Visualisierung kontinuierlicher, multimodaler Schmerz Scors am Beispiel akustischer Signale

Masterarbeit

Franz Anders
HTWK Leipzig

Januar 2017

Abstract

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen der medizinischen Schrei-Forschung **2**
2.1 Schmerz Scores . . . 2
2.2 Schmerz-Schrei aus medizinischer Sicht . . . 4
2.3 Phyisio-Akustische Modellierung des Weinens . . . 5

3 Überwachtest Lernen **7**
3.1 Klassifizierung . . . 8
3.1.1 ID3 . . . 9
3.1.2 C4.5 . . . 13
3.1.3 Gütemaße binärer Klassifikatoren . . . 15
3.2 Regression . . . 16
3.2.1 Simple Linear Regression . . . 17
3.2.2 Multiple Linear Regression . . . 18
3.3 Methoden der experimentellen Evaluation . . . 20
3.3.1 k-fold Crossvalidation . . . 20

4 System zur Visualisierung akustischer Schmerz-Scores **22**
4.1 Architektur . . . 22
4.2 Preprocessing . . . 24
4.3 Voice Activity Detection . . . 25
4.3.1 Windowing . . . 26
4.3.2 Feature Extraction . . . 26
4.3.3 Thresholding . . . 34
4.3.4 Markierung der Cry-Units . . . 38
4.3.5 Decision Smoothing . . . 40
4.4 Segmentierung . . . 42
4.5 Feature-Extraction . . . 46
4.6 Ableitung der Schmerz-Scores . . . 46
4.7 Visualisierung . . . 46

5 Zusammenfassung **47**

Appendices **51**

Abbildungsverzeichnis

2.1 Veranschaulichung des Grundvokabulars 6

3.1 Entscheidungsbaum, der durch den ID3-Algorithmus für den Datensatz aus Beispiel 3.2 erzeugt wurde. 10
3.2 Knoten eines Entscheidungsbaums für kontinuierliche Werte 14
3.3 Confusion-Matrix 15
3.4 Beispiel eines einfachen linearen Regressionsmodells $X = Latitude$ und $Y = MortalityofSkinCancer$ [30] 17
3.5 Zwei Kandidaten für ein simples lineares Regressionmodell 18
3.6 Lineare Regression eines zweidimensionalen Feature-Raumes [10, S. 73] . . . 19
3.7 Schematische Darstellung der Aufteilung in Test- und Trainingsdatensatz bei einer 5-fold Crossvalidation 21

4.1 Architektur des Systems 23
4.2 Parameter eines Audio-Kompressors 25
4.3 Ergebnis des Preprocessings 25
4.4 Markierung von Schreigeräuschen im Audiosignal 26
4.5 Features des Zeitbereiches 27
4.6 Features des Frequenz-Bereiches 28
4.7 Entstehung des Cepstrums 30
4.8 Cepstrum 31
4.9 Features des Cepstrums 31
4.10 Autokorrelation eines Signals 32
4.11 Features der Autokorrelation 32
4.12 Differenz-Signal des RMS-Wertes 33
4.13 Thresholding eines Feature-Signales 34
4.14 Zusammenfassung klassifizierter Signalfenster zu Cry-Units 39
4.15 Beziehung zwischen agrenzenden Segmenten 39
4.16 Klassifizierung vor dem Decision Smoothing 41
4.17 Klassifizierung vor und nach dem Decision Smoothing 42
4.18 Ergebnis der Segmentierung 43
4.19 Mögliche Segmentierungen eines Signals 43

.1 Boxplot-Auswertung über Sensitivity, Specificity und Accuracy der beiden VAD-Modelle 53

1 Einleitung

2 Grundlagen der medizinischen Schrei-Forschung

2.1 Schmerz Scores

Bei erwachsenen Menschen wird der Schmerzgrad typischerweise durch eine Selbsteinschätzung des Patienten unter der Leitung gezielter Fragen des Arztes vorgenommen. Bei Kindern unter 3 Jahren ist diese Selbsteinschätzung nicht möglich. Schmerz drückt sich in Veränderungen des psychologischen, körperlichen und biochemischen Verhaltens des Säuglings aus. Die für den Arzt am leichtesten feststellbaren Verhaltensänderungen sind von außen wahrnehmbaren Merkmale, wie zum Beispiel ein Verkrampfen des Gesichtsausdrucks, erhöhte Körperbewegungen oder lang anhaltendes Weinen. Um eine weitesgehend objektive Schmerzfeststellung zu ermöglichen, wurden sogenannte *Pain-Scores* entwickelt, die durch ein Punktesystem den insgesamten Schmerzgrad des Babies quantifizieren.[29] Es existieren *eindimensionale* Pain-Scores, die den Schmerz nur Aufgrund der Beobachtung eines Merkmals beurteilen, so wie beispielsweise die reine Beurteilung des Gesichtsausdruckes. *Mehrdimensionale* (auch *multimodale*) Pain-Scores beziehen mehrere Faktoren in das Scoring mit ein.[1]. Tabelle 2.1 zeigt das Scoring-System „Neonatal Infant Pain Scale“(NIPS) als Beispiel für eine multimodale Pain-Score. Der Säugling wird anhand der aufgeführten Kategorien bewertet und alle vergebenen Punkte aufsummiert. Ein insgesamter Wert von > 3 zeigt Schmerz an, ein Wert von > 4 großen Schmerz.[12]

Tabelle 2.1: NIPS-Scoring

NIPS	**0 points**	**1 point**	**2 points**
Facial Expr.	Relaxed	Contracted	-
Cry	Absent	Mumbling	Vigorous
Breathing	Relaxed	Different than basal	-
Arms	Relaxed	flexed/stretched	-
Legs	Relaxed	flexed/stretched	-
Alertness	Sleeping	uncomfortable	-

In den meisten mehrdimensionalen Scoring-Systeme werden die Schreigeräusche mit einbezogen. Tabelle 2.2 zeigt eine Übersicht über eine ausgewählte Menge an multimodalen Pain-Scores. Alle Pain-Scores sind für Kleinkinder bis 3 Jahren gedacht. In der Übersicht wird nicht wiedergegeben, welche weiteren Merkmale jeweils in das Scoring mit einbezogen werden, oder welche Insgesamtpunktzahlen auf welche Schmerzintensität hinweisen. Es soll an dieser Stelle nur verdeutlicht werden, welche unterschiedlichen Ansätze zur Bewertung des Schreiens aus medizinischer Sicht im Zusammenhang mit Pain-Scores existieren. Folgende Beobachtungen lassen sich aus der Übersicht ziehen:

1. Die zu beobachtenden Eigenschaften des Weinens werden mit subjektiv behafteten Werten charakterisiert. Beispielsweise wird im N-PASS-System ist ein Schmerz-Schrei

als „High-pitched or silent-continuous crying“ beschrieben. Es wird nicht fest definiert, was als „crying“ gilt oder welche Tonhöhe als „high-pitched“ ist. Auch die Erstquellen geben keine festen Definitionen.

2. Es gibt verschiedene Ansätze zur Bewertung des Weinens. Bei CRIE ist die Tonhöhe, bei BIIP die Länge und bei COMFORT die Art des Weinens entscheidend.

3. Die Beschreibungen sind kurz und prägnant gehalten, der Arzt hat in keinem der Modelle auf mehr als drei Parameter des Schreiens zu achten. Die Begründung liegt darin, dass bei allen Modellen a.) das Schreien nur eines von mehreren Faktoren ist, und b.) Die Schmerzbestimmung in einem vorgegebenen Zeitrahmen durchführbare sein muss.

System	**P.**	**Description**
FLACC[36]	0	No cry (awake or asleep)
	1	Moans or whimpers; occasional complaint
	2	Crying steadily, screams or sobs, frequent complaints
N-PASS[31]	-2	No cry with painful stimul
	-1	Moans or cries minimally with painful stimuli
	0	Appropiate Crying
	1	Irritable or Crying at Intervals. Consolable
	2	High-pitched or silent-continuous crying. Not consolable
BIIP[9]	0	No Crying
	1	Crying <2 minutes
	2	Crying >2 minutes
	3	Shrill Crying >2 minutes
CRIES[3]	0	If no cry or cry which is not high pitched
	1	If cry high pitched but baby is easily consoled
	2	If cry is high pitched and baby is inconsolable
COVERS[16]	0	No Cry
	1	High-Pitched or visibly crying
	2	Inconsolable or difficult to soothe
PAT[13]	0	No Cry
	1	Cry
DAN[4]	0	Moans Briefly
	1	Intermittent Crying
	2	Long-Lasting Crying, Continuous howl
COMFORT[27]	0	No crying
	1	Sobbing or gasping
	2	Moaning

	3	Crying
	4	Screaming
MBPS[28]	0	Laughing or giggling
	1	Not Crying
	2	Moaning quiet vocalizing gentle or whimpering cry
	3	Full lunged cry or sobbing
	4	Full lunged cry more than baseline cry

Tabelle 2.2: Übersicht über Pain-Scores

2.2 Schmerz-Schrei aus medizinischer Sicht

Die Frage ist: Woher kommen diese unterschiedlichen Bewertungen des Weinens in Tabelle 2.2? Gibt es eine Pain-Score, die aus wissenchaftlicher Sicht „recht hat“? Dieser Fragestellung unterliegen unterliegen zwei grundlegendere Fragen: 1.) Ist es überhaupt möglich, anhand der akustischen Eigenschaften den Grund für den Schrei abzuleiten, also beispielsweise Hunger, Einsamkeit oder Schmerz? Anders formuliert: Gibt es überhaupt so etwas wie einen Schmerz-Schrei? 2.) Ist es möglich, anhand der akustischen Eigenschaften den Schweregrad des Unwohlseins abzuleiten (also beispielsweise den Grad des Schrei-Versursachenden Schmerzes)?

Die Annahme, dass es möglich ist, aus dem Schreien den Grund abzuleiten, wird als „Cry-Types Hypothesis“ bezeichnet. Die berühmtesten Befürworter dieser Hypothese ist eine skandinavische Forschungsgruppe, auch bezeichnet als „Scandinavian Cry-Group“, die diese Idee in dem Buch „Infant Crying: Theoretical and Research Perspectives“[2] publik machte. Die Annahme ist, dass die verschiedenen Ursachen *Hunger, Freude, Schmerz, Geburt und Anderes* klare Unterschiede hinsichtlich ihrer akustischen Merkmale aufweisen, welche an einem Spektogramm ablesbar seien. Entsprechende Beispiele werden in dem Buch gegeben. Nur einige Jahre Später zeigte Müller et al [7] in einem Paper, dass bei leichter Veränderung der Bedingungen der Experimente die Unterscheidung nicht möglich ist. Die Gegenhypothese ist, dass Weinen „nichts als undifferenziertes Rauschen“ sei. 50 Jahre später liegt kein anerkannter Beweis für die eine oder andere Hypothese vor. Es gibt nur starke Hinweise dafür, dass die Plötzlichkeit des Eintretens des Schreigrundes hörbar ist. Ein plötzliches Ereignis, wie ein Nadelstich oder ein lautes Geräuch, führen auch zu einem plötzlich beginnenden Schreien. Ein langsam einretendes Ereignis, wie ein langsam immer stärker werdender physischer Schmerz oder langsam eintretender Hunger führen auch zu einem langsam eintretenden Weinen. Da keine Einigung herrscht, wird empfohlen, den Grund aus dem Kontext abzuleiten.[35]

Die Zweite Frage nach der Ableitung der Stärke des Unwohlseins aus den akustischen Eigenschaften des Geschreis wird in der Fachsprache unter dem Begriff *Cry as a graded Signal* subsumiert. Je „stärker“ das Weinen, desto höher das Unwohlsein (*Level of Distress (LoD)*) des Säuglings. Tatsächlich bemessen wird dabei der von dem Beobachter vermutete Grad des Unwohlsein des Babies, und nicht der tatsächliche Grad, da dieser ohne die Möglichkeit der direkten Befragung des Kindes nie mit absoluter Sicherheit bestimmt werden kann. Dieser vermutete LoD wird entweder durch das subjektive Empfinden der Beobachter oder durch Pain-Scores festgestellt. Ein hohes Level of Distress hat vor allem

eine schnelle Reaktion der Aufsichtspersonen zur Beruhigung des Babies zur Folge, womit dem Geschrei eine Art Alarm-Funktion zukommt. Es gibt starke Hinweise darauf, dass das Level of Distress anhand objektiv messbarer Eigenschaften des Audiosignals bestimmt werden kann. So herrscht beispielsweise weitesgehend Einigung darüber, dass ein „lang“ anhaltendes Geschrei auf einen hohen Level of Distress hinweist. Insofern aus dem Kontext des Schreiens Schmerz als wahrscheinlichste Ursache eingegrenzt werden kann, kann aus einem hohen Level of Distress ein hoher Schmerz abgeleitet werden. [35] und [33]

Es herrscht wiederum keine Einigung darüber, welche akustischen Eigenschaften im Detail ein hohes Level of Distress anzeigen. Carlo V Bellieni et al [4] haben festgestellt, dass bei sehr hohem Schmerz in Bezug auf die DAN-Scala (siehe Tabelle 2.2) die Tonhöhe des Geschreis steigt. Qiaobing Xie et al [33] haben festgestellt, dass häufiges und „verzerrtes“ Schreien (ohne feststellbares Grundfrequenz, da der Ton stimmlos erzeugt wird) auf einen hohen Level of Distress hinweist.[35] Diese Uneinigkeit hat wahrscheinlich zu den verschiedenen Bewertungen in den Pain-Scores geführt. 2.2.

2.3 Phyisio-Akustische Modellierung des Weinens

Das Ziel dieses Kapitels ist die Schaffung eines einheitlichen Vokabulares, auf den sich bezogen wird, um das Schreien eines Babys zu beschreiben. Die hier vorgestellten Begriffe stammen sowohl aus dem Buch „A Physioacoustic Model of the Infant Cry “ H Golub und M Corwin [14] als auch aus dem Paper „Rythmic organization of the Sound of Infant Cry “ von Zeskind et al.[32]

Die Lautäußerung eines Neugeborenen, umgangssprachlich auch als „Weinen“ oder „Schreien“ bezeichnet, lässt sich im allgemeinen beschreiben als das „rythmische Wiederholen eines beim ausatmen erzeugen Geräusches, einer kurzen Pause, einem Einatmungs-Geräusch, einer zweiten Pause, und dem erneuten Beginnen des Ausatmungs-Geräusches.“[40].

Das Vokabular, welches insbesondere von H Golub und M Corwin geschaffen wurde, ist sehr umfassend. An dieser Stelle wird eine Auswahl grundlegender Begrifflichkeiten vorgestellt, die in dieser Arbeit gebraucht werden. Sie werden in Abbildung 2.1 veranschaulicht.

Expiration beschreibt den Klang, der bei einem einzelnen, ununterbrochenem Ausatmen mit Aktivierung der Stimmbänder durch das Baby erzeugt wird. [32]. Der von Golub et al [14] verwendete Begriff **Cry-Unit** wird in dieser Arbeit synonym verwendet. Umgangssprachlich ist handelt es sich um einen einzelnen, ununterbrochenen *Schrei*.

Inspiration beschreibt den Klang, der beim Einatmen durch das Baby erzeugt wird.

Burst beshreibt die Einheit von einer Expiration und der darauf folgenden Inspiration. Das heisst, dass die zeitliche Dauer eines Bursts sowohl das Expiration-Geräusch, das Inspiration-Geräusch als auch die beiden Pausen zwischen diesen Geräuschen umfasst. Praktisch ergibt sich das Problem, dass vor allem bei stärkerem Hintergrundrauschen die Inspiration-Geräusche häufig weder hörbar noch auf dem Spektrogramm erkennbar sind. Daher wird die Zeitdauer eines Bursts oder Cry-Unit vom Beginn einer Expiration bis zum Beginn der darauf folgenden Expiration definiert und somit allein von den Expirations auf die Bursts geschlossen. Implizit wird somit eine Inspiration zwischen zwei Expirations angenommen.

Cry die insgesamte klangliche Antwort zu einem spezifischen Stimulus. Eine Gruppe meh-

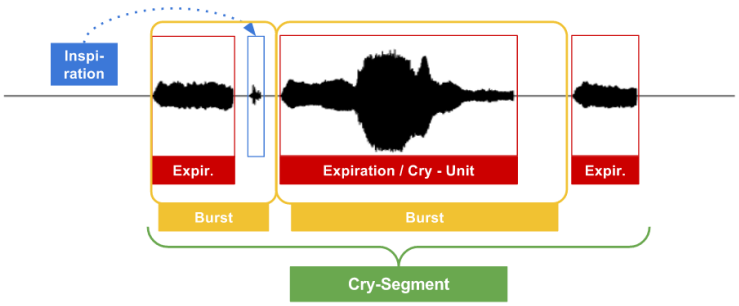

Abbildung 2.1: Veranschaulichung des Grundvokabulars

rerer Cry-Units.[14] In dieser Arbeit wird ein *Cry* als **Cry-Segment** bezeichnet, um Verwechslungen zu vermeiden.

Weiterhin wurden von H Golub und M Corwin [14] Cry-Units in eine der folgenden drei Kategorien eingeführt:

Phonation beschreibt eine Cry-Unit mit einer „vollen Vibration der Stimmbänder“ mit einer Grundfrequenz zwischen 250 und 700 Hz. Entspricht umgangssprachlich einem Weinen mit einem „klaren, hörbaren Ton“.

