmale bzw. die Entscheidungsbäume außer für die jeweiligen Benutzer selbst jedoch kaum einen Erkenntnisgewinn bringen, wird an dieser Stelle auf eine umfangreiche Darstellung der Benutzermerkmale verzichtet. Die Entscheidungsbäume für die vier Benutzer unterscheiden sich in gleichem Maße wie die zugehörigen Geschmäcker. Nach einer Selektion und dem anschließenden Lernschritt kann die gelernte Hypothese dazu benutzt werden, dem Benutzer Vorschläge über ihm unbekannte Musik zu machen.

Tabelle 9.3 zeigt die Werte für Accuracy, Precision und Recall für alle vier Benutzer. Der durchschnittliche Klassifikationsfehler liegt bei $9,39\%$. Für die betrachteten Datensätze und zugrundeliegenden Verteilungen täuscht sich ein System, welches Benutzern Vorschläge unterbreiten soll, nur in ungefähr einem Zehntel aller Vorschläge. Interessant war dabei das Ergebnis des zweiten Benutzers. Sowohl bei den positiven wie auch bei den negativen Stücken waren Lieder aus dem Bereich Jazz vertreten. Das System war trotzdem in der Lage, zwischen diesen zu unterscheiden und eine gute Hypothese zu generieren. Die Benutzer eins und drei haben jeweils einen umfassenden Querschnitt durch viele Genres geliefert. Diese waren in beiden Klassen vertreten.

	BENUTZER$_1$	**BENUTZER**$_2$	**BENUTZER**$_3$	**BENUTZER**$_4$
Accuracy	$95,19\%$	$92,14\%$	$90,56\%$	$84,55\%$
Precision	$92,70\%$	$98,33\%$	$90,83\%$	$85,87\%$
Recall	$99,00\%$	$84,67\%$	$93,00\%$	$83,74\%$
Fehler	$4,81\%$	$7,86\%$	$9,44\%$	$15,45\%$

Tabelle 9.3.: Die Tabelle zeigt die Performanzmaße Accuracy, Precision, Recall und den Klassifikationsfehler für die Klassifikation der Benutzerpräferenzen aller vier Benutzer. Die Werte wurden ebenfalls bestimmt mit einer 10-fachen Kreuzvalidierung einer SVM.

Das schlechtere Ergebnis für die Musikstücke des vierten Benutzers ist darauf zurückzuführen, dass überwiegend Lieder von gleichen Alben verwendet wurden, die alle sehr ähnliche Merkmalsausprägungen besitzen. Dies deckt sich mit der Beobachtung, dass das Ziehen mehrere Beispiele aus jedem Musikstück keine weitere Verbesserung der Vorhersageergebnisse bewirkt. Es konnte bereits bei dem Datensatz HIPHOP / POP gezeigt werden, dass die ähnlichen Musikstücke eines Albums Ballungen im Merkmalsraum bilden und diese nicht zu einer Verbesserung des Lernergebnisses beitragen.

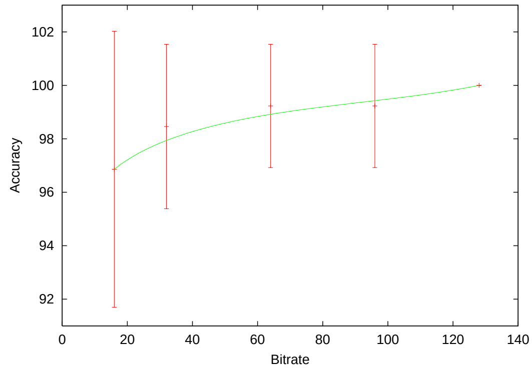
Die Erwartung war, dass das Lernen von Benutzerpräferenzen deutlich schwieriger ist als die Klassifikation nach Genre. Es hat sich jedoch gezeigt, dass die Schwierigkeit vergleichbar ist mit der Schwierigkeit der Unterscheidung von Genres wie Techno und Pop. Problematisch ist jedoch die Beschaffung negativer Beispiele, da diese selten vorliegen und häufig nicht bekannt sind.

9.4. Abhängigkeit von der Qualität der Daten

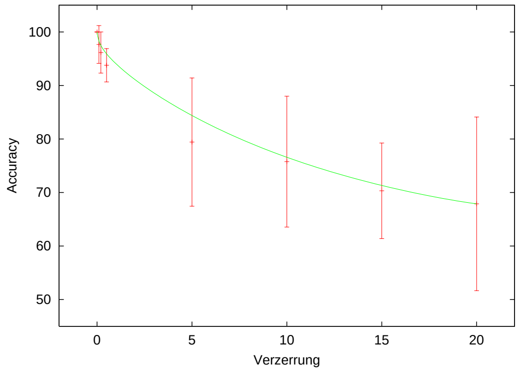
Als letztes sollte die Robustheit der Merkmale experimentell untersucht werden. Je geringer die Anfälligkeit für verschiedene Qualitäten des Ausgangsmaterials ist, desto größer ist der Nutzen für praktische Anwendungen. Es existiert eine Vielzahl von Kodierungsverfahren, die das akustische Material abhängig von einem Parameter für die Kodierungsqualität (*Bitrate*) unterschiedlich gut digitalisieren. Für die weit verbreiteten Formate MP3 und Ogg gilt, dass mit einer Bitrate jenseits von 128 kbit/s kaum ein Unterschied zur Qualität einer CD wahrgenommen werden kann. Daher besitzen die meisten Musikstücke dieser Formate auch eine entsprechende Bitrate, vereinzelt sind jedoch auch Lieder mit niedrigeren Bitraten wie 96 oder 64 kbit/s in Benutzung. Bei einer Kodierung unterhalb von 64 kbit/s ergeben sich jedoch starke Kodierartefakte, die die Qualität der Musikstücke stark beeinträchtigen.