Hyper-Phonation beschreibt eine Cry-Unit mit einer „falsetto-artigem Vibration der Stimmbänder“ mit einer Grundfrequenz zwischen 1000 und 2000 Hz. Entspricht umgangssprachlich einem Weinen mit einem „sehr hohen, aber klaren, hörbaren Ton“.

Dysphonation beschreibt eine Cry-Unit ohne klar feststellbare Tonhöhe, produziert durch Turbulenzen an den Stimmbändern. Entspricht umgangsprachlichen dem „Brüllen oder Krächzen“.

Eine Cry-Unit gehört dabei mindestens einer dieser Kategorien an, kann aber auch in seinem zeitlichen Verlauf die Kategorie wechseln. H Golub und M Corwin [14] stellen weiterhin eine Reihe an charakteristischen Eigenschaften vor, die in Bezug auf ein Cry-Segment berechnet werden.

Latency-Period beschreibt die Dauer zwischen dem zufügen eines Schmerz-Stimulus und dem beginn des ersten Cry-Bursts des Segmentes

Duration beschreibt die insgesamte Zeitdauer des Cry-Segmentes. Es wird keine genaue Definition gegeben, wodurch Beginn und Ende definiert werden. Das Segment endet dort, wo es „scheint, aufzuhören“.

Maximum-Pitch beschreibt die höchste festgetellte Grunfrequenz des Segmentes.

... und viele weitere, die in [14] nachgelesen werden können, aus Platzgründen an dieser Stelle jedoch nicht vollständig genannt werden.

3 Überwachtest Lernen

Überwachtes Lernen ist ein Wissenschaftsgebiet des *Maschinellen Lernen*. Es existieren verschiedene Definitionen für Maschinelles Lernen. Eine der meist zitierten Definitionen lauteet wie folgt:

> A Computer Program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.[26, S. 2]

Diese weitreichende Definition lässt sich auf verschiedene Wissenschaftsgebiete anwenden. Ein Beispiel ist ein Computer Programm, welches lernt, Dame zu spielen. Angewandt auf die eben genannte Definition lassen folgende Aufgabenbereiche definieren:

- **Task** T: Dame spielen.
- **Performance Measure** P: Prozentsatz gewonnener Spiele gegen Gegner
- **Training Experience** E: Übungsspiele gegen sich selber.

Selbstverständlich könnten P und E in diesem Beispiel beliebig anders gewählt werden. P könnte auch die Freude sein, die der menschliche Gegenspieler beim Spiel erfindet. [26, S. 2 -3]

Die Klasse an Aufgaben, die für diese Arbeit Bedeutung besitzen, ist die des *Überwachten Lernens*. Beim Überwachten exisitert ein *Trainings-Datensatz* mit korrekten Antworten auf die Problem-Fragestellung. Der Algorithmus generalisiert diese Trainings-Beispiele, um auf alle möglichen Datensätze die richtige Lösung zu gewährleisten. [25, S 6]

Ein Beispiel für ein Problem des überwachten Lernens ist die *Erkennung von Handschrift*. Die Aufgabenbereiche werden folgendermaßen identifiziert:

- **Task** T: Erkennung handgeschriebener Worte in Bildern und Zuordnung zu dem tatsächlichen Wort.
- **Performance Measure** P: Prozentsatz korrekt erkannter Wörter
- **Training experience** E: Eine Datenbank mit handgeschriebener Wörter und mit dem tatsächlich geschriebenen Wort. [26, S. 3 - 4]

Dieses Problem gehört zur Unterkategorie der *Klassifzierung*, beschrieben in Kapitel 3.1.

Eine zweite Klasse an Aufgaben des überwachten Lernens, die Bedeutung in dieser Arbeit hat, ist die *Regression*. Ein Beispiel für eine Regressionsaufgabe ist die *Schätzung des Verkaufspreieses einges gebrauchten PKWs*. Folgende Aufgebenbereiche werden identifiziert:

- **Task** T: Schätzung des Marktwertes eines gebrauchten PKWs.
- **Performance Measure** P: Abweichung des geschätzten Wertes zum tatsächlichen Verkaufswert

- **Training experience** E: Eine Datenbank mit gebrauchten PKWs und ihrem tatsächlichen Verkaufswert.

3.1 Klassifizierung

Das Klassifzierungs-Problem wird folgendermaßen modelliert:

Es existieren *Instanzen* x. Jede Instanzen hat eine Reihe an Eigenschaften, bezeichnet als *Features* oder *Attribute* f, wobei jedes Feature einen eigenen Wertebereich, bezeichnet als *Domain* hat. Menge aller möglichen Feautre-Kombinationen wird als *Feature-Raum* X bezeichnet.

$$\begin{aligned} \text{Feature-Raum}: \quad & X = \{\ dom(f_1)\times, \ldots, \times dom(f_n)\ \} \\ \text{Instanz}: \quad & x \in X \end{aligned} \tag{3.1}$$

Außerdem exisistiert eine Menge an *Klassen* $C = \{1, \ldots, k\}$. Die *Klassifizierungs-Funktion*, *Predictor* oder *Classifier* c bestimmt für eine Instanz eine Klasse.

$$\begin{aligned} \text{Classes}: \quad & C = \{1, \ldots, k\} \\ \text{Classifier}: \quad & c : X \mapsto C \end{aligned} \tag{3.2}$$

Es gibt einen Datensatz D mit einer Menge an Instanzen. Für jede der Instanzen ist die zugehörige Klasse bekannt. Ein Paar aus Instanz und Klasse wird als *Example* e bezeichnet. Die einer Instanz x_i zugewiesenen Klasse c_i wird als *Label* beschrieben.

$$\begin{aligned} \text{Datensatz}: \quad & D = \{\langle x_1, c_1\rangle, \ldots, \langle x_n, c_n\rangle\} \\ \text{Example}: \quad & e \in D \end{aligned} \tag{3.3}$$

Die Fehlerfunktion E zählt für einen Datensatz die Menge aller nicht richtig klassifzierten Instanzen

$$\mathrm{E}(D, c) = \operatorname*{count}_{\langle x_i, c_i\rangle \in D} (c(x_i) \neq c_i) \tag{3.4}$$

Das Ziel des Klassfikations-Problems ist es, diejenige Funktion C zu finden, die für einen Test-Datensatz $D_{test} \subseteq D$ die Anzahl falsch klassifzierter Examples minimiert. Nach dem in Kapitel 3 vorgestellten Muster nach T, P und E ergibt sich folgende Aufgabenbeschreibung. [6, S. 8 - 9] [22, S. 14] [25, S. 7 - 10, 18]

- **Task** T: Für einen Test-Datensatz $D_{test} \subseteq D$, finde eine Klassfikations-Funktion c, die die Funktion E minimiert, das heißt: $E(D_{Test}, c) \mapsto min$
- **Performance Measure** P: Die Fehler-Funktion $E(D_{test}, c)$.
- **Training experience** E: Ein Trainings-Datensatz $D_{training} \subseteq D$

Im Zusammenahang mit Klassifikation haben die Klassen C die Eigenschaft, dass die Klassen eine *qualitativen* Charakter, und keinen *quantitativen*. Das heißt, dass die Klassen untereinander keine hierarchische Ordnung besitzen, bei der eine Klasse „besser ist als die andere“. Auch, wenn die Klassen durch Zahlen beschrieben werden sollten (z.B. $C = \{1, 0\}$),

hat diese Zahl einen rein beschreibenden Charakter und macht keine Aussage über die Güte der Instanz. Außerdem handelt es sich um eine diskrete Menge, und keinen kontinuierlichen Zahlenbereich. [10, S. 28, 127]. Ein besonderer Fall der Klassifikation ist ein sogenannter *binärer Klassfikator*, bei dem es nur zwei Klassen gibt: $C = \{0, 1\}$ (oder $C = \{yes, no\}$). Die Domains der Features können ebenfalls qualitativer natur sein, das heißt einen Wert in einem diskreten, ungeordneten Raum annehmen, oder quantitativer Natur, das heißt, einen Wert in einem kontinuierlichen, geordneten Zahlenraum annehmen. [26, S. 54]

Eine andere Art und Weise, die Aufgabenstellung der Klassifikation zu betrachten, ist die *Generalisation zur Prongnose*. Das heißt, dass die Ableitung der Klassen aus den Instanzen des Datensatzes verallgemeinert wird, um in Zukunft für neue, noch nicht bekannte Instanzen die Klasse vorhersagen (prognostizieren) zu können. [25, S. 6 - 7]

Tabelle 3.1 gibt einen Beispiel-Datensatz für eine Klassifkationsproblem. In diesem Beispiel geht es darum, ob abhängig von der Tageszeit und der Temperatur ein Federball-Match Spaß gemacht hat, oder nicht. Das Problem wird folgendermaßen modelliert:

- Es gibt zwei Features: $f_1 = Temperatur$, mit $dom(f_1) = R$, ein quantitives Feature. $f_2 = Tageszeit$, mit $dom(f_2) = morgens, mittags, abends$, ein qualitatives Feature. Der Feature-Raum ist $X = dom(f_1) \times dom(f_2)$.
- Es gibt zwei Klassen, $C = Ja, Nein$.
- Der Datensatz hat fünf Instanzen, $D = \{x_1, \ldots, x_5\}$. Für jede Instanz ist das Label $c_1, \ldots, c_5$ bekannt, welches besagt, ob das Federball spielen Spaß gemacht hat, oder nicht.

Tabelle 3.1: Beispieldatensatz D für eine Klassifikation

x_i	Temperatur	Tageszeit	c_i = Spaß?
x_1	20	morgens	Ja
x_2	15	abends	Ja
x_3	8	mittags	Nein
x_4	23	mittags	Ja
x_5	10	morgens	Nein

Das Ziel ist, in Zukunft abschätzen zu können, allein durch die Kenntniss der Temperatur und Tageszeit abschätzen zu können, ob das Federball-Spielen Spaß machen wird, damit man von vorneherein keine Matches beginnt, die keine Aussicht auf Spaß haben. An dieser Stelle wählt man einen Algorithmus, der einen Classifier baut, der dieses Problem löst.

Es gibt eine Reihe an Algorithmen, die Classifikatoren nach unterschiedlichen Methoden erstellen. Beispiele sind für *k-NN*, *Support-Vector-Maschienen* oder *künstliche Neuronale Netze*. Eine Klasse an Klassifikations-Algorithmen, die in dieser Arbeit Anwendung finden, sind die sogenannten *Entscheidungsbäume*

3.1.1 ID3

Es gibt drei Algorithmen zur Erzeugung von Entscheidungsbäumen, die weitreichende Einsatz finden: *ID3*, *C.45* und *CART*, wobei die letzteren Erweiterungen der grundlegenden

Idee des *ID3*-Algorithmus darstellen. Daher wird an dieser Stelle zuerst der *ID3*-Algorithmus vorgestellt.

Es wird zunächst davon ausgegangen, dass alle Features diskret und nicht kontinuierlich sind. Tabelle 3.2 gibt einen Beispieldatensatz, an dessen Beispiel ein Classificator mit Hilfe des ID3 erzeugt wird. Es geht ähnlich dem Beispiel aus Tabelle 3.1 um die Frage, ob Federball-Spielen abhängig von Temperatur und Tageszeit Spaß macht, nur sind in diesem Fall alle Features diskret.

Tabelle 3.2: Beispieldatensatz D für die Kassfikation mit ID3

x_i	Temperatur	Tageszeit	c_i = Spaß?
x_1	warm	Tag	Ja
x_2	kalt	Tag	Ja
x_3	normal	Nacht	Nein
x_4	kalt	Nacht	Nein
x_5	normal	Tag	Ja
x_6	warm	Nacht	Ja

Abbildung 3.1 zeigt einen Klassifikator, den der ID-3 Algorithmus für diesen Datensatz baut. Es handelt sich um einen Entscheidungsbaum. In Jedem Knoten steht ein Feature, welches einen Ast für jeden möglichen Wert dieses Features bildet. In den Blättern stehen die Klassen.[25, S. 134]

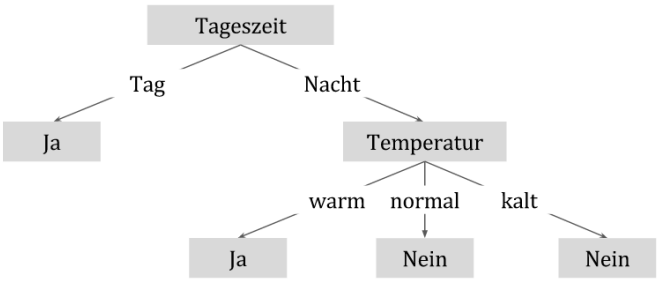

Abbildung 3.1: Entscheidungsbaum, der durch den ID3-Algorithmus für den Datensatz aus Beispiel 3.2 erzeugt wurde.

Der Entscheidungsbaum lässt sich in eine Reihe von `if ... then ...` -Regeln transformieren. Jeder Weg von der Wurzel bis zu einem Blatt ergibt eine Entscheidungsregel, bei der Feauture-Werte der betretenen Kanten konjunktiv Verknüpft werden und die Klasse implizieren. Die Entscheidungsregeln für den Baum aus Abbildung 3.1 sind: [25, S. 134]

- `if` *Tageszeit* = *Tag* `then` *Spaß* = *Ja*
- `if` *Tageszeit* = *Nacht* `and` *Temperatur* = *warm* `then` *Spaß* = *Ja*
- `if` *Tageszeit* = *Nacht* `and` *Temperatur* = *normal* `then` *Spaß* = *Nein*
- `if` *Tageszeit* = *Nacht* `and` *Temperatur* = *kalt* `then` *Spaß* = *Nein*

Der Klassifikator, das heißt der Entscheidungsbaum, wird beim ID3 Algorithmus nach folgenden Muster erstellt: Der Baum wird Top-Down erzeugt, dass heisst beginnend bei der Wurzel bis zu den Blättern. Da in jedem Knoten genau ein Feature aufgespalten wird,

wird an der Wurzel die Frage gestellt „*Welches Feature sollte zuerst getestet werden?*“. Um diese Frage zu beantworten, wird jedes Feature einem statistischen Test unterzogen und festzustellen, wie „gut“ es zur Klassifikation der Trainings-Daten beiträgt. Das „beste“ Attribut wird ausgewählt und als Wurzel festgelegt. Nun wird ein Kind für jeden möglichen Wert des Features gebildet. Der Datensatz des Elternknotens wird in disjunkte Teilmengen aufteilt, wobei jedes Kind die Untermenge erhält, die den jeweiligen Feature-Wert bestitzt. Daraufhin beginnt für jedes Kind der Prozess des Auswählen des „besten“ Attributes von vorn. Ein Kind wird dann zu einem Blatt, wenn seine Teilmenge an Daten nur noch aus Instanzen einer Klasse besteht und somit kein weiteres Aufteilen notwendig ist.[26, S. 55]

Das Wort „gut“ wurd ein dieser Beschreibung in Anführungsstrichen geschrieben, da es subjektiv ist und quantifiziert werden muss. Zur Quantifizierung der Information wird die Entropie nach Formel 3.5 als Hilfsmittel definiert. p_i ist die Wahrscheinlichkeit, dass in einem Datensatz D eine Instanz mit der Klasse $i \in C$ angetroffen wird.

$$H(p) = -\sum_{i \in C} p_i \cdot \log_2 p_i \tag{3.5}$$

Die Entropie quantifiziert die *Unreinheit des Datensatzes*. Angenommen, ein Datensatz hat zwei Klassen, $C = \{+, -\}$. Existiert der gesamte Datensatz nur aus einer der beiden Klasse, ist die Entropie $-p_+ \log_2 p_+ - p_- \log_2 p_- = -1log_21 - 0log_20 = 0$. Das heißt, dass die *Unreinheit des Datensatzes* 0 beträgt. Ist die *Unreinheit des Datensatzes* hingegen maximal, das heißt es liegen exakt 50% positive und 50% negative Samples vor, ist die Entropie $-p_+ \log_2 p_+ - p_- \log_2 p_- = -0.5log_{0.5}1 - 0.5log_20.5 = 1$. [25, S. 135]

Es ist das Attribut in einem Knoten zu wählen, welches den höchsten *Informatoinsgewinn* gewährleistet, das heißt, zu einer bestmöglichen *Reinheit* bei der alleinigen Unterteilung des Datensatzes auf Basis dieses Attributs führt. Der informationsgewinn eines Features f für den Datensatz D wird nach Formel 3.6 definiert. v sind alle möglichen Werte dieses Features. $|D|$ beschreibt die Anzahl an Instanzen des Datensatzes. D_v ist die Untermenge an Instanzen, die für das Feature f den Wert v besitzen.[25, S. 136 - 137]

$$\text{Gain}(D, f) = H(D) - \sum_{v \in dom(f)} \frac{|D_v|}{|D|} H(D_v) \tag{3.6}$$

Für das Beispiel aus Tabelle 3.1 ergibt sich für den ersten Test folgende Berechnung des Informationsgewinnes der beiden Features *Temperatur* und *Tageszeit*. Da die Tageszeit den höheren Informationsgewinn gewährleistet, wird dieses Features in der Wurzel gewählt.

$$H(D) = -p_+ \log_2 p_+ - p_- \log_2 p_- = -\frac{4}{6}\log_2(\frac{4}{6}) - \frac{2}{6}\log_2(\frac{2}{6}) = 0.91 \tag{3.7}$$

$$Gain(D, Tageszeit) = 0.91 - \Big(\overbrace{\frac{3}{6} \cdot (-\frac{3}{3}\log_2\frac{3}{3} - -\frac{0}{3}\log_2\frac{0}{3})}^{Tag} \underbrace{\frac{3}{6} \cdot (-\frac{1}{3}\log_2\frac{1}{3} - -\frac{2}{3}\log_2\frac{2}{3})}_{Nacht} \Big) = 0.86 \tag{3.8}$$

$$Gain(D, Temperatur) = 0.91 - \Big(\overbrace{\frac{2}{6} \cdot (-\frac{2}{2}\log_2\frac{2}{2} - -\frac{0}{2}\log_2\frac{0}{2})}^{warm} \overbrace{\frac{2}{6} \cdot (-\frac{1}{2}\log_2\frac{1}{2} - -\frac{1}{2}\log_2\frac{1}{2}}^{normal} \overbrace{\frac{2}{6} \cdot (-\frac{1}{2}\log_2\frac{1}{2} - -\frac{1}{2}\log_2\frac{1}{2})}^{kalt} \Big) = 0.66 \tag{3.9}$$

Algorithmus1 zeigt den Ablauf des ID-3 in Pseudocode. D ist die Menge aller Test-Examples, X ist die Menge aller Features, C ist die Menge aller Klassen, f_{parent} das Feature des momentanen Eltern-Knotens und v_{parent} der Wert des zum momentan konstruierten Knotens eingehenden Kante. [25, S. 139] [26, S. 56]

Algorithm 1 ID3-Algorithmus in Pseudocode

```
1: tree = {}
2: function ID3(D, X, C, f_parent, v_parent )
3:                                  ▷ If all Examples have the same label, return a leaf with that Label
4:     if ∀e ∈ D : ∃k ∈ C : e.c = k then
5:         tree = tree ∪ {(f_parent, v_parrent, k)}
6:         return
7:     else
8:                                                  ▷ If there are no Features left to test, return a leaf with
9:                                                ▷ the most common Label of the Examples remaining in D
10:        if isEmpty(X) then
11:            tree = tree ∪ {(f_parent, v_parrent, most common Label in D)}
12:            return
13:        else
14:                          ▷ Choose the feature that maximizes the Information-Gain to be the next node
15:            f_best = max_{f∈X} Gain(D, f)
16:                                                                  ▷ Add a Branch to this node
17:            tree = tree ∪ {(f_parent, v_parrent, f_best)}
18:                                                ▷ Remove the feature from the set of features
19:            X_/f ← X/f_dom
20:            for v ∈ f_best do
21:                ▷ Calculate the new Dataset D_/f by removing all instances with the corresponding value
22:                D_/f ← ∀e ∈ D : e.f_best = v
23:                                                              ▷ Recursivly call the algorithm
24:                ID3(D_/f, X_/f, f_dom, v)
25:            end for
26:        end if
27:    end if
28: end function
```

Der ID3-Algorithmus hat folgende **Vorteile**:

Kurze Entscheidungsbäume Der Klassifizierer versucht, möglichst kurze Entscheidungsbäume zu bauen, indem Features mit hohem Informationsgewinn bevorzugt werden. Dies ist eine Umsetzung von *Ocam's Razor*: „Bevorzuge die kürzeste Hypothese“

Verständlichkeit Der Klassifikator ist für den Menschen verständlich, da er sich in Regeln übersetzen lässt (im Gegensatz zu zum Beispiel Neuronalen Netzen). Es existiert

die unbewiesene Hypothese, dass der Mensch bei der Klassifizierung intuitiv ähnlich vorgeht wie der ID3-Algorithmus.[26, S. 63 - 65]

Der ID3-Algorithmus hat folgende **Nachteile**

Nur Diskrete Werte Der Algorithmus akzeptiert keine kontinuierlichen Werte [26, S. 72]

Overfitting Der Algorithmus neigt zu *Overfitting*. Overfitting bedeutet, dass der erzeugte Klassfikator c zwar einen möglichst geringen Fehler in Bezug auf den *Trainings-Datensatz* hat, es jedoch einen anderen Klassifikator c' gibt, welcher in Bezug auf den Trainings-Datensatz einen höheren Fehler erzeugt, jedoch einen geringeren Fehler als c in Bezug auf *alle möglichen Instanzen dieses Typs* erzeugt. Anders formuliert bedeutet Overfitting, dass der Klassifikator den Trainings-Datensatz „auswendig gelernt hat“ und nicht mehr genügend generalisiert, um auf im Training nicht enthaltene Instanzen angewandt werden zu können. Overfitting im Zusammenhang mit dem ID-3 Algorithmus wird durch *Rauschen im Trainings-Datensatz* bedingt. Es gibt keinen festen Beweis für das vorhandensein von Overfitting. Methoden zum Feststellen von Overfitting sind:

- Verwendung eines seperaten Test-Datensatzes, welcher bestätigt, dass der für den Trainings-Datensatz erzeugte Klassifikationsfehler auch bei bisher unbekannten Instanzen erzeugt wird.
- Verwendung von Statistischen Tests, die eine signifikante Reduktion des Klassifikationsfehlers bei Erweiterung des Entscheidungsbaumes beweisen.
- Expertenwissen über applikationstypischen Tiefen von Entscheidungsbäumen.[26, S. 66 - 70]

Lokale Maxima Der Algorithmus bevorzugt greedy Attribute, die zum Zeitpunkt der Berechnung den höchsten Inforamtionsgewinn gewährleisten. Dabei besteht die Gefahr, dass der Algorithmus in ein lokales Maximum läuft.[26, S. 66 - 70]

3.1.2 C4.5

Der *C4.5*-Algorithmus erweitert den *ID3*, um dessen Nachteile auszumerzen, das heißt die Möglichkeit der Einführung kontinuierlicher Attribute sowie Lösungsansätze für das Overfitting. An dieser Stelle wird der *C4.5* nicht im Detail vorgestellt, sondern die Erweiterungskonzepte vorgestellt. [26, S. 66]

Kontinuierliche Werte

Der *C4.5* ermöglicht sowohl diskrete als auch kontinuierliche Attribute. Für ein kontinuierliches Attribut wird in ein *boolsches Attribut* entworfen, das heißt ein Attribut mit dem Wertebereich $\{0, 1\}$. Zur Abbildung des kontinuierlichen Attributs auf das boolsche Attribut wird durch einen Grenzwert c auf dessen kontinuierlichen Wertebereich festgelegt, in Form von `if` $f_c > c$ `then` 1 `else` 0. Abbildung 3.2 visualisiert einen solchen Knoten in einem Entscheidungsbaum. [26, S. 72]

Angenommen, das Attribut *Temperatur* aus Tabelle 3.2 wird in ein kontinuierliches Attribut umgewandelt, in dem die Temperatur in Grad ausgedrückt wird, wie Tabelle 3.3 zeigt.

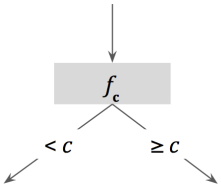

Abbildung 3.2: Knoten eines Entscheidungsbaums für kontinuierliche Werte

Tabelle 3.3: kontinuierliche Attribut-Werte des Features „Temperatur“

Temperatur	23	5	15	-2	18	26
Spaß	Ja	Ja	Nein	Nein	Ja	Ja

Das Ziel ist nun, denjenigen Grenzwert c zu finden, der den größt möglichen Informationsgewinn für dieses Attribut erzeugt. Das Vorgehen zum finden dieses Grenzwertes ist wie folgt:

1. Ordnen aller Examples nach dem kontinuerlichen Feature f_c.
2. Identifizierung benachbarter Examples mit unterschiedlicher Klasse als Kandidaten für einen Grenzert
3. Errechnung des Informationsgewinn bei Setzung des Grenzwertes auf den Attributwert jedes gefundenen Kandidaten. Wahl desjenigen Grenzwetes, der den höchsten Informationsgewinn bringt. [26, S. 73]

Tabelle 3.4 zeigt die nach dem Temperatur-Wert geordneten Examples aus Tabelle 3.3 zur Verdeutlichung des Vorgehens. In diesem Beispiel bilden die Examples mit Temperatur-Werten von $-2, 5, 15$ und 18 Kandidaten. Der höchste Inforamtionsgewinn wird mit einem Informationsgewinn nach der Entscheidungsregeln `if` $Temperatur > 15$ `then` 1 `else` 0.

Tabelle 3.4: kontinuierliche Attribut-Werte des Features „Temperatur“

Temperatur	-2	5	15	18	23	26
Spaß	Nein	Ja	Nein	Ja	Ja	Ja

Pruning

Das in Kapitel 3.1.1 als Overfitting beschriebene Problem lässt sich vermeiden, in dem die Tiefe des Entscheidungsbaumes reduziert wird. Diese Begrenzung wird als *Beschneiden* oder *Pruning* bezeichnet. Es gibt grundlegend zwei verschiedene Ansätze: (1) Ab der Überschreitung einer bestimmten Tiefe der Algorithmus frühzeitig stoppen, oder (alias *Pre-Pruning*) (2) zuerst den kompletten Entscheidungsbaum aufbauen und Overfitting zulassen, um diesen im Nachhinein in seiner Tiefe reduzieren (alias *Post-Pruning*). Post-Pruning hat sich insgesamt als erfolgreicher herausgestellt. [26, S. 68 - 69]

Eines der am weitesten verbreiteten Post-Pruning-Algorithmen ist das sogenannte *Reduced Error Pruning*. Dabei wird ein Knoten des Entscheidungsbaumes zu einem Blatt umgewandelt und diesem Blatt das Label zugewiesen, welches in seinem Sub-Baum am häufigsten vorkommt. Daraufhin wird der originale Entscheidungsbaum und sowie der beschnittene Entscheidungsbaum verwendet, um den Test-Datensatz zu klassifizieren. Ist

der Klassifizierungsfehler des beschnittenen Baumes nicht schlechter als der des originalen Baumes, wird das Pruning übernommen. Dieses Vorgehen wird für jeden Knoten des Entscheidungsbaumes angewandt.

3.1.3 Gütemaße binärer Klassifikatoren

Ein binärer Klassifikation ist eine, bei dem es nur zwei Klassen gibt, das heißt $|C| = 2$. Applikationsabhängig werden die beiden Klassen als *Positive* und *Negative*, 1 und 0 oder *True* und *False* beschrieben. Eine Klassifikation, bei der ein tatsächliches Positive richtig als Positive vorhergesagt wird, spricht man von einem *True Positive* [TP]. Wird hingegen ein tatsächliches Positive fälschlicherweise als Negative vorhergesagt, spricht man von einem *False-Negative* [FN]. Das System wird entsprechend für die Klassifikation tatsächlicher Negatives angewandt und ergibt. *True-Negatives* [TN] und *False-Positives* [FP]. Die *Confusion Matrix* in Abbildung 3.3 gibt eine Übersicht über die vier möglichen Klassifikations-Ergebnisse. [21, S. 213 - 214]

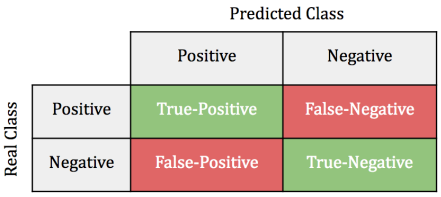

Abbildung 3.3: Confusion-Matrix

Die insgesamte Güte einer Klassifikation wird durch die *Accuracy* nach Formel 3.10 bestimmt. Eine Accuracy von 100% bedeutet, dass *alle* Instanzen richtig klassifiziert werden, eine Accuracy von 50% bedeutet, dass die Hälfte aller Instanzen richtig klassifiziert werden, was der Güte einer rein zufälligen Wahl entspricht. [21, S. 214]

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{3.10}$$

Die Accuracy beziffert die insgesamte Performance des Klassifikators, gibt jedoch keinen Aufschluss darüber, ob der Klassifikator eher eine Tendenz zur falschen Klassifizierung von Positives oder Negatives hat. Bei einer Datenbank mit der selben Anzahl an Positives und Negatives kann eine Accuracy von 50% beispielsweise dadurch entstehen, dass *alle* Instanzen als Positives markiert werden, also sowohl die Positives richtigerweise als Positives, aber die Negatives fälschlicherweise ebenfalls als Positives. Im Umgedrehten Fall ergibt die Klassifizierung aller Instanzen als Negatives ebenfalls eine Accuracy von 50%. In einem dritten Fall irrt sich die Klassifikator gleich oft bei der Einordnung der Negatives und Positives. Die Maße *Sensitivity* und *Specificity* geben Aufschluss über die Güte der Klassifikation hinsichtlich der Positives und Negatives. Die *Sensitivity*, auch bezeichnet als *True-Positive-Rate*, bemisst den Anteil tatsächlicher Positives, die auch als solche erkannt wurden, nach Formel 3.11. Eine Sensitivity von 100% bedeutet, dass alle Positives durch den Klassifikator erkannt wurden. Die Erkennungsrate der Negatives hat keinen Einfluss auf die Sensitivity. Eine hohe Sensitivity lässt sich somit „einfach“ erzielen, in dem man *alle* Instanzen immer als Positives klassifiziert. Die Specificity nach Formel 3.12 bestimmt analog zur Sensitivity den Anteil der korrekt als Negatives bestimmten Instanzen. Ein

Klassifikator, der alle Instanzen als Positives markiert, hat zwar eine Sensitivity von 100%, aber eine Specificity von 0%. Ergeben zwei verschiedene Klassifikationsmodelle sehr ähnliche Accuracies, hilft die Bestimmung der Sensitivity und Specificity bei der Auswahl des für den Anwendugnsfall Adäquteren Klassifikators. So ist beispielsweise bei der Bestimmung von schweren Krankheiten eventuell ein Klassifikator mit höherer Sensitivity wünschbar, um die Wahrscheinlichkeit zu minimieren, dass die entsprechende Krankheit nicht erkannt wird. [23] [21, S. 222]

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{3.11}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{3.12}$$

3.2 Regression

Während der Klassfikator c bei der Klassifikation die Instanzen auf eine *diskrete, quantitative* Menge abbildet werden, werden bei der Regression die Instanzen auf eine *kontinuierliche, qualitative* Menge abgebildet. Der Klassifikator c wird in der Regression als *Regressor* r bezeichnet, und die Menge, auf die die Instanzen abgebildet werden, als *Output*, *Predicted Attribut* oder *abhängige Variable* Y. Wird keine explizite Einschränkung für Y gegeben, so gilt $Y = \mathbb{R}$. [6, S. 24] [25, S. 8] [10, S. 28]

Die in Kapitel 3.1 benannten Definitionen für die Begriffe *Feature-Raum* und *Instanzen* aus Gleichung 3.1 sowie für *Datensatz* aus Gleichung 3.3 werden bei der Formalisierung der Regression übernommen. Die Defintion von *Classes* und *Classifier* aus Formel 3.2 wird Ersetzt durch *Output* und *Regressor* nach Gleichung 3.13.[6, S. 24]

$$\begin{aligned} \text{Output}: &\quad Y = \mathbb{R} \\ \text{Regressor}: &\quad r : X \mapsto Y \end{aligned} \tag{3.13}$$

Ein Datensatz D enthält bei der Regression Tupel aus Samples und dem bekannten Output.

$$\text{Datensatz}: \quad D = \{\langle x_1, y_1\rangle, \ldots, \langle x_n, y_n\rangle\} \tag{3.14}$$

Die Fehlerfunktion entspricht dem aus Gleichung 3.15 als Summe aller quadrieten differenzen zwischen dem tatsächlichen und dem durch den Regressor geschätzten Output definiert.[10, S. 29] [6, S. 25]

$$\text{Error}: \quad E(D, r) = \frac{1}{|D|} \sum_{\langle x_i, y_i\rangle \in D} (r(x_i) - y_i)^2 \tag{3.15}$$

Die Regressions-Aufgabe lautet somit wie die in 3.1 beschriebene Aufgabe, mit dem Unterschied, dass die Fehlerfunktion nach Formel 3.15 anstatt der Fehlerfunktion der Klassifikation verwendet wird. [6, S. 25]

Wie bei der Klassifikation existieren verschiedene Algorithmen zur Lösung dieser Aufgabe. An dieser Stelle werden zwei Methoden vorgestellt:

3.2.1 Simple Linear Regression

Bei der simplen linearen Regression geht es um die Prädiktion des Outputs Y aus einer Prädiktions-Variable X. Diese Predictionsvariable X ist ein Feature-Raum mit nur einem kontinuierlichen Feature, das heißt $X = dom(f_c) = \mathbb{R}$. Bei der simplen linearen Regression wird davon ausgegangen, dass einen annährend linearen Zusammenhang zwischen X und Y gibt, das heißt: [10, S. 61]

$$Y \approx \beta_0 + \beta_1 X \tag{3.16}$$

Die Konstanten β_0 und β_1 bezeichnen den *Schnittpunkt* sowie die *Steilheit* des linearen Modells. Kennt man die Konstanten, so kann der Wert y_i eines Examples x_i prognostiziert werden mit $y_i \approx \beta_0 + \beta_1 x_i$. Das Ziel ist somit die Bestimmung β_0 und β_1 zur Definition des einfachen Regressionsmodells.

Abbildung 3.4zeigt ein Beispiel für ein einfaches lineares Regressionsmodell. Der Regressor bildet das Attibut $X = Latitude$ auf die Sterblichkeitsrate bei Hautkrebs Y ab. Die blauen Punkte sind die Samples des Datensatzes X, die rote Linie das Regressionsmodell. Das durch die rote Linie dargestellte Regressionmodell wird auch als „best fitting Line“, da es es sich vereinfacht betrachtet um die Linie handelt, welche den Verlauf von Y mit steigendem X am besten annähert. [30, What is Simple Linear Regression?]