Abbildung 9.4 (a) zeigt die Abhängigkeit der Accuracy von der verwendeten Bitrate für den Datensatz KLASSIK / POP. Für höhere Bitraten als 128 kbit/s war eine Klassifizierung fehlerfrei möglich. Für alle gängigen Bitraten unterhalb dieses Wertes ist allenfalls ein leichter Anstieg des Klassifikationsfehlers zu beobachten. Bei einer Bitrate von 16 kbit/s beträgt die Accuracy immer noch $96,88\%$. Bei dieser Qualität sind die Kodierungsartefakte bereits so stark, dass das Erkennen der Lieder durch menschliche Hörer kaum noch möglich ist. Die Merkmale weisen bezüglich der Kodierungsqualität also eine genügend hohe Robustheit auf, für alle gängigen Bitraten ist kein signifikanter Fehleranstieg zu verzeichnen.

Eine weitere Experimentreihe sollte die Anfälligkeit für Rauschen und Verzerrung überprüfen. Dazu wurden die Musikstücke mit einem normalverteilten Rauschen mit verschiedenen Standardabweichungen überlagert und dadurch stark verzerrt. Abbildung 9.4 (b) zeigt die Ergebnisse



(a) Bitrate



(b) Verzerrung

Abbildung 9.4.: Die linke Abbildung zeigt die Accuracy für den Datensatz KLASSIK / POP für unterschiedliche Bitraten bei der Kodierung. Rechts wurde die Abhängigkeit von verschieden starken Verzerrungen gemessen. Die Merkmale sind robust genug, um auch mit großen Qualitätsverminderungen umgehen zu können.

dieser Messreihe. Die Auswirkungen einer Verzerrung sind wesentlich stärker als die unterschiedlicher Kodierungsqualitäten. Viele der Merkmale basieren auf dem Frequenzspektrum der Lieder, Rauschen fügt diesem weitere Frequenzen hinzu. Trotzdem bewirkt eine Verzerrung, deren Standardabweichung der Hälfte des Maximalwertes der Lieder entspricht, lediglich eine Verschlechterung auf 93%. Bei diesem Rauschen sind die Lieder bereits so stark verzerrt, dass eine Erkennung einzelner Titel unmöglich ist. Selbst mit einer Verzerrung, die dem 20-fachen des Maximalwertes entspricht, ist eine Klassifizierung mit einer Accuracy von 67% möglich. Es zeigt sich also erneut die Robustheit der Merkmale, da Verzerrungen in diesem Ausmaß in der Praxis kaum vorkommen dürften und eine Klassifikation solchen Materials ohnehin nicht von Interesse ist.

Zusammenfassung und Ausblick

Die Klassifikation von umfangreichen Wertereihen wie Audiodaten erfordert die effiziente Extraktion von aussagekräftigen Merkmalen direkt aus der Wellenform der Daten. Die Methoden der statistischen Zeitreihenanalyse wurden hierzu in Gruppen eingeteilt und systematisiert. Außerdem wurde die Effizienz aller Methoden durch Laufzeitanalysen nachgewiesen. Zusammen mit einem erweiterten Konzept der Fensterung können Methodenbäume zur Merkmalsextraktion konstruiert werden. Es wurde gezeigt, dass die Fensterung effizienter Methoden ebenfalls effizient arbeitet, das gleiche gilt für die Methodenbäume. Sie können also für die Extraktion aus umfangreichen Wertereihen verwendet werden.

Darüberhinaus erlaubt die Systematisierung und die Effizienz der Methoden die Anwendung automatisierter Verfahren, um eine optimale Merkmalsmenge zu extrahieren. Der in dieser Arbeit vorgestellte Ansatz basiert auf genetischer Programmierung. Die Individuen sind Methodenbäume, die nur auf einer Teilmenge der Daten evaluiert werden brauchen. Mutationen ändern die Struktur der Bäume oder tauschen Methoden aus, Kreuzungen sind realisiert durch Umhängen von Teilbäumen. Als Ergebnis wird ein Methodenbaum geliefert, der auf dem kompletten Datensatz zur Merkmalsextraktion eingesetzt werden kann.

Die Transformation von Wertereihen zu Merkmalsvektoren wurde anhand verschiedener Klassifikationsaufgaben für Audiodaten demonstriert. Mit einem automatisch extrahierten Merkmalssatz war eine Genreklassifikation Klassik / Pop fehlerfrei möglich, die wesentlich schwerere Unterscheidung der Genres Techno / Pop bzw. Hip Hop / Pop gelingt mit einem Fehler von 7% bzw. 17%. Die Klassifikation von Benutzerpräferenzen ist mit einem Fehler von durchschnittlich 9% möglich. Weitere Experimente zeigten, dass die Robustheit der Merkmale weit über praktische Anforderungen hinaus gehen. Es ist kein Qualitätsunterschied für alle gängigen Kodierungsbitraten festzustellen. Erst wenn das Signal so stark verzerrt ist, dass auch menschlichen Hörern keine Genrezuordnung mehr gelingt, ist eine Verfälschung der Vorhersageergebnisse zu beobachten.

Im ersten Kapitel wurden Kriterien genannt, an denen sich die vorgestellte automatisierte Merkmalsextraktion messen soll. Diese werden nun im einzelnen untersucht:

Klassifikationsgüte: Die vorgestellten Merkmale eignen sich für eine Klassifikation von Audiodaten. Für die Klassifikationsaufgabe KLASSIK/POP wurde eine fehlerfreie Vorhersage auf dem verwendeten Datensatz erreicht. In [54, 55] wurde für diese Lernaufgabe eine Accuracy von 93% angegeben. Für die Lernaufgabe HIPHOP/POP, für die aus [55] eine Accuracy von etwa 66% berechnet werden kann, wurde eine Genauigkeit von 82% ermittelt. Auch wenn keine standardisierten Datensätze existieren, so entspricht die Klassifikationsgüte zumindest der gleichen Größenordnung oder ist besser als die bisher erreichten Genauigkeiten in der Vorhersage.