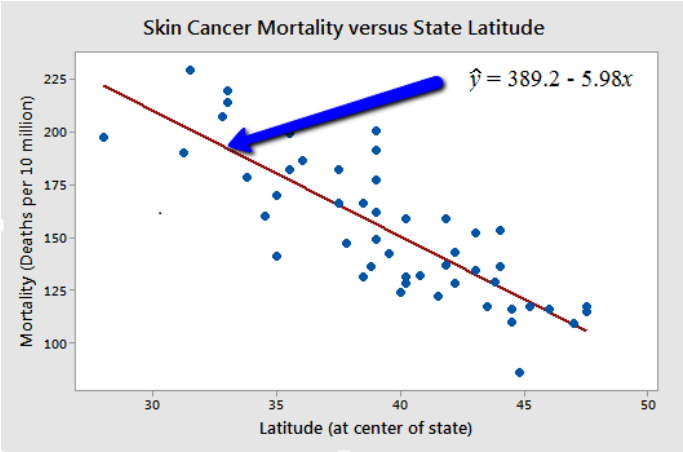

Abbildung 3.4: Beispiel eines einfachen linearen Regressionsmodells $X = Latitude$ und $Y = MortalityofSkinCancer$ [30]

Die Frage ist: Nach welchem Kriterium wird festgelegt, was genau die „best fitting Line“ bestimmt? Abbildung 3.5 verdeutlicht die Fragestellung, in dem zwei Kandidten für simple lineare Regressionsmodelle für einen Datensatz gezeigt werden. Welches dieser beiden Kandidaten schätzt den Verlauf von Y besser? [30, What is the Best Fitting Line?]

Es sei x_i die i-te Prädiktions-Variable, y_i der tatsächliche Output dieser Prädiktionsvariable und $\hat{y}_i = \beta_0 + \beta_1 x_i$ der für diese Variable durch den Regressor geschätzte Wert. Nach dem in Formel 3.15 derjenige Regressor der beste, welcher die Fehlerfunktion der quadrierten Differnenzschen zwischen dem tatsächlichen und dem geschätzten Output minimiert. Da

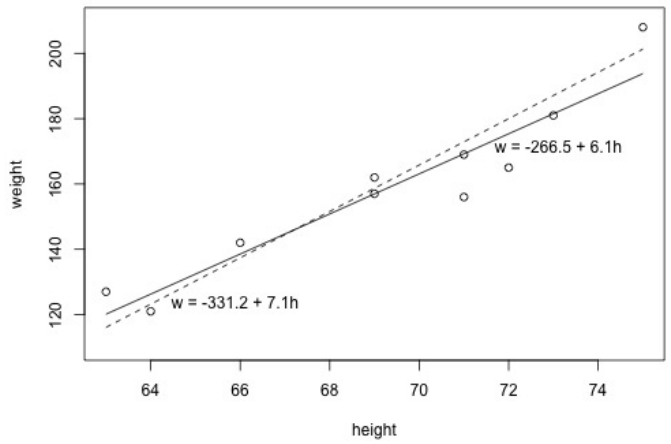

Abbildung 3.5: Zwei Kandidaten für ein simples lineares Regressionmodell

das Teilen des Ergebnisses durch die Anzahl der Elemente einen linearen Skalierungsfaktor darstellt, wird diese Division in dieser Rechnung ausgelassen.

$$E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.17}$$

Durch das Einsetzen von $\hat{y}_i = \beta_0 + \beta_1 x_i$ in Gleichung3.17 und anschließendes Umformen lassen sich die Parameter β_0 und β_1 wie in Formel 3.18 definiert errechnen. Die Parameter $\bar{x}$ und $\bar{y}$ sind die Durchschnittswerte der Prädiktions-Variablen bzw. der Output-Variablen. Auf einen Beweis wird an dieser Stelle aus Platzgründen verzichtet.[10, S. 62]

$$\begin{aligned} \beta_0 &= \bar{y} - \beta_1 \bar{x} \\ \beta_1 &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i_1}^{n}(x_i - \bar{x})^2} \end{aligned} \tag{3.18}$$

3.2.2 Multiple Linear Regression

Bei der *multiplen linearen Regression* (auch *mehrdimensionale lineare Regression*) werden im Gegensatz zur einfachen linearen Präditktions-Variablen eine beliebige Anzahl von Prädiktions-Variablen $X_1, \ldots X_n$ nach Y abgebildet. Das heißt, dass eine Instanz nun mehrere Features besitzen kann, der Feature-Raum wird definiert durch $X = X_1 \times, ... \times X_n$. Das Regressionsmodell entspricht nun einer Hyperebene. Beispielsweise entspricht bei einem zweidimensionalen Featureraum $X = \mathbb{R}^2$ entspricht das Regressionsmodell einer Ebene im dreidimensionalen Raum, wobei die dritte Dimension dem Output Y enspricht. Die einfache lineare Regression ist dementsprechend ein Spezialfall der mehrdimensionalen linearen Regression, bei dem der Feature-Raum nur eine Dimension besitzt. [30, Multiple Linear Regression] [10, S.71] Abbildung 3.6 zeigt ein Beispiel für die lineare Regression eines zweidimensionalen Feature-Raumes.

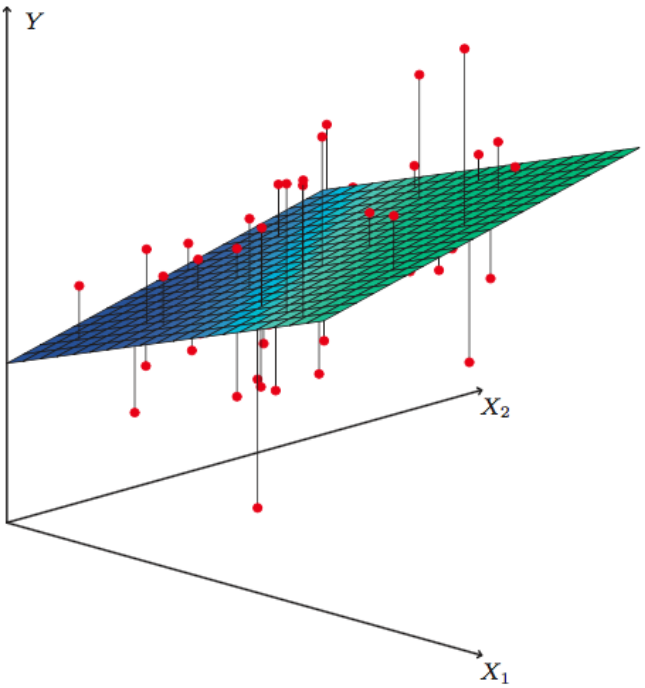

Abbildung 3.6: Lineare Regression eines zweidimensionalen Feature-Raumes [10, S. 73]

Aufbauend auf dem in Gleichung 3.16 definierten einfachen linearen Regressionsmodell wird das mehrdimensionale Regressionsmodell nach Gleichung 3.19 definiert. [10, S. 71]

$$Y \approx \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k = \beta_0 + \sum_{k=1}^{p} \beta_k X_k \tag{3.19}$$

Die in Gleichung 3.17 definierte Fehlerfunktion ist auch bei der mehrdimensionalen linearen Regression anwendbar.

Wie bei der simplen linearen Regression werden die Koeffizienten $\beta = \beta_0 \ldots \beta_k$ durch Umformung durch Einsetzen von Gleichung 3.19 in Gleichung 3.17 und anschließender Umformung hergeleitet. Der Beweis dieser Herleitung ist aufwendig und wird aus Platzgründen an dieser Stelle ausgelassen. Eine detailliete Beschreibung kann in dme Kapitel „A Matrix Formulation of the Multiple Regression Model“ in Quelle [30] gefunden werden. Gleichung 3.20 definiert das Ergebnis zur Berechnung der Koeffizienten β.

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ b_{p-1} \end{bmatrix} = (X'X)^{-1}X'Y \tag{3.20}$$

Die simple und die mehrdimensionale lineare Regression wurden vorgestellt, um dem Leser eine Idee von den Methoden der Regression zu geben. Diese Methoden sind jedoch nur dazu in der Lage, lineare Abhängigkeiten zwischen den Attributen und dem Output zu modellieren. Für nicht-lineare Zusammenhänge jeder Art wurde eine Vielzahl an Methoden und Algorithmen entwickelt.

polinomiellen Regression werden polinomielle Zusammenhänge hergestellt. Das heißt, dass Features der Form X_k^2 modelliert werden können.

Step Functions erlauben die Einteilung des kontinuierlichen Feature-Raumes in Bereiche. Für jeden Bereich wir ein lineares Regressionsmodell gebildet.

Regression Splines stellen eine Kombination der polinomiellen Regression und von Step Functions dar, in dem polinomioelle Regressionsmodelle für den in Bereiche eingeteilten Feature-Raum berechnet werden.[10, S. 265 - 266]

Classification and Regression Trees (*CART*), Entescheidungsbäume ähnlich den in Kapitel 3.1 vorgestellten *ID3* und *C4.5*, die neben der Klassfikation auch zur Regression in der Lage sind. [25, S. 145]

3.3 Methoden der experimentellen Evaluation

Sowohl bei der Klassizierung als auch bei der Regression wird ein Test- und ein Trainingsdatensatz zur Konstruktion und zur Evaluierung des Prediktors verwendet. Im folgenden werden verschiedene Methoden der Aufteilung des Datensatzes vorgestellt.

Einfache Ansätze

Die einfachste Variante ist, keine Aufspaltung in Test- und Trainings-Datesatz festzulegen. Der Gesamte Datensatz wird sowohl zum Training als auch für das Testing verwendet, das heißt $D = D_{Test} = D_{Training}$. Der Nachteil dieser Methode ist, dass nicht feststellbar ist, ob Overfitting vorliegt.

Eine weitere Variante ist, den Datensatz manuell in einen Test- und einen Trainings-Datensatz aufzuspalten. Die Klassifikation/Regression wird mit verschiedenen Paramter-Einstellungen für den jeweiligen Algorithmus auf Basis des Trainings-Datensatzes durchgeführt, und dasjenige Modell gewählt, welches für den Test-Datensatz den geringsten Klassifikationsfehler erzielt. [21, S. 227]

Random Subsampling

Beim Random Subsampling wird das Training und die Evaluation des Klassifikators k mal wiederholt. In jeder Wiederholung wird der gesamte Datensatz in einen Test- und einen Trainingsdatensatz aufgespalten, wobei beide Datensätze 50% der Examples erhalten und die Zuteilung zufällig erfolgt. Die insgesamte Fehlerrate wird als Durchschnitt der aus allen Fehlerraten der k Wiederholungen berechnet. [21, S. 227 - 228]

3.3.1 k-fold Crossvalidation

Eine erweiterung des random Subsampling. Dabei wird der Datensatz in k gleichgroße Untermengen aufgeteilt, bezeichnet als *Folds*. Nun wird die Konstruktion des Prädiktors k mal durchgeführt, wobei bei jeder Wiederholung ein Fold als Test-Datensatz und die anderen als Trainings-Datensatz verwendet werden. Die Fehlerrate wird wie beim Random Subsampling aus dem Durchschnitt aller k Durchläufe errechnet. Der Vorteil im Vergleich

zum Random Subsampling ist, dass garantiert werden kann, dass jedes Example exakt einmal zum testing verwendet wird. Abbildung 3.7 visualsiert das Vorgehen bei der k-fold Crossvalidation schematisch. [21, S. 228]

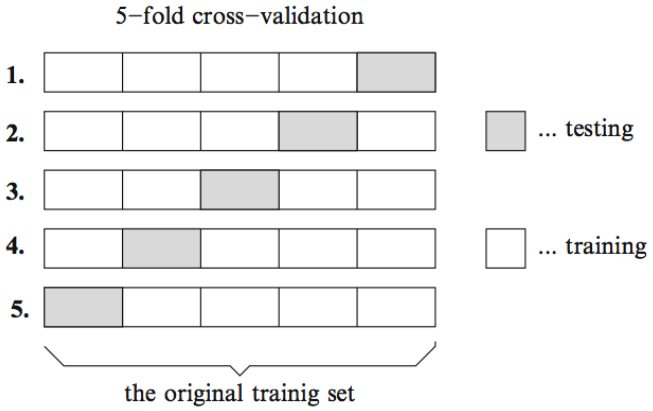

Abbildung 3.7: Schematische Darstellung der Aufteilung in Test- und Trainingsdatensatz bei einer 5-fold Crossvalidation

4 System zur Visualisierung akustischer Schmerz-Scores

4.1 Architektur

Das Ziel dieser Arbeit ist die Ableitung des Schmerz-Scores aus einem Audiosignal sowie die darauf folgende Visualisierung dieser Schmerz-Scores. Um diese komplexe Aufgabenstellung zu lösen, wird sie in einzelne Arbeitspakete zerlegt. Welche Arbeitspakete formuliert werden, ist davon abhängig, aus welcher Perspektive die Aufgabenstellung grundlegend betrachtet wird. So kann sie beispielsweise als Unterproblem der Schrei-Forschung betrachtet werden, woraufhin sie mit Hilfe des in Kapitel 2.3 vorgestellten Vokabulars modelliert wird. Ebenso kann die Aufgabe grundlegend als Problem der automatisierten Spracherkennung verstanden werden, woraufhin ein in diesem Wissenschaftsgebiet üblicher Verarbeitungsprozess entworfen werden würde. Dieser Ansatz wurde beispielsweise in dem von Cohen et al vorgestellten System zur Kategorisierung von Audioaufnahmen von Babys in (0) Nicht-Weinen und (1) Weinen gewählt.[34]

In dieser Arbeit wurde sich für den Ansatz entschieden, die Arbeitspakete aus Sicht der Schreiforschung zu modellieren und dabei das in 2.3 vorgestellte Vokabular zu verwenden. Die einzelnen Arbeitspakete werden daraufhin mit Methoden der Signalverarbeitung gelöst. Hintergrund ist, dass die Arbeitsweise des Systems in Zukunft einfacher Medizinern vermittelbar ist, da die verwendeten Begrifflichkeiten leichter zugänglich sind. Das folgende Vorgehen wird dazu vorgeschlagen:

1. Kontinuierliche Betrachtung des Eingangssignals
2. Markierung der Cry-Units / Expirations in diesem Einganssignal (und eventuelle Kategorisierung in Phonation, Hyper-Phonation und Dysphonation)
3. Gruppierung der markierten Cry-Units zu Cry-Segments
4. Errechnung der charakteristischen Eigenschaften der Segmente
5. Ableitung der Pain-Score aus den Eigenschaften der Segmente.
6. Visualisierung dieser Pain-Score.

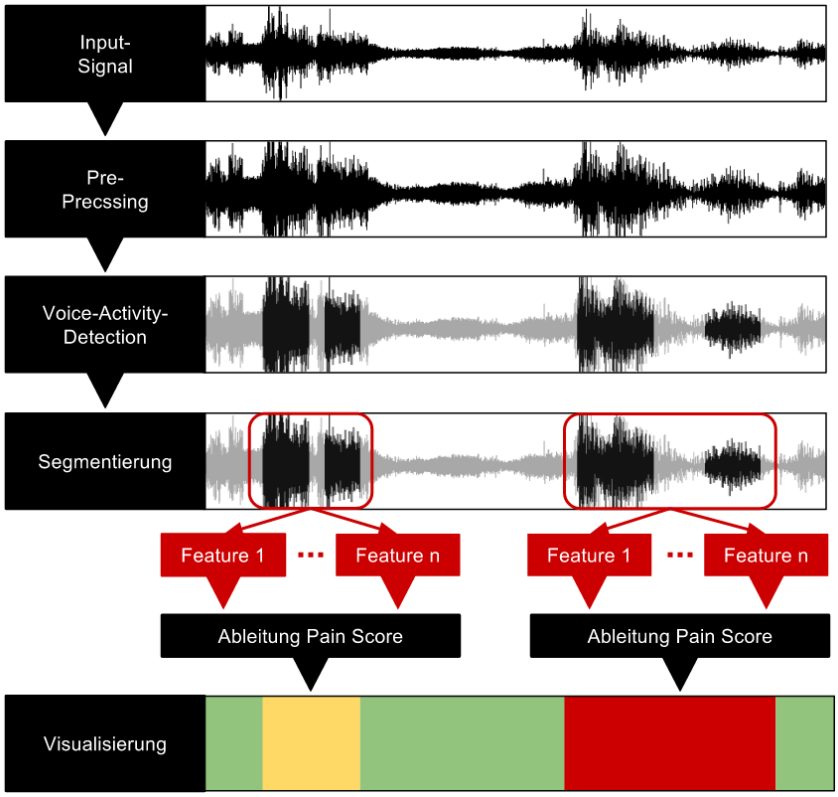

Abbildung 4.1: Architektur des Systems

Diese Arbeitspakete werden nun auf Probleme der automatisierten Spracherkennung abgebildet. Abbildung 4.1 veranschaulicht die Architektur, die sich daraus ergibt. Eingang in das System ist das Audiosignal, welches kontinuierlich in das System gegeben wird. Das Signal passiert daraufhin die folgenden Verarbeitungsschritte:

1. **Pre-Processing**. Das Signal wird vorverarbeitet. Durch die Anwendung von Filtern (linearen wie nicht-linearen) werden die darauf folgenden Verarbeitungsschritte vereinfacht. Das Ergebnis ist ein vorverareitetes Signal. Dieser Verarbeitungsschritt wurde zu den Zeiten der manuellen Schrei-Analyse noch nicht mit einbezogen, ist jedoch in der automatisierten Spracherkennung üblich.
2. **Voice-Activity-Detection**. In dem Signal müssen diejenigen Stellen erkannt werden, die die Stimme des Babys enthalten. Dazu wird das Signal in kurze Fenster weniger Millisekunden Länge zerlegt und jedes Fenster als *Stille* (0) oder *Stimme* (1) markiert. Zusammehängende, als stimmhaft erkannte Signalfenster werden daraufhin zu *Cry-Units* gruppiert.
3. **Segmentierung**, das heißt die Gruppierung der markierten Cry-Units zu *Cry-Segments*.
4. **Feature-Extraction**, das heißt die Berechnung von Eigenschaften für jedes Segment, die für die Ableitung der Pain-Scores von Interesse sind, wie beispielsweise die Duration. Die errechneten Features müssen sich dabei nicht auf diejenigen beschränken, die von H Golub und M Corwin [14] vorgestellt wurden, da diese zur (vermeintlichen) Unterscheidung der Wein-Ursache, und nicht zur Ableitung einer Pain Score gedacht waren.
5. **Ableitung der Pain Score** aus den errechneten Eigenschaften.