Verständlichkeit: Alle extrahierten Merkmale waren in der Domäne der Musikdaten erklärbar. Die Interpretation der Winkel und Abstände der Elemente im Phasenraum gestaltete sich am schwierigsten, da diese Merkmale bisher nicht für Audiodaten extrahiert wurden und eine Rekonstruktion des Zustandsraums für Audiodaten noch nicht durchgeführt wurde.

Minimierung des Benutzeraufwandes: Durch die automatisierte Auswahl und Kombination der Methoden wird der Benutzeraufwand für die Merkmalsextraktion minimiert. Es müssen nur noch die Daten bereitgestellt und die Klassen definiert werden. Das Ergebnis ist ein Methodenbaum, der zur Merkmalsextraktion vor einem beliebigen Lernverfahren auf beliebigen Daten angewendet werden kann. Durch die Integration in die maschinelle Lernumgebung YALE kann die gleiche Umgebung zur Merkmalsextraktion und zum Lernen verwendet werden.

Einfache Extraktion von Hand: Der Benutzer hat jederzeit die Möglichkeit, eigene Methodenbäume zu konstruieren und auf diese Weise von Hand Merkmale zu extrahieren. Diese können leicht anderen Methodenbäumen hinzugefügt werden, auch solchen, die automatisiert generiert wurden. Die Integration in YALE erlaubt eine einheitliche Baumstruktur und Parametereinstellung, durch die graphische Benutzeroberfläche ist das Zusammenstellen von Methodenbäumen denkbar einfach.

Vollständigkeit: Sofern bekannt wurden alle bereits genutzten Methoden zur Merkmalsextraktion für die Klassifikation von Wertereihen integriert. Eine Vollständigkeit für alle Verfahren der Zeitreihenanalyse wurde nicht angestrebt. Durch das Konzept der allgemeinen Fensterung sind neue Möglichkeiten der Kombination entstanden.

Systematisierung: Die Systematisierung der Methoden zur Wertereihenanalyse und die Einteilung in die Gruppen Transformationen und Funktionale ermöglicht sowohl eine automatisierte Validitätsprüfung der Methodenbäume als auch eine gute automatisierte Merkmalsextraktion. Bis jetzt konnten alle Methoden in diese Systematik gruppiert werden. Dies gilt auch für die Untergruppe der Auszeichnungstransformationen, die bisher nicht einzuordnen waren.

Robustheit: Die extrahierten Merkmale sind robust gegenüber verschiedene Qualitäten der Kompression sowie unterschiedlich starke Verzerrungen. Die Anfälligkeit für Verzerrungen ist etwas höher. Allerdings ändert sich die Klassifikationsgüte erst für nicht mehr als hörbar zu bezeichnende Daten.

Effizienz: Alle eingesetzten Methoden wurden auf ihre Effizienz überprüft. Die meisten arbeiten mit linearer Laufzeit $O(n)$ oder mit $O(n \log n)$. Lediglich die diskrete Fourier-Transformation arbeitet mit quadratischer Laufzeit, kann aber durch die schnelle Fourier-Transformation ersetzt werden.

Echtzeitfähigkeit: Die Übertragung der Ergebnisse einer automatisierten Merkmalsextraktion für einen Echtzeitbetrieb sind nur bedingt möglich. Sie gelingt nur für allgemeingültige Lernaufgaben wie die Genreklassifikation. Dort erzeugt man einmal die Merkmale und implementiert die Extraktion effizient, so dass eine Extraktion und Klassifikation während des Hörens möglich ist. Bei der Klassifikation nach Benutzerpräferenz ist dieses Vorgehen nicht möglich, da eine ständige Adaption geschehen müsste und die Anzahl der Generationen bei der automatisierten Merkmalsextraktion unbekannt ist.

Die meisten der Kriterien wurden erfüllt. Die Übertragung der Merkmalsextraktion auf Echtzeitanwendungen gelingt jedoch nur bei allgemein festgelegten Klassifikationsaufgaben wie der nach Genre. Zudem wurden einige Detailfragen nur angeschnitten und verdienen eine genauere

Betrachtung. Auch wurden einige weitergehende Anforderungen deutlich. Im einzelnen waren dies:

Visualisierung: Eine Merkmalsextraktion durch den Benutzer kann nur mit Wissen über den Datenraum durchgeführt werden. Einer Merkmalsextraktion und dem Lernen geht daher häufig eine Dateninspektion der Wertereihe in verschiedenen Räumen und nach Anwendung von Filtern voraus. Wünschenswert wäre die Definition von Schnittstellen, so dass verschiedene Systeme auf unterschiedliche Formen der Visualisierungen zurückgreifen können und die Einbindung weiterer Operatoren zur Datenvisualisierung.

Zusätzliche Transformationen: Zusätzliche Basistransformationen und Filter sind denkbar. Eine Transformation, die verschiedene Wavelets in der Reihe sucht, und die Berechnung von TFIDF Werten der vorkommenen Wavelets könnte ebenfalls interessante Merkmale liefern. Auch eine weitergehende Berücksichtigung nichtlinearer dynamischer Systeme ist wünschenswert, so z. B. die Bestimmung von Attraktoren und ihrer Invarianten.

Anwendung auf anderen Datensätzen: Liefern die behandelten Methoden auch auf anderen umfangreichen Datensätzen ähnlich gute Ergebnisse? Eine praktische Evaluierung steht noch aus.

Weniger Beispiele: Es hat sich gezeigt, dass eine Beschaffung der Daten gerade für die Klassifikation der Benutzerpräferenzen, problematisch ist. Kaum ein Benutzer kann ausreichend negative Beispiele nennen. Eine Bestimmung von wahrscheinlich negativen Beispielen aus einer Musikdatenbank und anschließender Rückfrage beim Benutzer (*active learning* [53]) könnte dieses Problem beseitigen.