6. **Visualisierung** der errechneten Pain-Score.

Für jedes dieser Module werden verschiedene Lösungsansätze vorgestellt und zumindest ein Lösungsansatz implementiert. Das Pre-Processing wird in Kapitel 4.2 vorgestellt, die Voice-Activity-Detection in Kapitel 4.3, die Segmentierung in Kapitel .....

4.2 Preprocessing

Beim Preprocessing wird das Signal so vorverarbeitet, dass Störeinflüsse auf die darauf folgenden Verarbeitungsschritte von vorneherein minimiert werden. Welches Pre-Processing durchgeführt wird, ist Abhägig von der konrketen Aufgabenstellung. So werden beispielsweise bei einigen Algorithmen zur Voice-Activity-Detection, also dem markieren stimmhafter Signalabschnitte, Tiefpass, Hochpass- und Bandpassfilter eingesetzt, um diejenigen Frequenzanteile herauszufiltern, die von der Stimme nicht produziert werden können [20] [37] [19]. Bei einigen Pitch-Detection-Algorithmen wird *Centerclipping* eingesetzt, also das 0-Setzen von Samples mit $x[i] < 0.5 \cdot$ Maximalaussteuerung.[8]

In dieser Arbeit wurde sich für eine Vorverarbeitung entschieden, bei der das Signal hinsichtlich seiner Dynamik im Zeitbereich eingegrenzt wird. Dies ist ein typischer Vorverarbeitungschritt bei Sprachaufnahmen. Hintergrund ist, dass sehr kurz, aber sehr laute Pegelspitzen weit über dem Durchschnittspegel des Gesamtsignals den Maximalwert des Signals unnötig begrenzen und die Signalenergie so gering halten. Da die Testsignale, die in dieser Arbeit verwendet werden, aus inhomgenen Quellen stammen und sehr unterschiedliche Lautstärken haben, wird so gewährleistet, dass sie zumindest ähnliche Energien haben. An dieser Stelle werden (noch) keine Frequenanteile herausgefiltert, um keine Frequenzen zu verlieren, die in den späteren Verareitungsschritten wieder Voice-Activity-Detection 4.3 oder der Feature-Extraction eventuell noch benötigt werden.

Die Dynamikeinschränkung wird mit Hilfe eines Audiokompressor umgesetzt. Ein Audiokompressor verringert Signalspitzen, die über einen festgelegten *Schwellwert (Threshold)* liegen, um ein festgelegtes *Verhältnis (Ratio)*. Ein Threshold von 0.3 mit Ratio von 0.5 bedeutet beispielsweise, dass alle Signalspitzen, die den Wert 0.3 überschreiten oder -0.3 unterschreiten, um 50% verringert werden. Ein Kompressor kann auf die Überschreitung des Thresholds erst nach einer als *Attack* bezeichneten Verzögerung reagieren, und bei erneuten Verlassen des Thresholdes mit einer als *Release* bezeichneten Verzögerung nachwirken. Signalspitzen werden so verringert und die Lautstärke-Dynammik eingeschränkt. Die tatsächliche Erhöhung der Signalenergie geschieht im Anschluss durch die Anhebung der insgesamten Signallautstärke, wie Beispielsweise der Normalisierung des Signals auf den Maximalpegel. Abbildung 4.2 zeigt die Parameter eines solchen Audio-Kompressors.

Der entwickelte Kompressor automatisiert die Einstellung von Threshold und Ratio auf Grundlage des Root-Mean-Square (RMS) des Signales x der Länge N. Der RMS-Wert ist ein Maß für die durchschnittliche Signalenergie und wird wie nach Formel 4.1 berechnet. Threshold und Ratio werden nach den Formeln 4.2 und 4.3 berechnet, wobei der Parameter r_a den Ziel-RMS-Wert anbgibt und mit dem Wert.

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x[n]^2} \tag{4.1}$$

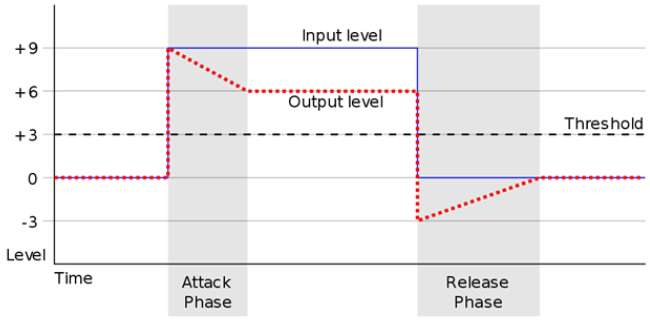

Abbildung 4.2: Parameter eines Audio-Kompressors

$$\mathrm{THold}(x) = \left[\frac{\mathrm{RMS}(x[])}{r_a}\right]^2 \tag{4.2}$$

$$\mathrm{Ratio}(x) = \left[\frac{\mathrm{RMS}(x[])}{r_a}\right]^2 \tag{4.3}$$

Abbildung 4.3 zeigt das ein Signal vor und nach dem Preprocessings. Zu sehen ist, dass die Lautstärke der einzelnen Schrei-Einheiten nach der Anpassung einheitlicher ist.

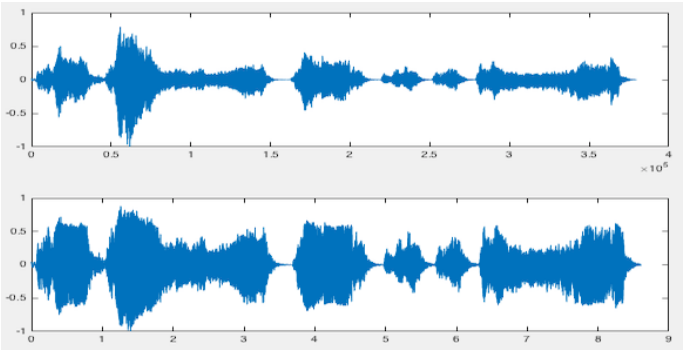

Abbildung 4.3: Ergebnis des Preprocessings

4.3 Voice Activity Detection

Das Ziel ist, in einem Audiosignal diejenigen Stellen zu markieren, in denen Schreigeräusche vorliegen. Abbilung 4.4 visualisiert ein Beispiel für eine solche Markierung: Zu sehen ist der Zeitbereich eines Audiosignales mit drei klar erkennbaren Cry-Units. Die Rote Linie, die das Signal überspannt, bildet die Zeiteinheiten des Eingangssignales in die binären Kategorien *Stimmhaft* und *Stille* ab.

Das zu lösende Probem wird als *Voice Activity Detection (VAD)* oder auch *Speech Detection* bezeichnet. Das Ziel ist die Unterscheidung von denjenigen Zeiträumen im Signal, in denen Stimme enthalten ist, von denen, in denen keine Stimme enthalten ist. Zeiträume ohne Stimme werden von nun als „Stille“, und Zeiträume mit Stimme als „Stimmhaft“ bezeichnet. Die größte Herausforderung ist für VAD-Systeme die robuste Erkennung bei Signalen mit Rauschen unbekannter Stärke und Natur.[17]

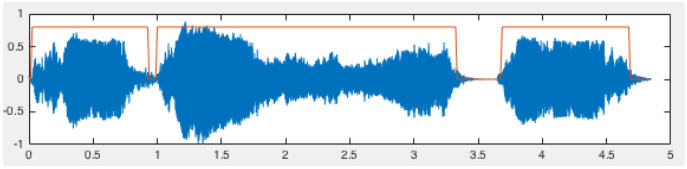

Abbildung 4.4: Markierung von Schreigeräuschen im Audiosignal

Der Grundlegende Aufbau eines VAD-Algorithmus ist wie folgt:

1. **Windowing**: Unterteilung des Signals in (einander überlappende) Fenster, typischerweise zweistellingen Millisekunden-Bereich.
2. **Featureextraction** aus den einzelnen Fenstern
3. **Thresholding / Entscheidung** über Präsens oder Nicht-Präsens von Stimme für jedes Zeitfenster auf Grundlage der Features mit Hilfe von Entscheidungsregeln wie Grenzwerten.
4. **Decision-Smoothing**, das nachträgliche Hinzufügen oder Entfernen von Entscheidungen mit Hilfe von kontextuellen Informationen der umliegenden Entschiedungen.[17]

Der an dieser Stelle entwickelte Ansatz ist eine Kombination aus den Ideen, die in [24], [39], [20], [37] und [18] vorgestellt wurden.

4.3.1 Windowing

Das Signal x wird in Fenster $x_1 ... x_n$ unterteilt. Dieser Process wird als „Windowing“ bezeichnet. Es wurde sich für die in [20] vorgestellte Fensterlänge von 25 ms entschieden, als Kompromiss zwischen den in [24] empfohlenen 10 ms und den in [37] empfohlenen 40 ms Die einzelnen Fenster überlappen einander um 50%, das heisst 12.5 ms.

4.3.2 Feature Extraction

Für jedes Signalfenster $x_1 ... x_n$ à 25 ms werden die folgenden Features aus den Kategorien **Zeitbereich**, **Frequenzbereich**, **Cesptrum** und **Autokorrelation** berechnet.

Zeit-Bereich

Im **Zeit-Bereich** werden die beiden Features *Root-Mean-Square*-Wert [RMS] und *Zero-Crossing-Rate* [ZCR] berechnet.

In [24] wird die Energy eines Zeitfensters für die Voice Activity Detection berechnet. Um den errechneten Wert besser ins Verhältnis zum Signalpegel setzen zu können, wird anstelle der Energie der RMS-Wert nach Formel 4.1 verwendet. Ein höherer RMS-Wert weist auf das Vorhandensein von Stimme im Signalfenster hin, da diese typischerweise eine höhere Energie als das Hintergrundrauschen hat. Als zweites Feature des Zeitbereiches wird die in [37] verwendete *Zero-Crossing-Rate* (ZCR) berechnet. Dabei wird der relative Anteil an Vorzeichenwechseln des Signalfensters x_i nach Formel 4.4 berechnet. Eine höhere ZCR

weist auf Stille hin, da Rauschen typischerweise einen höheren ZCR hat als periodische Signale mit tieferer Grundfrequenz.

$$\mathrm{ZCR}(x_i[]) = \sum_{1}^{N-1} |\mathrm{sng}(x_i[n]) - \mathrm{sng}(x_i[n-1])| \tag{4.4}$$

Abbildung 4.5 visualisiert ein Beispielsignal mit Babygeschrei mit einem Hintergrundrauschen mit einem Signal/Rausch-Abstand von 25 25 dB. Bei dem Signal wurden die stimmhaften Segmente magenta markiert. Für jedes Signalfenster wurden der RMS-Wert und der ZCR-Wert berechnet und als Signale mit im Graphen dargestellt, in dem für den Anfangszeitpunkt eines Zeitfenster der jeweilige berechnete Feature-Wert abgetragen wird. Die beiden Signale wurden so skaliert, dass ihr Maximalwert 1 nicht überschreitet, um ihr Verhalten bezüglich des Vorhandenseins von Stimme klarer erkennbar zu machen.

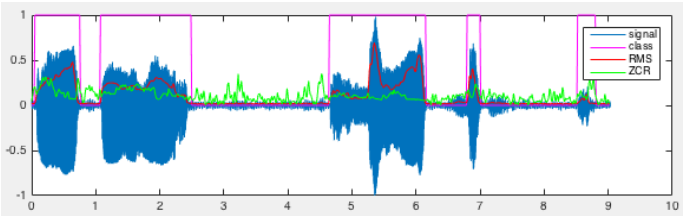

Abbildung 4.5: Features des Zeitbereiches

Frequenz-Bereich

Aus dem **Frequenz-Bereich** werden die drei Features *unnormalisierte spektrale Entropie* $[SEnt_u]$, *normalisierte spektrale Entropie* $[SEnt_n]$ und *dominanteste frequenzkomponenten* $[f_{Dom}]$ berechnet.

Als Vorbereitungsschritt wurde die *Kurzzeit-Fourier-Transformation* (engl. *Short-Time-Fourier-Transform* STFT) nach dem Algorithmus von [38] implementiert. So wird jedes Signalfenster x_i mit einem 25 ms-Hamming-Window multipliziert und daraufhin mit einer 2048-Punkte langen *schnellen Fouriertransformation* (FFT) in den Frequenzbereich $F\{x_i\}$ transformiert. Weiterhin wurde für jedes Frequenz-Fenster des Magnitudensignal $X_i = |F\{x_i\}|$ berechnet und das Phasensignal jedes Zeitfensters verworfen, da für keines der Features Phaseninformationen von Interesse waren. Das Ergebnis dieser Transformation aus den Signalfenstern in Zeitbereich $x_1 ... x_n$ sind die Frequenzfenster $X_1 ... X_n$.

In [39] wird die *spektrale Entropie* für die Voice-Activity-Detection berechnet. Dabei wird das Spektrum des Frequenzfensters X_i als Wahrscheinlichkeitsverteilung betrachtet. Die *normalisierte spektrale Entropie* wird nach der Formel 4.6 berechnet. Der Wert px_i ergibt sich durch die Normalisierung des N-Punkte langen Spektrums nach Formel 4.5. Neben der in [39] vorgestellten normalisierten spektralen Entropie wird zusätzlich die *unnormalisierte Spektrale Entropie* nach Formel 4.7 berechnet. Bei dieser wird das Spektrum nicht normalisiert, das heißt, es gilt $px_i[f] = X_i[f]$. Somit hat die Energie des Signals einen größeren Einfluss auf den berechneten Wert. Bei der unnormalisierten spektralen Entropie ist zu erwarten, dass Signalfenster mit Stimme eine höherer Spektrale Entropie haben als Fenster mit Stille. Bei der normalisierten spektralen Entropie ist zu erwarten, dass Fenster mit Stille eine höhere Entropie haben als Fenster mit Stimme.

In die Berechnungen wurden nur Frequenzen von 250 - 8000 Hz betrachtet, da nach [33] die Grundfrequenz der Stimme eines Babies mindestens 250 Hz beträgt, und nach [18] Stimme keine wichtigen Informationen oberhalb von 8000 Hz überträgt.

$$px_i[f] = \frac{X_i[f]}{\sum_{k=1}^{N} X_i[k]} \tag{4.5}$$

$$\mathrm{SEnt}_n(X_i) = -\sum_{k=1}^{N} px_i[f] \cdot \log(px_i[k]) \tag{4.6}$$

$$\mathrm{SEnt}_u(X_i) = -\sum_{k=1}^{N} X_i[k] \cdot \log(X_i[k]) \tag{4.7}$$

Weiterhin wird die in [24] vorgestellte *dominanteste Frequenzkomponente* berechnet. Für jedes Frequenzfenster X_i wird diejenige Frequenz nach Formel 4.8 berechnet, welches die höchste Magnitude hat. Ein stimmhaftes Signal hat typischerweise eine höhere f_{Dom} als ein nicht stimmhaftes Signal.

$$f_{Dom}(X_i) = \arg\max_{f}\{X_i[f]\} \tag{4.8}$$

Abbildung 4.6 visualisiert diese Features für das selbe Eingangssignal aus 4.5. Die Features wurden wie bei Abbildung 4.5 beschrieben für die Darstellung skaliert

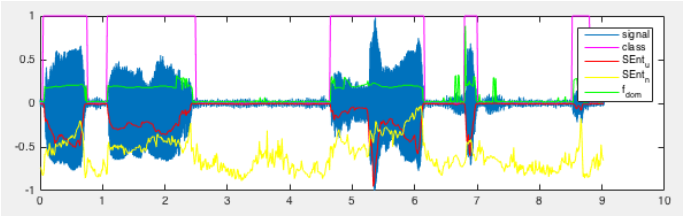

Abbildung 4.6: Features des Frequenz-Bereiches

Cepstrum

Das **Cepstrum** wird definiert als die inverse Fourier-Transformation des Logarithmus des Magnituden-Signals nach Formel 4.9.

$$c[n] = F^{-1}\{\log|F\{x[n]\}|\} \tag{4.9}$$

Die Idee wird anhand von Abbildung 4.7 verdeutlicht. Zu sehen ist ein typisches absolutes Spektrum eines Signalfensters mit Stimme $|F\{x\}|$. Durch die Logarithmisierung des Spektrums wird die Dynamik der Amplitudenunterschiede eingegrenzt. Das entstandene Signal $\log|F\{x[n]\}|\}$ zeigt klar die Lage der Formanten, dargestellt durch die durchgezogene, schwarze Linie. Die vielen, kleinen Signalspitzen, die sich entlang der Formantenlinie bewegen, sind die harmonsichen Obertöne, die von dem periodischen Signal der Stimmbänder

erzeugt wurden. Der Grundgedanke ist, dieses Spektrum wiederum als Signal zu interpretieren. Wäre dieses Signal im Zeitbereich (obwohl es eigentlich den Frequenzbereich darstellt), könnte es durch die Amplituden-Modulation eines Quasi-periodischen Signales entstanden. Das höherfrequente Signal (Spektrum der Stimmbänder) wäre dementsprechend der Träger, und das niederfrequente Signal (Spektrum der Formanten) der Modulator.Um diese beiden Signalanteile voneinander zu trennen, würde man eine weitere Fourier-Transformation implementieren (welche an dieser Stelle mathematisch betrachtet einer inversen Fourier-Transformation gleichkommt, da das Phasensignal entfernt wurde). Man erwartet in dem so entstehenden Spektrum $F^{-1}\{\log|F\{x[n]\}|\}$ eine Spitze im niederfrequenten Bereich, bedingt durch den niederfrequenten Modulator, und eine Spitze im höheren Frequenzbereich, bedingt durch den höherfrequenten Träger.

Der Bereich dieser „Fouriertransformation der Foueriertransformation“ wird als *Cepstrum* bezeichnet. Cepstrum ist ein ein Wortspiel, welches durch die Umkehrung der ersten vier Buchstaben des Wortes SSpectrumënsteht. Die Unabhängige Variable des Cepstrum folgt dem Wortspiel und wird als *Quefrency* bezeichnet. Damit wird verdeutlicht, dass die unabhängige Variable des Cepstrum zwar mathematisch betrachtet die Zeit darstellt, jedoch als Frequenz interpretiert wird.

Insbesondere das Vorhandensein einer Spitze im Quefrency-Bereich ab 3 ms im Cepstrums weist auf das Vorhandensein von harmonischen Obertönen eines periodischen Eingangssignal hin, wie es von Stimmbändern erzeugt wird. Eine Grundfrequenz von f_0 dieser periodischen Schwingung entspricht einer Signalspitze bei der Quefrency von $\frac{1}{f_0}$. Abbildung 4.8 zeigt das Log-Spektrum und das Cepstrum eines Sprachsignales, welches in Fenster à 50 ms zerlegt wurde. Die Frames 1-7 sind stimmlos, die Frames 8-15 stimmhaft. Die Stimmhaftigkeit spiegelt sich klar im auftauchen einer Signalspitze bei einer Quefrency von 10 ms wieder.[11]

Die Informationen des Cepstrum eigenen sich ebenfalls zur Voice-Activity-Detection. Es werden die Features *Upper Cepstrum Peak* [$Ceps_{mag}$] und *Upper Cepstrum Peak Location* [$Ceps_{loc}$] berechnet.

Wie in [37] und [39] beschrieben, wurde die *höchste Magnitude im oberen Quefrency-Bereich* (Upper Cepstrum Peak) nach Formel 4.10 berechnet. Nach [33] liegt die Grundfrequenz eines Kinderweinens zwischen 250 und 2000 Hz, was einem Quefrency-Bereich von 5 - 40 ms entspricht. Folglich werden bei der Berechnung nach Formel 4.10 nur Quefrency-Werte in diesem Bereich betrachtet werden. Eine hoher $Ceps_{mag}$-Wert weist auf das Vorhandensein von Stimme für das aus dem Fenster x_i Berechneten Cepstrum x_i hin. Als zweites Features des Cepstrums wird die Quefrency der so gefundenen höchsten Magnitude (Upper Cepstrum Peak Location) nach Formel 4.11 berechnet. Bei Signalfenstern mit Stille ist es wahrscheinlicher, dass sich die Spitze am Mindest- oder Maximum-Wert des Cepstrum befindet.

$$Ceps_{mag}(c_i) = \max_k\{\text{mag}(c[k])\} \tag{4.10}$$

$$Ceps_{loc}(c_i) = \arg\max_k\{\text{mag}(c[k])\} \tag{4.11}$$

Abbildung 4.9 visualisiert die beiden Features des Cepstrums auf die selbe Art und Weise, wie es bei Abbildung 4.5

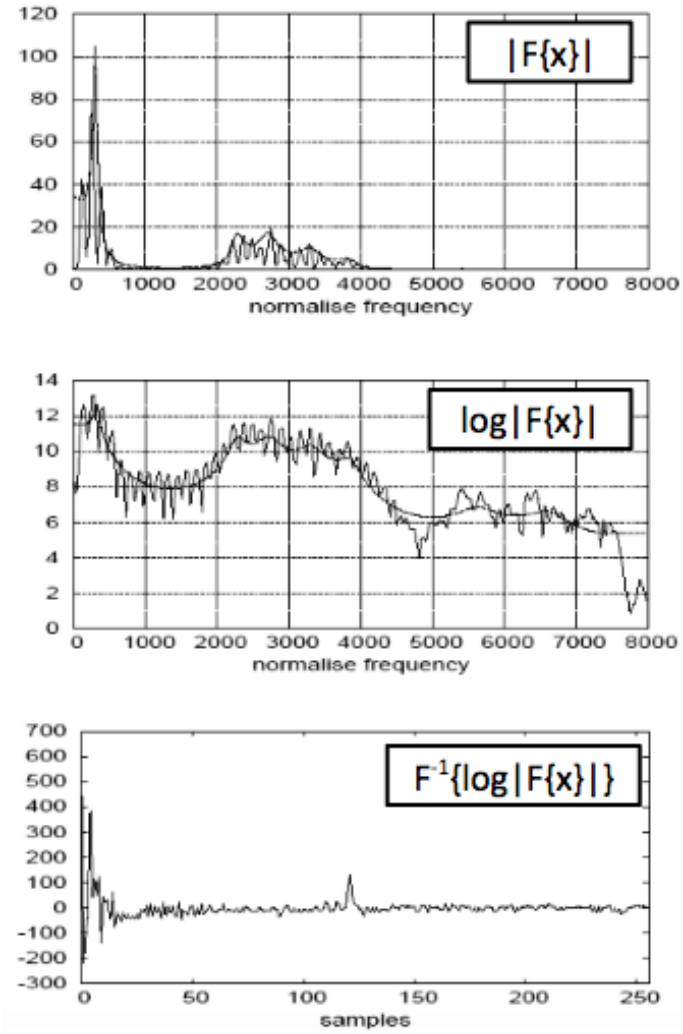

Abbildung 4.7: Entstehung des Cepstrums

Autokorrelation

Die vierte Methode, die zur Voice-Activity-Detection verwendet wird, ist die **Autokorrelation**. Dabei wird das Signalfenster x_i mit einer verzögerten Variante von sich selber nach Formel 4.12 korreliert. Der Parameter k wird als Lag bezeichnet und definiert die Verzögerung des Signals, mit dem die Korrelation durchgeführt wird. Ein hoher Autokorrelationswert bei einem bestimmten Lag k weist darauf hin, dass das Signal eine Periodizität bezüglich diesem Verzögerungswert aufweist. Möchte man beispielsweise berechnen, ob ein bei 16 000 Hz gesampeltes Signal bezüglich 400 Hz periodisch ist, so wird die Autokorrelation mit einem Lag von $k = \frac{\text{Sampling-}f_s}{\text{Detect-}f_s} = \frac{16\,000\,\text{Hz}}{400\,\text{Hz}} = 40$ durchgeführt. Formel 4.13 zeigt die *normalisierte Autokorrelation*, bei der das Autokorrelations-Signal sowohl bezüglich der Signalenergie als auch der Anzahl der Lags normalisiert wird.

$$a_i[k] = \sum_{n=k}^{N} x_i[n] \cdot x_i[n-k] \tag{4.12}$$

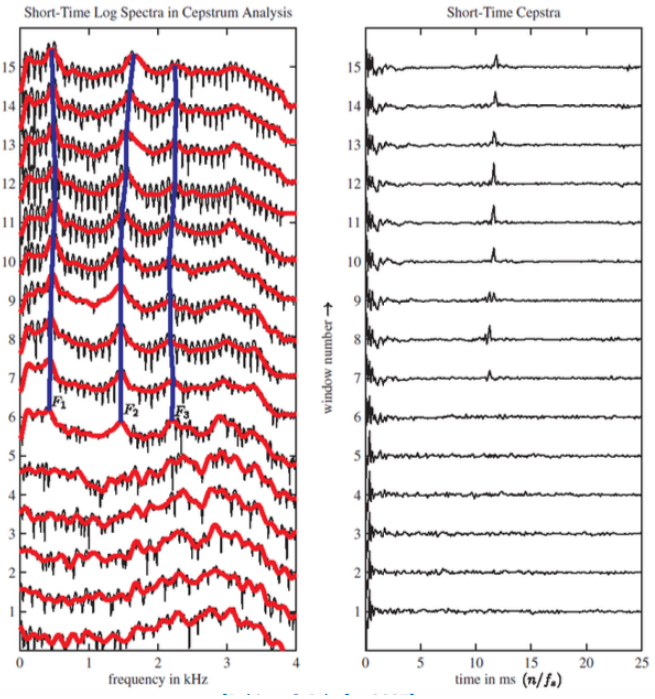

Abbildung 4.8: Cepstrum

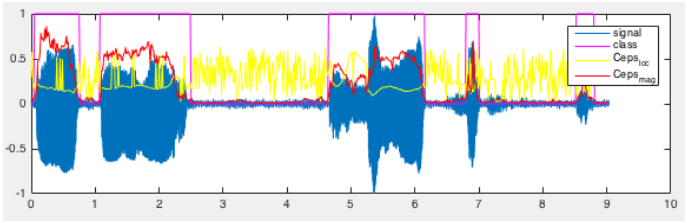

Abbildung 4.9: Features des Cepstrums

$$acorr_i[k] = \frac{\sum_{n=k}^{N} x_i[n] \cdot x_i[n-k]}{\sqrt{(\sum_{n=1}^{N-k} x_i[n]^2)} \cdot \sqrt{(\sum_{n=k}^{N} x_i[n]^2)}} \tag{4.13}$$

Abbildung 4.10 zeigt das $acorr_i$-Signal eines Sprachsignals mit den Lags 50 - 320. Die höchste Signalmagnitude bei Lag = 145 ist ein Indikator dafür, dass bei einer Samplingrate von 16 000 Hz die Grundfrequenz $\frac{1600\,\text{Hz}}{145} = 110.34\,\text{Hz}$ am dominantesten im Signal enthalten ist, was bei einem Stimmsignal die Grundfrequenz f_0 ist.

Da Stimmsignale eine höhere Periodizität aufweisen als das Hintergrundrauschen, eignet sich die Autokorrelation zur Voice-Activity-Detection. Es werden die Features *Maximum Autocorrelation Peak* [*aMax*] und (*Autocorrelation Peak Count*) [*aCount*] berechnet.

Beide Features werden in [17] zur VAD verwendet. Die *höchste Magnitude der Autokorrelation* (*Maximum Autocorrelation Peak*) wird nach der Formel 4.14 definiert. Eine höherer *aMax*-Wert weist auf das Vorhandensein von Periodizität im Signal und somit auf Stimme hin. Das zweite Feature ist die *Anzahl an Autokorrelatoins-Spitzen* nach Formel 4.15. Da

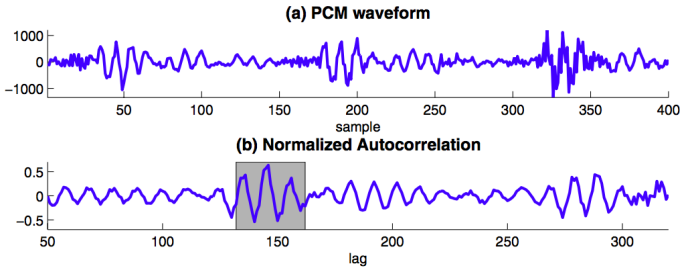

Abbildung 4.10: Autokorrelation eines Signals

nach [33] die Grundfrequenz eines Kindergeschreis zwischen 250 und 2000 Hz liegt, wurden auch nur Lags für diesen Bereich verwendet.

$$\text{aMax}(x_i) = \max_k \{\text{mag}(acorr_i[k])\} \tag{4.14}$$

$$\text{aCount}(x_i) = \text{count}_k \{\text{mag}(acorr_i[k])\} \tag{4.15}$$

Abbildung 4.11 visualisiert die Features der Autokorrelation auf die selbe Art und Weise wie bei Abbildung 4.5.

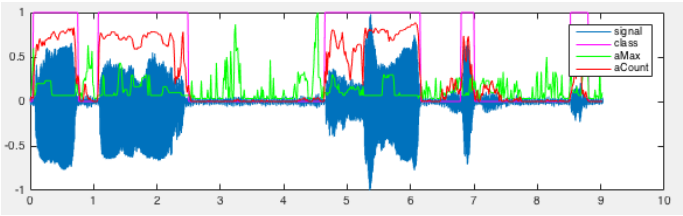

Abbildung 4.11: Features der Autokorrelation

Konstruktion des Feature-Vektors

Für jedes Signalfenster à 25 ms werden die 9 vorgestellten Features *RMS, ZCR,* $SEnt_u$, $SEnt_n$, f_{Dom}, $Ceps_{mag}$, $Ceps_{loc}$, *aMax* und *aCount* berechnet. Wie beschrieben, sollten Signalfenster mit Stimme einen höheren Wert des jeweiligen Features erzeugen als Signalfenster mit Stille (oder, abhängig vom Feature wie der ZCR, einen tieferen).

Abbildung 4.12 zeigt in (a) die RMS-Werte eines Signals mit einem Signal-Rausch-Abstand (SNR) von 50 dB. Die Zeiträume mit Stille haben einen weitaus niedrigeren RMS-Wert als die Signalteile mit Stimme. In (b) ist das selbe Signal mit einem Signal-Rausch-Abstand von 3 dB zu sehen. Nun liegen die RMS-Werte des Rauschens nur noch knapp unter denen des Sprachsignals. Zu sehen ist, dass starkes Hintergrundrauschen ähnlich hohe Feature-Werte erzeugen kann wie die Stimme.

In [24] und [20] wird die Idee präsentiert, den Wert des jeweiligen Features zu messen, der in den stimmlosen Segmenten durch das Hintergrundrauschen erzeugt wird. Jedoch ist für ein Signalfenster x_i zum Zeitpunkt der Berechnung des Features noch nicht bekannt, ob es sich um ein Zeitfenster mit Stille oder Stimme handelt. Genau diese Entscheidung wird

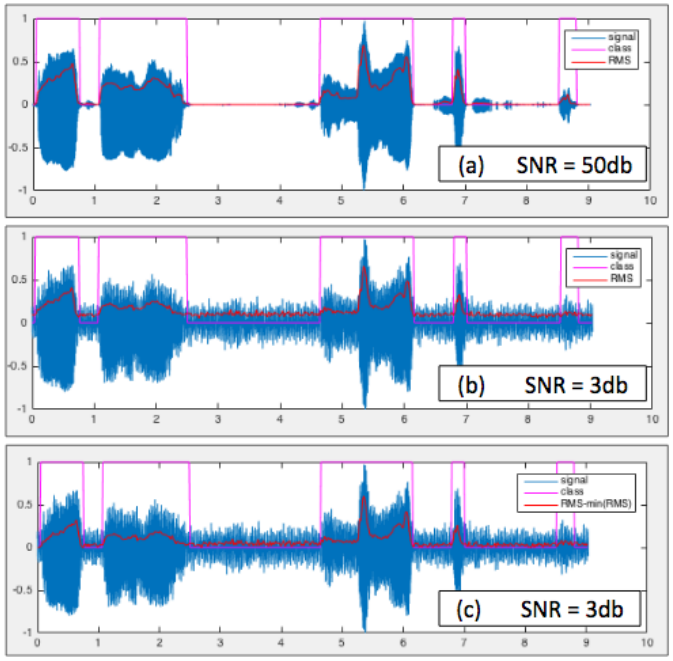

Abbildung 4.12: Differenz-Signal des RMS-Wertes

später auf Basis der Features getroffen. Man kann jedoch davon ausgehen, dass kein Baby-Schrei länger eine bestimmte Zeitraum t_{max} dauert, bevor das Babie Luft holen muss und somit ein Zeitfenster mit Stille entsteht. Der Mindestwert eines Features in einem Zeitraum t_{max} ist somit der Wert des Features, der mindestens durch das Hintergrundrauschen erzeugt wird. Er wird nach Formel berechnet, wobei t_{xi} die Länge eines Signalfensters x_i ist (in diesem Fall 25 ms). Das *Differenzfeature* wird definiert nach Formel 4.17 als die Differenz des für das aktuelle Signalfenster berechneten Features und dem Mindestwert dieses Feature der letzten t_{max} Sekunden.

$$\text{MinF}(Feat(x_i)) = \min_{k=i-z\ldots i}(\text{Feat}(x_k)), z = \frac{2t_{max}}{t_{xi}} \tag{4.16}$$

$$\text{DiffF}(Feat(x_i)) = Feat(x_i) - MinF(Feat(x_i)) \tag{4.17}$$

Der Featurevektor, der schlussendlich für jedes Signalfesnter $x_i[]$ berechnet wurde, besteht neben den 9 vorgestellten Features, zusätzlich aus den Differenzfeatures für einen Zeitraum von $t_{max} = 4$ Sekunden. Features, deren Wert bei Stimmhaften Signalfenstern geringer ist als bei stimmlosen, werden an der x-Achse gespiegelt, um Formel 4.16 anwenden zu können (das Betrifft die Features ZCR, SEnt_u und aCount). Das einzige Feature, welches nicht als Differenzfeature dem Feautrevektor beigefügt wurde, ist der *Upper Cepstral Peak Location*-Feature $[Ceps_{loc}]$, da es bei Stille sowohl einen höheren als auch einen niedrigeren Wert annehmen kann. Der Feature-Vektor V des Signalfensters x_i wird nach Formel 4.18

berechnet umfasst 17 Features, wobei 9 absolute Features und 8 Differenzfeatures verwendet werden.

$$V(x_i) = \Big(\text{RMS}(x_i), ..., \text{aCount}(x_i) \\ \text{DiffF}(\text{RMS}(x_i))....\text{DiffF}(-\text{aCount}(x_i))\Big) \tag{4.18}$$

4.3.3 Thresholding

Finden der Grenzwerte

Für jedes Signalfenster $x_1...x_n$ liegt nun ein Featurevektor $v_1...v_n$ vor. Das Ziel ist nun, Grenzwerte für die Features zu finden, bei deren Überschreitung das Signalfenster als *Stimmhaft* kategorisiert wird. Abbildung 4.13 verdeutlicht das Prinzip für das Feature $Ceps_{mag}$. Eine binäre Kategorisierung nach dem Muster $C(x_i) = \{1, \text{wenn } \text{Ceps}_{mag}(x_i) \geq 0.2, 0 \text{ sonst}\}$ würde auf den ersten Blick eine weitgehend richtige Kategorisierung vornehmen.