Beispielsgewichtung: Manche Lieder sind prototypisch für ein bestimmtes Genre oder den Benutzergeschmack. Diese dürfen keinesfalls falsch klassifiziert werden. Eine Gewichtung der Beispiele oder einzelner Passagen soll diesem Umstand Rechnung tragen.

Effizientere Fensterung: Das allgemeine Konzept der Fensterung hat sich als gut und effizient erwiesen. Der Allgemeinheit wird allerdings eine weitere Effizienzsteigerung geopfert. Der gleitende Durchschnitt einer Reihe kann z. B. leichter berechnet werden, indem man sich die Summe der Werte merkt und stets bezüglich des aktuellen Fensters aktualisiert und anschließend durch die Fensterbreite dividiert. Es stellt sich die Frage, ob diese Effizienzsteigerung, die im speziellen für gleitende Durchschnitte durchgeführt werden kann, auch im Allgemeinen durch die Entwicklung geeigneter Datenstrukturen funktioniert. Kann man also auch bei anderen Funktionalen und Transformationen „updaten“, um Laufzeit zu sparen?

Im Bereich der Audiodaten sind mit Hilfe der vorgestellten Merkmale verschiedene Anwendungen denkbar. Ein Musik-Einkaufsberater kann inhaltsbasierte Vorschläge für Benutzer erstellen. Online Portale und Versandhändler können Kunden auf diese Weise individuell bewerben. Ein erweitertes Abspielprogramm kann momentane Stimmungen berücksichtigen und das Benutzerverhalten analysieren. Überspringt der Benutzer einzelne Lieder werden diese als negativ gekennzeichnet und die Abspielliste kann entsprechend adaptiert werden. Zu guter Letzt ist auch ein elektronischer Diskjockey (DJ) denkbar, der die Lieder in ihrer Abfolge betrachtet. Dabei muss die Ähnlichkeit der Zeitreihen untereinander beachtet und eine vorgegebene Stimmung oder Situation berücksichtigt werden. Auf diese Weise kann das Programm für verschiedene Veranstaltungen wie Hochzeiten oder Diskotheken benutzt werden.

ANHANG A

Implementierung

Die Implementierung aller vorgestellten Methoden erfolgte in Java. Im Vergleich zu einer auf einen bestimmten Prozessortyp optimierten C++ - Implementierung ist diese zwar langsamer, dafür jedoch in die Experimentierumgebung YALE integriert. Dadurch sind die Methoden schnell benutzbar, kombinierbar, und erlernbar. Auf diese Weise können konzeptionelle Ideen überprüft werden, eine Umsetzung in reale Systeme muss dann gesondert erfolgen.

Das Konzept von YALE basiert auf Bäumen von Operatoren, welche durch ihre Eingabe, ihre Ausgabe und ihre Funktion definiert sind [12, 33]. Damit lässt sich das Konzept der Methodenbäume direkt in YALE eingliedern. Jede Transformation und jedes Funktional ist ein YALE Operator, der das Interface `ValueSeriesOperator` implementiert. Es ist durchaus möglich, dass mehrere Funktionale in einem Operator vereinigt sind, z. B. Durchschnitt und Varianz oder dass eine Berechnung auf mehrere Transformationen und Funktionale verteilt wird (Bsp: Winkel im Phasenraum). Diese allgemeine Schnittstelle erlaubt beliebige Operationen auf einer Wertereihe, unter anderem die besprochenen Transformationen und die Extraktion von Merkmalen bzw. die Anreicherung mit Auszeichnungen. Weiterhin werden Methoden bereitgestellt, die die Änderung eines Parameters übernehmen für die Benutzung in einem Automatisierungsansatz wie der in Kapitel 8 besprochenen genetischen Programmierung. Jeder Operator kann darüberhinaus definieren, ob er überhaupt für eine solche Automatisierung benutzt werden darf.

Von dem Interface erben weitere Schnittstellen, die von den Funktionen und Transformationen implementiert werden müssen. Für jede dieser Schnittstellen wird eine abstrakte Klasse bereit gestellt, so dass die meisten Methoden bereits sinnvoll implementiert werden. Alle hier besprochenen Methoden werden durch eine solche Klasse implementiert, die zugleich ein YALE Operator als auch ein Objekt des Typs `ValueSeriesOperator` ist. Abbildung A.1 zeigt die Schnittstellen und ihre Hierarchie.

- interface de.ingo_mierswa.valueseries.ValueSeriesOperator
- interface de.ingo_mierswa.valueseries.functions.Function
- interface de.ingo_mierswa.valueseries.transformations.Transformation
  - interface de.ingo_mierswa.valueseries.transformations.basis.BasisTransformation
  - interface de.ingo_mierswa.valueseries.transformations.filter.Filter
  - interface de.ingo_mierswa.valueseries.transformations.markup.MarkupTransform.

Abbildung A.1.: Die Schnittstellen-Hierarchie für die Methoden zur Wertereihenbehandlung

Zusätzlich zu den Methoden stehen auch noch die in Kapitel 9 besprochenen Operatoren zur Visualisierung, zum Einladen der Reihen (insbesondere der Musikdaten) und zur Aufteilung der Vorverarbeitung zur Verfügung. Genauere Informationen zu den Klassen und Methoden sind der API-Dokumentation zu entnehmen, die dieser YALE Erweiterung beiliegt. Der Quellcode ist unter Berücksichtigung der GNU Public License (GPL) frei einsehbar.

Verfügbare Operatoren zur Behandlung von Wertereihen

Die Beschreibung aller YALE Operatoren zur Behandlung von Wertereihen würde den Rahmen dieses Dokumentes sprengen. Daher steht eine englische Dokumentation der Operatoren und ihrer Parameter zur Verfügung [32]. An dieser Stelle werden lediglich alle Operatoren in ihren jeweiligen Gruppen aufgelistet und kurz beschrieben. Jeder Operator definiert durch die Implementierung einer abstrakten Methode, ob er im Rahmen automatischer Vorverarbeitung eingesetzt werden kann oder nicht.

Allgemeine Operatoren

Allgemeine Operatoren zur Behandlung von Wertereihen, die auch die Möglichkeit zum Einladen von Reihen mit anschließender Merkmalsextraktion bieten. Weitere Operatoren visualisieren die Ergebnisse oder schreiben sie in Dateien.

Branching: Teilt die weitere Vorverarbeitung auf.

GnuPlotWriter: Schreibt die Reihe in ein Gnuplot Dateiformat.

MusicExampleSource: Liest Musikstücke ein und bildet Beispiele aus ihnen (ExampleSet).

MusicPlay: Spielt eine Wertereihe als Musikstück ab.

MusicPreprocessing: Liest Musikstücke ein und extrahiert Merkmale gemäß der inneren Operatoren. Bildet aus den Merkmalsvektoren Beispiele (ExampleSet).

MusicWriter: Schreibt die Reihe in ein Musik-Dateiformat.

SinusGenerator: Generiert eine Superposition von harmonischen Schwingungen (zu Testzwecken).

ValueSeriesGP: Führt auf den gegebenen Daten eine automatisierte Merkmalsextraktion mit Hilfe genetischer Programmierung durch.

ValueSeriesPreprocessing: Extrahiert Merkmale aus den als Instanzenmenge gegebenen Reihen (ExampleSet).

Visualizer: Zeigt die Reihe und Merkmale an.

Funktionale

Funktionale umfassen alle Operatoren zur Merkmalsextraktion, d. h. alle Methoden aus Kapitel 7 über Funktionale. Einige Operatorketten erlauben die Definition innerer Funktionen und bilden somit die Kombinationsfunktionale.

AbsoluteAverage: Berechnet den Durchschnitt der Beträge.

Amplitude: Berechnet die größte Auslenkung.

AmplitudeIndex: Berechnet die Stelle der größten Auslenkung (wie Max- oder MinIndex).

Average: Berechnet eine Durchschnittsfunktion, z. B. das arithmetische oder das geometrische Mittel.

Centroid: Berechnet den Zentroiden.

CombinedFunction: Berechnet die arithmetische Kombination zwei anderer Funktionale (Kinder).

Interval2SingleValues: Berechnet Merkmale aus den Intervallen der Wertedimensionen.

LengthGenerator: Berechnet die Länge der Reihe.

LinearRegression: Berechnet Steigung, y-Achsenabschnitt und Diskrepanz der Regressionsgeraden.

Max: Berechnet das Maximum der Reihe.

MaxIndex: Berechnet die Stelle des Maximums der Reihe.

Min: Berechnet das Minimum der Reihe.

MinIndex: Berechnet die Stelle des Minimums der Reihe.

NonZeroValues: Berechnet alle Werte, die nicht Null sind.

NullGenerator: Liefert alle Werte der Reihe als Einzelwerte (Merkmale).

PeakFinder: Liefert die Amplitude, Stelle und Breite der k höchsten Peaks.

SingleCombinedFunction: Berechnet eine beliebige Funktion eines anderen Funktionals.

StatisticalIntervalFunction: Berechnet statistische Werte aus den Intervallen (auch Indexdimension).

WeightedAverage: Berechnet das gewichtete arithmetische Mittel.

Transformationen

Bevor wir die Implementierung der einzelnen Methoden vorstellen, diskutieren wir einige allgemeine Transformationen, die bei der Analyse von Wertereihen helfen können.

Complex2Real: Wandelt die komplexen Zahlenwerte der Reihe entweder in Realanteil, Imaginäranteil oder Betrag der komplexen Zahl um.

DisplacementIntervalWindowing: Eine Fensterung, die von der Auszeichnung in der Indexdimension abhängt.

FirstKValues: Wählt die ersten k Werte als Sample.

FromToValues: Wählt einen definierbaren Bereich der Reihe als Sample.

LastKValues: Wählt die letzten k Werte als Sample.

RandomKValues: Wählt zufällig k zusammenhängende Werte als Sample.

RandomPercentValues: Wählt zufällig einen Anteil zusammenhängender Werte als Sample.

ValueDimensionChange: Wechselt die betrachtete Wertedimension im Falle multivariater Daten.

Windowing: Die allgemeine Fensterung mit beliebigen Schrittweiten, Fensterbreiten oder Überlappungsgraden. Durch Festlegung des Überlappungsgrades ist dieser Operator auch als dynamische Fensterung verwendbar.

Basistransformationen

Diese Operatoren entsprechen den in Kapitel 3 vorgestellten Transformationen zur Änderung des Raums der Reihen.

AngleTransformation: Bildet die Winkel zwischen Teilstücken und verwendet diese als neue Reihe.

AutoCorrelation: Berechnet die Autokorrelation, z. B. zur Bestimmung des Tempos.

CompleteIntervalsTransformation: Bildet aus jedem Wert einen neuen Wert in Abhängigkeit des Intervalls des Wertes.

DiscreteFourierTransform: Führt eine diskrete Fourier-Transformation durch.

DisplacementDifferences: Bildet die Abstände der Werte ungleich Null in der Indexdimension und verwendet diese als neue Reihe.

DistanceTransformation: Bildet die Abstände der Elemente der Reihe und verwendet diese als neue Reihe.

FastFourierTransform: Führt eine schnelle Fourier-Transformation durch.

IntervalTransformation: Bildet für jedes Intervall der alten Reihe einen neuen Wert.

PhaseSpaceTransformation: Führt eine Rekonstruktion des Zustandsraums durch. Bildet hierzu die Transformierte der Reihe im Phasenraum.

Filter

Diese Operatoren entsprechen den in Kapitel 4 vorgestellten Transformationen zur Filterung von Reihen.

CharacteristicaExtractor: Behält die Werte der Extrema und setzt die übrigen Werte auf Null.

DifferenceFilter: Berechnet die Differenz zwischen dem aktuellen Wert und seinem Vorgänger für alle Werte.

DisplacementSubValue: Subtrahiert von jedem Wert in der Indexdimension den korrespondierenden Wert in der Wertedimension.