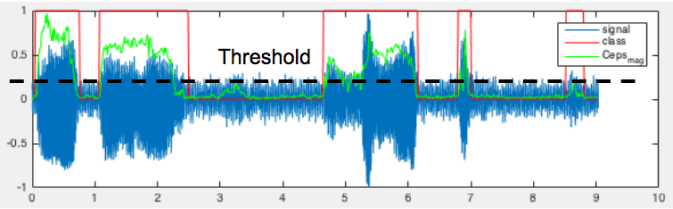

Abbildung 4.13: Thresholding eines Feature-Signales

Es sind Kombinationen von Grenzwerten in Form von Entscheidungsbäumen denkbar, wenn mehrere Features in die Kategorisierung einfließen. Ein Beispiel wird in Listing 4.1 dargestellt, bei dem die Kategorsierung hierarchisch zuerst nach einem Grenzwert für Ceps_{mag} und danach für den RMS-Wert entschieden wird.

Listing 4.1: Beispiel eines CART-Entscheidungsbaums

```
if Ceps_mag(x_i) > 0.2
|    if RMS(x_i) < 0.13
|    |    C(x_i) = 0
|    |else
|    |    C(x_i) = 1
|else
|     C(x_i) = 1
```

Zur Festlegung der Grenzwerte wird ein *Classification And Regression Tree* (CART)-Algorithmus verwendet. CART-Algorithmen sind Klassifizierungsalgorithmen, bei denen Entscheidungsbäume wie in Listing 4.1 konstruiert werden. Sie haben den Vorteil, dass die Klassifizierung in Form von Regeln mit Grenzwerten dargestellt werden können, die für den Menschen nachvollziehbar sind (im Gegensatz zu z.B. Neuronalen Netzen). Der durch den CART-Algorithmus konstruierte Entscheidungsbaum wird auch als *Klassifizierungs-Modell* bezeichnet.[5]

Einer der bekanntesten CART-Algorithmen ist der ID3-Algorithmus, entwickelt von J. Ross Quinlan an der University of Sidney. Als Trainingsdatensatz S wird eine Menge an

Instanzen verwendet, deren Klassenzugehörigkeit bereits bekannt ist. Der ID-3 Algorithmus funktioniert nur für diskrete Features und Klassen. Ein Beispiel für ein diskretes Feauture ist [*Lautstärke = {laut, leise}*]. Ein Beispiel für ein numerisches Feature, welches vom ID-3 Algorithmus nicht akzeptiert wird, ist [*Lautstärke* = $\mathbb{N}$]. Der Algorithmus funktioniert folgendermaßen:

1. Gehören alle Instanzen des Datensatzes S nur einer Klasse an?
   a) Ja: Markiere diesen Knoten als Blatt und STOP.
   b) Nein: Erzeuge einen neuen Knoten. Wähle das Feature F, welches den höchsten *Informationsgewinn* bringt. Das heisst, dass dieses Feature die stärkste Unterscheidung zwischen den einzelnen Klassen ermöglicht. Der Informationsgewinn $H(S)$ eines Features F mit den Untermengen X, welche aus den Instanzen der selben Klasse bestehen, wird mit Hilfe der Entropie nach Formel 4.19 berechnet.

$$H(S) = -\sum_{x \in X} F(x) \log_2 F(x) \tag{4.19}$$

2. Unterteilung den Trainingsdatensatz S in die Untermengen $S_1 ... S_n$, wobei eine Untermenge für jeden möglichen Attributbwert des ausgewählten Features gebildet wird.
3. Wiederhole die Schritte 1. und 2. für alle Untermengen $S_1 ... S_n$ [5]

Der C.45-Algorithmus ist eine Erweiterung des ID-3 Algorithmus, der neben anderen Erweiterungen ebenfalls mit numerischen Attributwerten umgehen kann. Dafür werden für ein Feature alle Instanzen des Datensatzes S nach den Werten dieses Features geordnet und derjenige Grenzwert gesucht, der den höchsten Informationsgewinn bringt. Der Datensatz wird daraufhin an den Knoten in genau 2 Untermengen aufgespalten.[15]

Zum Finden der Grenzwerte der in Kapitel 4.3.2 beschriebenen Features wurde die Implementierung des C.45-Algorithmus *REPTree*[1] der Open Source Data-Mining-Bibliothek *Weka*[2] verwendet. Diese hat den Vorteil, dass die Tiefe des Entscheidungsbaumes festlegbar ist und somit die Komplexität des Baumes begrenzt werden kann. Ein Nebeneffekt ist, dass so Overfitting, also die Überanpassung des Entscheidungsbaumes auf den Trainingsdatensatz, vermieden wird.

Trainings- und Testdatensätze

Zum Training des REPTree-Algorithmus musste in Trainingsdatensatz S erstellt werden. Dazu wird zunächst eine Menge an Audiosignalen benötigt. Es wurden 6 Audiosignale mit Weinen von Babies von der freien Online-Sound-Bibliothek `https://www.freesound.org/` heruntergeladen und zu Segmenten à 10 Sekunden beschnitten. Diese Audiosignale sind weitestgehend Rausch-frei. Die Segmente der Audiosignale wurden händisch Kategorisiert in die Klassen {1 = Stimme, 0 = Still}. Weiterhin wurden 3 verschiedene Rauschsignale heruntergeladen. Es handelt sich um "realistische"Rauschsignale mit Krankenhausatmosphären. Jedes dieser 3 Rauschsignale wurde mit den 6 Weinsignalen überlagert, einmal mit

[1] Dokumentation von REPTree: `http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/REPTree.html`
[2] Download von WEKA: `http://www.cs.waikato.ac.nz/ml/weka/`

einem Signal-Rausch-Abstand von 50 dB (fast unhörbares Rauschen), und einmal mit einem Signal-Rausch-Abstand von 3 dB (sehr starkes Rauschen). Außerdem wurde ein Testsignal erzeugt, welches eine siebte Tonaufnahme eines Kinderweinens enthält, dass mit einem vierten Rauschsignal mit einem SNR von 7 dB überlagert wurde. Dieses Signal spielt eine Sonderrolle, da es nur zur Verifikation verwendet wird und enthält daher nur ein Audiosignal (Siehe Kapitel 4.3.3) So wurden vier Mengen an Audiosignalen erzeugt:

$A_{50\,\mathrm{dB}}$ enthält $3 \cdot 6 = 18$ Audiosignale, bei dem alle 6 Wein-Signale mit den 3 Rauschsignalen bei einem Signal-Rausch-Abstand von 50 dB überlagert wurden

$A_{3\,\mathrm{dB}}$ enthält $3 \cdot 6 = 18$ Audiosignale, bei dem alle 6 Wein-Signale mit den 3 Rauschsignalen bei einem Signal-Rausch-Abstand von 3 dB überlagert wurden

$A_{50+3\,\mathrm{dB}}$ $= \{A_{50\,\mathrm{dB}} \cup A_{3\,\mathrm{dB}}\} = 32$ Audiosignale

$A_{7\,\mathrm{dB}*}$ enthält 1 Audiosignal, bei dem ein siebstes Wein-Signale mit einem vierten Rauschsignal bei einem Signal-Rausch-Abstand von 7 dB überlagert wurde.

Im nächsten Schritt werden die eigentlichen Trainingsdatensätze $S_{SNR,Feat}$ gebildet, in dem Audiosignale dieser Signalmengen (1) wie in Kapitel 4.2 vorverarbeitet werden, (2) wie in Kapitel 4.3.1 in die Signalfenster à 25 ms zerlegt werden und (3) für jedes Audiosignal die durch Gleichung 4.18 definierte Featurevektoren berechnet werden. Außerdem wird jedem Featurevektor die Klasseninformation *Stimme/Stille* zum Training des REPTree-Algorithmus mitgegeben.

Es ist rechnerisch zu Aufwendig, alle genannten Features in einem kontinuierlichen System zur Voice Activity Detection zu berechnen. Daher werden Untermengen der Features in den Datensätzen gebildet. Das Ziel ist es, diejenige Untermenge an Features zu finden, die sich am besten für die Voice-Activity-Detection sowohl bei niedrigem als auch bei starkem Hintergrundrauschen eignet. Die Untermengen werden in Bezug auf die Methode gebildet, durch die die Features berechnet werden. Das heißt, dass beispielsweise die Untermenge *Zeit* die in Kapitel 4.3.2 beschriebenen Features *RMS* und *ZCR* sowie die dazugehören Differenzfeatures *FDiff(RMS)* und *FDiff(ZCR)* beinhaltet.

Die 9 Untermengen sind: { Zeitbereich, Frequenzbereich, Cepstrum, Autokorrelation, Zeit + Frequenzbereich, Zeit + Cepstrum, Zeit + Autokorrelation, Frequenz + Cepstrum, Frequenz + Autokorrelation }.

So enthält beispielsweise der Datensatz $S_{3\,\mathrm{dB},Zeit}$ die Featurevektoren des Zeitbereiches für die Audiosignale mit einem Signal-Rausch-Abstand von 325 dB. Die Audiosignal-Mengen [$A_{50\,\mathrm{dB}}$], [$A_{3\,\mathrm{dB}}$] [$A_{50+3\,\mathrm{dB}}$] und [$A_{7\,\mathrm{dB}*}$] wurden in Datensätze umgewandelt. Es werden schlussendlich $4 \cdot 9 = 36$ Datensätze gebildet.

Training und Ergebnis

Der REPTree-Algorithmus entwirft einen Entscheidungsbaum, der für den angegebenen Trainingsdatensatz eine möglichst hohe Accuracy gewährleistet (unter den gegebenen Einschränkungen eines CART-Algoirthmus). Wird dem REPTree-Algorithmus beispielsweise der Datensatz $S_{3\,\mathrm{dB},Zeit}$ (also der Datensatz mit einem SNR von 3 dB unter Verwendung der Zeit-Features, siehe Kapitel 4.3.3) als Input gegeben, entwirft der Algorithmus einen Entscheidungsbaum auf Basis dieses Datensatzes. Wird das gebildete Modell daraufhin für die

Klassifikation der $A_{3\,\mathrm{dB}}$-Signalmenge verwendet, kann man aus dem Klassifikationsergebnis die Accuracy des Modells für diesen Signal-Rausch-Abstand berechnen.

Der Datensatz $S_{SNR,Feat}$, der als Input für den REPTree-Algorithmus und somit zur Bildung des Entscheidungsbaums verwendet wird, wird als *Trainings-Datensatz* bezeichnet. Die Signalmenge A_{SNR}, auf den das Modell angewandt wird und für den die Accuracy berechnet wird, wird als *Test-Datensatz* bezeichnet. Wird also beispielsweise der Trainings-Datensatz $S_{50\,\mathrm{dB},Zeit}$ und als Test-Signalmenge $A_{3\,\mathrm{dB}}$ verwendet, so kann man berechnen, wie gut sich ein Modell unter Verwendung der Zeit-Features zur Klassifizierung niedriger SNRs eignet, obwohl es für hohe SNRs entworfen wurde.

Das Ziel ist, den Entscheidungsbaum für eine Feature-Untermenge zu finden, die eine möglichst hohe Klassifikations-Accuracy für sowohl hohe als auch niedrige SNR gewährleistet. Die Frage ist, ob ein Entschiedungsbaum, der für einen niedrigen SNR gebildet wird, auch für hohe SNR gut funktioniert, oder ob das Gegenteil zutreffend ist. Daher werden die Entscheidungsbäume sowohl auf Basis verschiedener SNRs als auch verschiedener Feature-Untermengen gebildet. Die Modelle werden daraufhin gegen die Signale mit den hohen und niedrigen SNRs getestet.

Es wurden, wie in Kapitel 4.3.3 beschrieben, $3 \cdot 9 = 27$ Trainings-Datensätze erzeugt ([3 SNR-Werten: 3 dB, 50 dB und 50+3 dB] $\times$ [9 Feature-Untermengen]. Der Datensatz mit einem SNR von 7 dB wird *nicht* zum Training verwendet). Mit diesen 27 Trainingsdatensätze werden mit Hilfe des REPTree-Algorithmus 27 Klassifikationsbäume erzeugt. Jeder Klassifikationsbaum wurde gegen die 3 Testdatensätze $A_{3\,\mathrm{dB}}$, $A_{50\,\mathrm{dB}}$ und $A_{7\,\mathrm{dB}*}$ getestet und die Accuracy berechnet. Das Signal $A_{7\,\mathrm{dB}*}$ erfüllt dabei eine Sonderrolle, da weder das Signal des Weinens noch das Rausch-Signal in den Trainingsdatensätzen enthalten sind und somit verifiziert wird, ob der Datensatz nur „auswendig gelernt“ wird oder das Modell auf neue Anwedungsfälle übertragen werden kann. Um Overfitting des Modells zu vermeiden und die Komplexität des Entscheidungsbaumes zu verringern, wurde die maximale Tiefe des REPTree auf 2 gesetzt. Die Ergebnisse sind in Tabelle .1 zu sehen.

Die Features, welche zu den höchsten Accuracy-Werten führten, sind die des *Cepstrum*-Bereiches, genauer gesagt das DiffF($\mathrm{Ceps_{mag}}$)-Feature, da es vom REPTree-Algorithmus als einziges Feature dieses Bereiches für die Entscheidungsbäumen ausgewählt wurde. Die Entscheidungsbäume, die mit dem DiffF($\mathrm{Ceps_{mag}}$)-Feature entworfen wurden, erreichten eine durchschnittliche Accuracy (das heißt, gemittelt über die Testsignale $A_{3\,\mathrm{dB}}$, $A_{50\,\mathrm{dB}}$ und $A_{7\,\mathrm{dB}*}$) von mindestens 91,45%. Der nächstbeste Entscheidungsbaum mit einer Accuracy von 86,96% wurde unter Verwendung der Features des Zeitbereiches und der rechnerisch aufwendigeren Autokorrelation auf dem Datensatz $S_{50+3\,\mathrm{dB},\mathrm{Zeit+Correlation}}$ entworfen. Sobald der Cepstrum-Bereich in Verbindung mit den Features der Bereiche *Zeit* und *Frequenz* verwendet wurde, wurde das DiffF($\mathrm{Ceps_{mag}}$)-Feature vom REPTree-Algorithmus bevorzugt, so dass die Features der anderen beiden Bereiche keine Anwendung mehr in den entsprechenden Bäumen fanden.

Auf Basis der Datensätze $S_{3\,\mathrm{dB},\mathrm{Ceps}}$, $S_{3\,\mathrm{dB},\mathrm{Zeit+Ceps}}$, $S_{3\,\mathrm{dB},\mathrm{Freq+Ceps}}$, $S_{50+3\,\mathrm{dB},\mathrm{Ceps}}$, $S_{50+3\,\mathrm{dB},\mathrm{Zeit+Ceps}}$ sowie
$S_{50+3\,\mathrm{dB},\mathrm{Freq+Ceps}}$ wurde der selbe Entscheidungsbaum erzeugt, der in Listing 4.2 zu sehen ist. Auf Basis der Datensätze $S_{50\,\mathrm{dB},\mathrm{Ceps}}$ und $S_{50\,\mathrm{dB},\mathrm{Zeit+Ceps}}$ wurde der Entscheidungsbaum in Listing 4.3 erzeugt. Es ist zu sehen, dass (a) beide Entscheidungsbaum einen einfachen Grenzwert für das DiffF($\mathrm{Ceps_{mag}}$)-Feature setzen, und zweitens (b) sich die beiden Modelle nur im konkreten Wert des Grenzwertes unterscheiden.