DistortionFilter: Überlagert die Wertereihe mit einem zufälligen Rauschen mit einer gegebenen Standardabweichung.

ExponentialAverage: Berechnet eine exponentielle Glättung.

FilterTransformation: Filtert Werte nach gegebenem Kriterium aus, z. B. für einen Hoch- oder Tiefpass.

Frequency2Bark: Skaliert ein Frequenzspektrum mit der Bark-Skala.

Frequency2ERB: Skaliert ein Frequenzspektrum mit der ERB-Skala.

NonZeroFilter: Entfernt alle Werte gleich Null. Dabei können auch aus äquidistanten Reihen nicht-äquidistante Reihen entstehen.

ScaleFilter: Wendet eine Skalierungsfunktion auf die Reihe an.

SmallValuesChopper: Entfernt alle Werte, die kleiner als $p\%$ vom Maximum sind.

TimeWindowFunction: Gewichtet jüngere Werte stärker.

WindowFunction: Gewichtet die Mitte eines Fensters stärker als die Ränder.

ZeroCrossingFilter: Erzeugt eine neue Reihe, bei denen alle Stellen mit Nulldurchgang den Wert Eins besitzen und die übrigen Null sind.

Auszeichnung

Diese Operatoren entsprechen den in Kapitel 5 vorgestellten Transformationen zur Auszeichnung von Reihen.

FunctionalIntervals: Beginnt ein neues Intervall in Abhängigkeit von einer wählbaren Entscheidungsfunktion.

KMeansIntervals: Bildet Intervalle in den Wertedimensionen mittels Clustering.

ValueIntervalBasedIntervals: Bildet Intervalle in der Indexdimension in Abhängigkeit von Intervallen in einer Wertedimension.

Forderungen an die Implementierungen

Die Forderung nach Äquidistanz kann ohne weiteres fallengelassen werden, da die Methoden so implementiert sind, dass sie auch mit nicht äquidistanten Daten umgehen können. Bei multivariaten Reihen wird jedoch nur eine Dimension der Reihe zu jedem Zeitpunkt betrachtet.

Die im Text gestellten Forderungen

- Wert x_i einer bestimmten Dimension der Reihe in $O(1)$,
- damit auch $index(i)$ in $O(1)$,
- die Bestimmung des Intervalls des i-ten Punktes in $O(1)$,

sind alle erfüllt. Sowohl die Werte der Index- als auch die der Wertedimensionen stehen als Array komplexer Zahlen zur Verfügung, wodurch ein Zugriff in konstanter Zeit ermöglicht wird. Die Bestimmung des Intervalls des i-ten Punktes in konstanter Zeit ist optional möglich (höherer Speicherverbrauch). Andernfalls wird die Bestimmung des Intervalls in linearer Zeit in Abhängigkeit von der Anzahl der Intervalle durchgeführt, im schlimmsten Fall also in $O(n)$ für eine Reihe der Länge n.

ANHANG B

Vorverarbeitung von Musikdaten

Die Überführung der Musikdaten in ein Zahlenformat, um aus den Daten Reihen zu bilden, ist auf mehrere Weisen möglich. Zunächst kann man die von Java bereitgestellten Routinen zum Einlesen von Audiodaten benutzen. Dies bringt jedoch mehrere Nachteile mit sich:

- Java unterstützt nur wenige Datei- und Kompressionsformate. Eine vorherige Konvertierung muss also ohnehin mit einem externen Programm erfolgen.
- Die Elongationen zu den verschiedenen Zeitpunkten können nicht direkt gelesen werden. Eine Konvertierung in ein Vektorformat ist aber unumgänglich.

Es hat sich daher angeboten, die Konvertierung der Musikdaten in ein Vektorformat mit einem externen Programm durchzuführen: `sox`[1] (*SOund eXchange tool*).

Sox

Das Kommandozeilenformat von `sox` ist

```
sox in.ext out.ext
```

wobei `ext` die Dateierweiterung eines der vielen unterstützten Formate ist. Die möglichen Formate umfassen neben den verschiedenen Wave-Formaten mit unterschiedlichen Kompressionsmethoden auch komprimierte Formate wie `ogg` oder `au`. Das Ausgabeformat `dat` liefert ein Musikstück in einem Textformat. Jede Zeile entspricht einem der äquidistanten sample points:

$$zeitpunkt_i \ elongation_i$$

und die Elongationen sind normiert auf das Intervall -1 bis 1. Außerdem kann mittels der Option -r eine sample rate eingestellt werden, die Option -c definiert die Zahl der Kanäle.

Aus lizenzrechtlichen Gründen kann `sox` jedoch nicht das weitverbreitete MP3 Format konvertieren. Daher ist ein Umweg über ein weiteres externes Programm namens `mpg123`[2] nötig. Dieses

[1] Erhältlich für alle Plattformen unter `http://sox.sourceforge.net/`.
[2] Erhältlich unter `http://www.mpg123.de/`, Versionen für andere Plattformen sind ebenfalls im Internet verfügbar.

erzeugt aus den MP3 Dateien zunächst Wave-Dateien mit der Endung `.wav`:

```
mpg123 -b 10000 -s filename.mp3 | sox -t raw -r 44100 -s -w -c 2 filename.wav
```

Die integrierten Operatoren und Einladeroutinen der YALE Operatoren sorgen für die richtige Konvertierung der Daten, es muss lediglich sichergestellt werden, dass die Programme `mpg123` und `sox` installiert und aufrufbar sind. Weitere Informationen über die betreffenden Programme und ihrer Optionen sind in den Manual-Einträgen zu finden.

ANHANG C

Installation

YALE ist verfügbar unter

`http://yale.cs.uni-dortmund.de`

Auf dieser Seite findet man auch ein Plugin für die Verarbeitung von Wertereihen, welche alle hier vorgestellten Funktionen enthält. Den Anweisungen der Seite folgend das Plugin in das Verzeichnis `lib/plugins` von YALE kopieren.

Danach stehen alle Operatoren zur Behandlung von Wertereihen zur Verfügung. Unter der oben angegebenen Adresse finden sich auch die API-Dokumentation und eine englische Beschreibung aller Operatoren [32].

Literaturverzeichnis

[1] ABARBANEL, H. D. I.: *Analysis of Observed Chaotic Data.* Springer-Verlag, 1996.

[2] BÄCK, T., U. HAMMEL und H.-P. SCHWEFEL: *Evolutionary computation: Comments on the history and current state.* IEEE Transactions on Evolutionary Computation, 1(1):3–17, 1997.

[3] BAUER, M., U. GATHER und M. IMHOFF: *The Identification of Multiple Outliers in Online Monitoring Data.* Technischer Bericht 29/1999, Sonderforschungsbereich 475, Universität Dortmund, 1999.

[4] BELLO, J., G. MONTI und M. SANDLER: *Techniques for Automatic Music Transcription.* In: *Proc. of International Symposium on Music Information Retrieval 2000*, 2000.

[5] BRADLEY, ELISABETH: *Intelligent Data Analysis: An Introduction*, Kapitel Time-Series Analysis. Springer, 1999.

[6] BRILLINGER, R. und R. A. IRIZARRY: *An investigation of the second- and higher- order spectra of music.* Signal Processing, 65:161–179, 1998.

[7] BURGES, C.: *A Tutorial on Support Vector Machines for Pattern Recognition.* Data Mining and Knowledge Discovery, 2(2):121–167, 1998.

[8] CEMGIL, A., B. KAPPEN, P. DESAIN und H. HONING: *On tempo tracking: Tempogram representation and Kalman filtering.* In: *Proceedings of the International Computer Music Conference*, Seiten 352–355, 2000.

[9] COOLEY, J. W. und J. W. TUKEY: *An algorithm for the machine computation of the complex Fourier series.* Mathematics of Computation, 19:297–301, April 1965.

[10] DIXON, SIMON: *An Interactive Beat Tracking and Visualisation System.* In: *Proceedings of the International Computer Music Conference*, Seiten 215–218, 2001.

[11] DROSTE, S., T. JANSEN und I. WEGENER: *On the analysis of the (1+1) evolutionary algorithm.* Technischer Bericht Reihe CI 21/98, SFB 531, Universität Dortmund, Germany, 1998.

[12] FISCHER, SIMON, RALF KLINKENBERG, INGO MIERSWA und OLIVER RITTHOFF: *YALE: Yet Another Learning Environment.* Technischer Bericht CI-136/02, Universität Dortmund, Lehrstuhl Informatik VIII, Juni 2002.

[13] FOOTE, J.: *Content-based retrieval of music and audio.* In: *In Multimedia Storage and Archiving Systems II, Proceedings of SPIE*, Seiten 138–147, 1997.

[14] GHIAS, ASIF., JONATHAN LOGAN, DAVID CHAMBERLIN und BRIAN C. SMITH: *Query by Humming: Musical Information Retrieval in an Audio Database.* In: *Proc. of ACM Multimedia*, Seiten 231–236, 1995.

[15] GUO, G. und S. Z. LI: *Content-Based Audio Classification and Retrieval by Support Vector Machines.* IEEE Transaction on Neural Networks, 14(1):209–215, January 2003.

[16] HARTIGAN, J.: *Clustering Algorithms.* John Wiley and Sons, New York, 1975.

[17] HASTIE, T., R. TIBSHIRANI und J. FRIEDMAN: *The Elements of Statistical Learning.* Springer Verlag, Basel, 2001.

[18] HOLTE, R. C.: *Very simple classification rules perform well on most commonly used datasets.* Machine Learning, 11(1):63–90, 1993.

[19] IMHOFF, M., M. BAUER, U. GATHER und D. LÖHLEIN: *Statistical pattern detection in univariate time series of intensive care on-line monitoring data.* Intensive Care Med, 24:1305–1314, 1998.

[20] IRIZARRY, RAFAEL A.: *Local Harmonic Estimation in Musical Sound Signals.* Journal of the American Statistical Association, 96(454), 2001.

[21] JAYANT, N. S. und P. NOLL: *Digital Coding of Waveforms: Principles and Applications to Speech and Video.* Prentice Hall, 1984.

[22] JOACHIMS, THORSTEN: *Advances in Kernel Methods - Support Vector Learning*, Kapitel Making large-Scale SVM Learning Practical. MIT Press, 1999.

[23] KOZA, J.R.: *Genetic Programming: On the programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, 1992.

[24] KURTH, F. und M. CLAUSEN: *Full-Text Indexing of Very Large Audio Data Bases.* In: *110th Convention of the Audio Engineering Society*, Amsterdam, May 2001.

[25] LEWIS, D. D.: *Evaluating Text Categorization.* In: *Proceedings of Speech and Natural Language Workshop*, Seiten 312–318. Morgan Kaufmann, 1991.

[26] LIU, H. und H. MOTODA: *Feature Extraction, Construction, and Selection: A Data Mining Perspective.* Kluwer, 1998.

[27] LIU, Z., Y. WANG und T. CHEN: *Audio Feature Extraction and Analysis for Scene Segmentation and Classification.* Journal of VLSI Signal Processing System, June 1998.

[28] LOY, G.: *Musicians make a standard: the MIDI phenomenon.* Computer Music Journal, 9(4), 1989.

[29] LU, L., H. JIAN und H. ZHANG: *A robust audio classification and segmentation method.* In: *Proc. of ACM Multimedia*, Seiten 203–211, 2001.

[30] MAROLT, M.: *SONIC: transcription of polyphonic piano music with neural networks.* In: *Proc. of Workshop on Current Research Directions in Computer Music*, Seiten 217–224, 2001.

[31] MIERSWA, INGO: *Beatles vs. Bach: Merkmalsextraktion im Phasenraum von Audiodaten.* In: *LLWA 03 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivität*, 2003.

[32] MIERSWA, INGO: *Value Series Preprocessing with Yale*, December 2003. http://yale.cs.uni-dortmund.de.

[33] MIERSWA, INGO, RALF KLINKENBERG, SIMON FISCHER und OLIVER RITTHOFF: *A Flexible Platform for Knowledge Discovery Experiments: YALE – Yet Another Learning Environment.* In: *LLWA 03 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivität*, 2003.

[34] MITCHELL, M. TOM: *Machine Learning.* McGraw Hill, New York, USA, 1996.

[35] MOORE, B. C. J. und B. R. GLASBERG: *A revision of Zwicker's loudness model.* ACTA Acustica, 82:335–345, 1996.

[36] MORIK, K. und S. WESSEL: *Making Robots Smarter – Combining Sensing and Action through Robot Learning*, Kapitel Incremental Signal to Symbol Processing, Seiten 185–198. Kluwer Academic Publ., 1999.

[37] MORIK, KATHARINA: *The Representation Race – Preprocessing for Handling Time Phenomena.* In: *Proceedings of ECML 2000*, 2000.

[38] MORIK, KATHARINA, MICHAEL IMHOFF, PETER BROCKHAUSEN, THORSTEN JOACHIMS und URSULA GATHER: *Knowledge Discovery and Knowledge Validation in Intensive Care.* Artificial Intelligence in Medicine, 19(3):225–249, 2000.

[39] MORIK, KATHARINA und HARALD LIEDTKE: *Learning about time.* Technischer Bericht, Univerität Dortmund, Lehrstuhl Informatik VIII, 2000. MiningMart Deliverable No. 3.

[40] PACKARD, N. H., J. P. CRUTCHFIELD, J. D. FARMER und R. S. SHAW: *Geometry from a time series.* Physical Review Letters, 45:712–716, 1980.

[41] PICKENS, JEREMY: *A Survey of Feature Selection Techniques for Music Information Retrieval.* Technischer Bericht, Center of Intelligent Information Retrieval, Department of Computer Science, University of Masschusetts, 1996.

[42] PYLE, DORIAN: *Data Preparation for Data Mining*, Kapitel Series Variables. Morgan Kaufmann, 1999.

[43] QUINLAN, ROSS: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Diego, CA, USA, 1993.

[44] RIBBROCK, A. und F. KURTH: *A Full-Text Retrieval Approach to Content-Based Audio Identification.* In: *International Workshop on Multimedia Signal Processing*, 2002.

[45] RITTHOFF, OLIVER, RALF KLINKENBERG, SIMON FISCHER und INGO MIERSWA: *A Hybrid Approach to Feature Selection and Generation Using an Evolutionary Algorithm.* Technischer Bericht CI-127/02, Universität Dortmund, Lehrstuhl Informatik VIII, Februar 2002.

[46] ROADS, CURTIS: *The Computer Music Tutorial.* MIT Press, 1996.

[47] RÜPING, STEFAN: *mySVM - Manual.* Universität Dortmund, Lehrstuhl Informatik VIII, Oktober 2000.

[48] RÜPING, STEFAN und KATHARINA MORIK: *Support Vector Machines and Learning about Time.* In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, 2003.

[49] SCHLITTGEN, RAINER und BERND H.J. STREITBERG: *Zeitreihenanalyse.* Oldenbourg Verlag München, 1997.

[50] STÜCK, HANS-HERMANN: *Statistik.* Wilhelm Heyne Verlag, 1976.

[51] TAKENS, F.: *Dynamical Systems and Turbulence*, Kapitel Detecting strange attractors in fluid turbulence, Seiten 366–381. Springer, 1981.

[52] TIPLER, PAUL A.: *Physics for Scientists and Engineers*. Worth Publishers, 1991.

[53] TONG, S. und D. KOLLER: *Support Vector Machine Active Learning with Applications to Text Classification*. In: LANGLEY, PAT (Herausgeber): *Proceedings of ICML-00, 17th International Conference on Machine Learning*, Seiten 999–1006. Morgan Kaufmann Publishers, San Francisco, US, 2000.

[54] TZANETAKIS, GEORGE: *Manipulation, Analysis and Retrieval Systems for Audio Signals*. Doktorarbeit, Computer Science Department, Princeton University, June 2002.

[55] TZANETAKIS, GEORGE, GEORG ESSL und PERRY COOK: *Automatic musical genre classification of audio signals*. In: *In Proceedings of the Int. Symposium on Music Information Retrieval (ISMIR)*, Seiten 205–210, 2001.

[56] WALLIS, W. ALLEN und HARRY V. ROBERTS: *Methoden der Statistik*. Rowohlt Verlag, 1975.

[57] WEIHS, C., S. BERGHOFF, P. HASSE-BECKER und U. LIGGES: *Mathematical Statistics and Biometrical Applications*, Kapitel Assessment of Purity of Intonation in Singing Presentations by Discriminant Analysis, Seiten 395–410. Josef Eul, Bergisch-Gladbach, Köln, 2001.

[58] WELTNER, KLAUS, HARTMUT WIESNER, PAUL-BERND HEINRICH, PETER ENGELHARDT und HELMUT SCHMIDT: *Mathematik für Physiker*. Vieweg, 1994.

[59] WIDMER, GERHARD: *Trying to Explain a Creative Act: Studying Expressive Music Performance with Learning Machines*. In: *In Proceedings of the ESCOM Conference on Musical Creativity, Liège, Belgium*, 2002.

[60] WITTEN, IAN H. und EIBE FRANK: *Data Mining*. Morgan Kaufmann, San Diego, CA, USA, 2000.

[61] ZHANG, T. und C. KUO: *Content-based Classification and Retrieval of Audio*. In: *SPIE's 43rd Annual Meeting - Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, San Diego, July 1998.

[62] ZWICKER, E.: *Psychoakustik*. Springer-Verlag, Berlin, 1982.

Index