Da das Modell aus Listing 4.2 eine durchschnittliche Accuracy von 92,22% und das Modell aus Listing 4.3 eine unwesentlich geringere Accuracy von 91,45% erreicht, wurden für beide Modelle die Specificity und Sensitivity berechnet, um eine Entscheidung für eines der beiden Modelle fällen zu können. Dazu wurden die Signalmengen $A_{3\,dB}$, $A_{50\,dB}$ und $A_{7\,dB^*}$ in Frames à 100 Windows zerlegt und für jedes Zeitfenster die Senstivity, Specificity und Accuracy bezüglich der beiden Modelle berechnet. Die Ergebnisse werden als Boxplots in Abbildung .1 dargestellt. Der Unterschied zwischen den Modellen ist am Stärksten beim Testing gegen die Signale mit einem SNR von 3 dB und 7 dB zu sehen. Das Modell mit dem Grenzwert von 0.03 erzielt in beiden Fällen eine höhere Specificity, aber geringere Senstivitiy als das Modell mit dem Grenzwert bei 0.02. Es wurde sich für das Modell für mit einem Grenzwert von 0.02 entschieden, da durch die höhere Senstivity mehr Wein-Signale erkannt werden, die in späteren Verarbeitungsschritten immernoch als False-Positives erkannt und verworfen werden können. Einmal im Prozess der VAD als Stimmlos markierte Fenster werden jedoch nicht weiter verarbeitet und gehen somit „verloren“.

Listing 4.2: Entscheidungsbaum für die VAD mit einem Cepstrum-Grenzwert von 0.02

```
if FDiff(Ceps_mag(x_i)) < 0.02
|   C(x_i) = 0
|else
|   C(x_i) = 1
```

Listing 4.3: Entscheidungsbaum für die VAD mit einem Cepstrum-Grenzwert von 0.03

```
if FDiff(Ceps_mag(x_i)) < 0.03
|   C(x_i) = 0
|else
|   C(x_i) = 1
```

Der Finale Funktion zur Klassifikation eines Signalfensters C(x) in 0 = *Stille* oder 1 = *Stimme* ist somit durch Gleichung 4.20 gegeben.

$$C(x) = \begin{cases} 1, & \text{if } \text{Ceps}_{mag}(x) >= 0.02 \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

4.3.4 Markierung der Cry-Units

Das Ergebnis der Voice-Activity-Detection ist eine Zurdnung aller Signalfenster $x_1...x_n$ zu den Klassen $C(x_i) = 0$ *Stille* oder $C(x_i) = 1$ *Stimme*. In [20] wird die Idee vorgestellt, zusammenhängende und ununterbrochene Ketten als *stimmhaft* klassifizierter Signalfesnter zu *Stimm-Segmenten* zusammenzufassen, welche in diesem Zusammenhang eine *Cry-Units* entsprechen. Abbildung 4.14 veranschaulicht diese Gruppierung. Formel 4.21 gibt die Definition des Datentypes *Cry-Unit* [CU]. Eine Cry-Unit definiert sich durch einen Anfangszeitpunkt $start$, einen Endzeitpunkt end und der Liste seiner Signalfenster $windows = [x_1...x_n]$.

Algorithmus 2 zeigt in Pseudo-Code, wie in der Liste aller Signalfenster $X_{windows} = [x_1...x_n]$ eine Liste von Cry-Units $CU_{all} = [cu_1...cu_m]$ markiert wird. Die Funktion C(x) ist

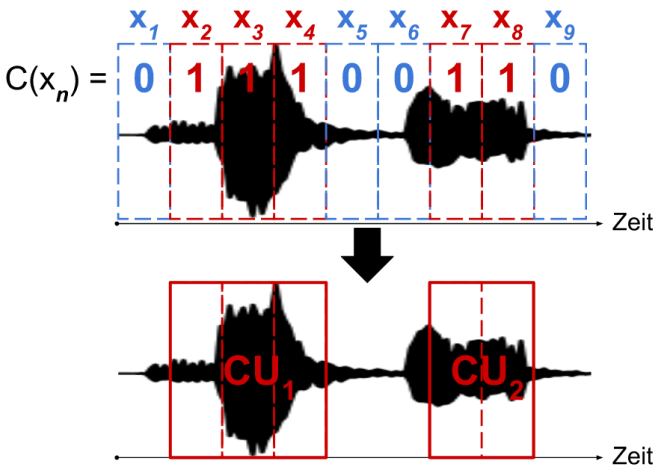

Abbildung 4.14: Zusammenfassung klassifizierter Signalfenster zu Cry-Units

die Klassifikations-Funktion der Signalfenster in Stille/Stimme nach Gleichung 4.20. Die Funktion getTimeOf(x) liefert die Anfangszeitpunkt des Signalfensters x.

$$CU = (windows = [x_1 ... x_n], start \in Zeit, end \in Zeit) \tag{4.21}$$

$$\lambda(CU) = CU.end - CU.start \tag{4.22}$$

$$\mathrm{d}(CU_i, CU_j) = CU_j.start - CU_i.end \tag{4.23}$$

Die Dauer eine Cry-Unit $\lambda(CU)$ wird nach Formel 4.22 berechnet. Der (Stille)-Zeitraum zwischen zwei Cry-Units $\mathrm{d}(CU_i, CU_j)$, wird nach Formel 4.23 berechnet. Diese Zusammenhänge werden in Abbildung 4.15 visualisiert.[20]

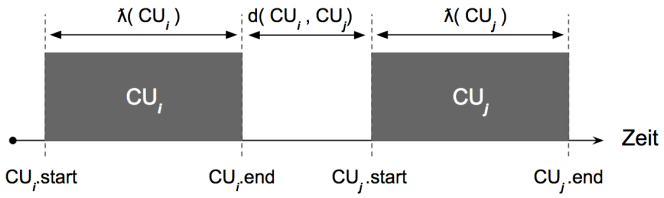

Abbildung 4.15: Beziehung zwischen agrenzenden Segmenten

Algorithm 2 Gruppierung von x-Windows zu Cry-Units

```
1: function turnWindowsIntoCryUnits(X_windows)
2:     CU_all ← []
3:     cu_i ← ([], 0, 0)
4:     for i = 1...length(X_windows) do
5:         c_i ← C(x_i)
6:                                                    ▷ Start of Cry-Unit
7:         if c_i == 1 ∧ isEmpty(cu_i.windows) then
8:             cu_i.start ← getTimeOf(x_i)
9:             cu_i.windows ← [cu_i.windows, x_i]
10:        end if
11:                                                   ▷ Inside Cry-Unit
12:        if c_i == 1 ∧ ! isEmpty(cu_i.windows) then
13:            cu_i.windows ← [cu_i.windows, x_i]
14:        end if
15:                                                   ▷ End of Cry-Unit
16:        if c_i == 0 ∧ ! isEmpty(cu_i.windows) then
17:            cu_i.end ← getTimeOf(x_i)
18:            CU ← [CU, cu_i]
19:            cu_i.windows ← []
20:        end if
21:    end for
22:                                  ▷ End last Cry-Unit by force if still open.
23:    if ! isEmpty(cu_i.windows) == 0 then
24:        cu_i.end ← getTimeOf(X_windows[end])
25:        CU_all ← [CU_all, cu_i]
26:    end if
       return CU_all
27: end function
```

4.3.5 Decision Smoothing

Abbildung 4.16 zeigt ein Audiosignal mit einem Signal-Rausch-Abstand von 3 dB, bei dem die Klassifizierung nach dem Entscheidungsbaum aus Listing 4.2 durchgeführt wurde. Die rote Linie zeigt die tatsächliche Klassifizierung, und die grüne Linie die gefundene Klassifizierung nach dem vorgestellten Algorithmus. Die tatsächlichen/gefunden Cry-Units sind klar zu erkennen als die Bereiche, die von der roten/grünen Linie überspannt werden. Es ist zu sehen, dass False-Negatives und False-Positives in der Klassifizierung enthalten sind. Im folgenden werden drei charakteristische Arten falscher Klassifizierungen näher erläutert:

False Negatives nach (a) : Eine korrekt erkannte, längere Cry-Unit wird zu früh beendet. Oft werden kurz nach dem Ende sehr kurze Cry-Units erkannt, die eigentlich noch zu der längeren, vorhergehenden Cry-Unit gehören.

False Positives nach (b): Kurze Cry-Units werden in eigentlichen Stille-Bereichen erkannt.

False Negatives nach (c): Eine Cry-Unit zerfällt in zwei kürzere Cry-Units, da einige Signalfenster in der Mitte als Stille erkannt wurden.

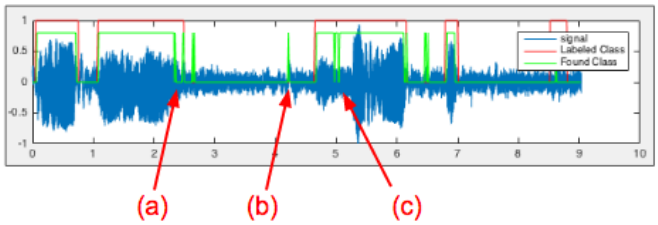

Abbildung 4.16: Klassifizierung vor dem Decision Smoothing

Im Process des **Decision Smoothing** werden kontextuelle Informationen genutzt, um nachträglich False-Positives und False-Negatives zu entfernen. Es werden dazu die in [20] präsentierten Ideen verwendet. Es werden zwei Parameter eingeführt: λ_{min}, die Mindestlänge einer akzeptierten Cry-Unit, und d_{min}, die Mindestlänge eines akzeptierten Stille-Segmentes. Das Decision Smoothing wird nach den folgenden Entscheidungsregeln durchgeführt:

- ist $\lambda(CU_i) \leq \lambda_{min}$?
  - wenn $\lambda(CU_{i-1}) > \lambda_{min}$ und $d(CU_{i-1}, CU_i) \leq d_{min}$, dann vereinige CU_i mit CU_{i-1} . $\Rightarrow$ behebt False-Negatives des Types (a)
  - ansonsten entferne $CU_i \Rightarrow$ behebt False-Negatives des Types (b)
- wenn $\lambda(CU_i > \lambda_{min}$ und $d(CU_{i-1}, CU_i) \leq d_{min}$, dann vereinige CU_i mit CU_{i-1} . $\Rightarrow$ behebt False-Negatives des Types (c)

Die Entscheidungsregeln greifen Algorithmus greifen nur auf die aktuellen und die letzten bekannte Cry-Unit um, um eine kontinuierliche Analyse zu gewährleisten, weshalb die Entscheidungsregeln jedoch auch komplex sind. Bei einer offline-Analyse können die Entscheidungsregeln vereinfacht werden, da False-Negative Type (a) und (c) mit der selben Regeln abgefragt werden können. Algorithmus 3 zeigt in Pseudo-Code, wie das Decision-Smoothing durchgeführt wird. Input der Funktion ist die Liste aller Cry-Units CU_{all}, die durch Algorithmus 2 entstanden ist, sowie die Grenzwerte λ_{min}, d_{min}. Ausgang der Funktion ist die Liste aller Cry-Units nach dem Decision-Smoothing $CU_{smoothed}$.

Algorithm 3 Decision-Smoothing of VAD

```
1:  function DECISIONSMOOTHING(CU_all, λ_min, d_min)
2:      CU_smoothed ← [CU_all[1]]
3:      for i = 2...length(CU_all) do
4:          cu_i ← CU_all[i]
5:          cu_{i-1} ← CU_smoothed[end]
6:          if λ(cu_i) > λ_min then
7:              if d(cu_{i-1}, cu_i) > d_min then
8:                  CU_smoothed ← [CU_smoothed, cu_i]
9:              else
10:                                          ▷ Erase False-Negative Type (c)
11:                 cu_i ← vereinige(cu_i, cu_{i-1})
12:                 CU_smoothed ← [CU_smoothed[1 : end − 1], cu_i]
13:             end if
14:         else
15:                                          ▷ Erase False-Negative Type (a)
16:             if λ(cu_i) > λ_min ∧ d(cu_{i-1}, cu_i) ≤ d_min then
17:                 cu_i ← vereinige(cu_i, cu_{i-1})
18:                 CU_smoothed ← [CU_smoothed[1 : end − 1], cu_i]
19:             else
20:                                          ▷ Don't accept cu_i. Erases False-Positives (b)
21:             end if
22:         end if
23:     end for
        return CU_smoothed
24: end function
```

Abbildung 4.17 zeigt das Signal vor und nach dem Decision-Smoothing. Die Parameter wurden experimentell mit $\lambda_{min} = 50ms$ und $d_{min} = 50ms$ bestimmt.

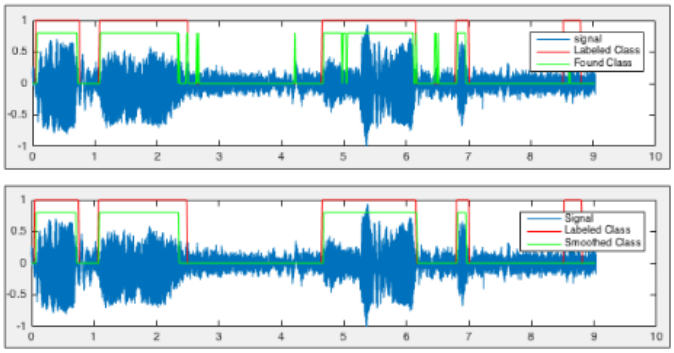

Abbildung 4.17: Klassifizierung vor und nach dem Decision Smoothing

4.4 Segmentierung

Das Ergebnis der Voice-Activiy-Detection ist eine Liste an Cry-Units $CU_1...CU_n$. Das Ziel ist nun, diese Cry-Units zu Cry-Segmenten zu gruppieren. Ein Cry-Segment definiert sich

nach Golub et al [14] als „die komplette klangliche Antwort auf einen spezifischen Stimulus. Sie kann mehrere Cry-Units entahlten “. Die Defintion lässt folgende Fragen offen:

- Beginnt das Segment bereits bei Zuführung des Stimulus, oder erst ab der ersten Cry-Unit?
- Wodurch definiert sich der Beginn, wenn ohne Zuführung eines Stimulus das Baby beginnt, zu weinen?
- Endet ein Cry-Segment mit Ende der letzten „Cry-Unit“, oder erstreckt es sich bis zu Beginn des nächsten Cry-Segmentes?

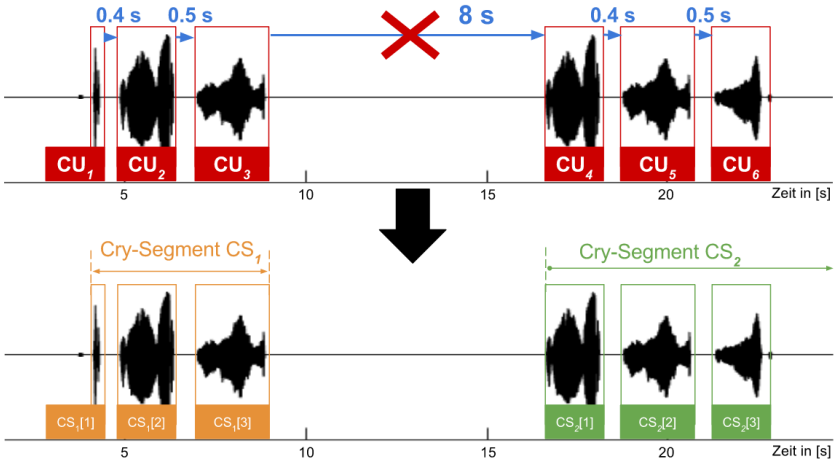

Abbildung 4.18: Ergebnis der Segmentierung

Die Zusammenfassung von Cry-Units zu Cry-Segmentes unterliegt einer gewissen subjektiven Einschätzung, welche Cry-Units als Zuammengehörig angesehen werden, insbesondere, wenn kein erkennbarer Stimulus vorliegt. Abbildung 4.19 verdeutlicht das Problem.

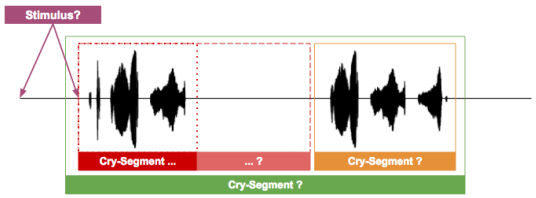

Abbildung 4.19: Mögliche Segmentierungen eines Signals

Um das Problem zu objektivieren, wird es mathematisch formuliert. Eine *Cry-Segment* $[CS]$ wird als Datentyp nach Formel 4.24 definiert. Ein Cry-Segment ist folglich eine Liste aufeinander folgenden Cry-Units, die gruppiert werden. Der Start-Zeitpunkt eines Cry-Segmentes wird nach Formel 4.25 als der Startzeitpunkt der ersten Cry-Unit des Segmentes definiert. Die Begründung für diese Entscheidung liegt darin, dass rein aus dem Audiomaterial der Zeitpunkt des Stimulus nicht festgestellt werden kann und das Segment mit Sicherheit erst bei der ersten feststellbaren Cry-Unit beginnt. Das Ende eines Segmentes wird definiert als das Ende der letzten Cry-Unit nach Gleichung 4.26. Die Begründung liegt darin, dass das Ende der Reaktion auf den Stimulus ebenfalls rein aus dem Audiosignal

abgeleitet werden kann und somit der einzig feststellbare Indikator die letzte Cry-Unit des Segmentes ist.

$$CS = [cu_1, ..., CU_n] \tag{4.24}$$

$$start(CS) = CS[1].start \tag{4.25}$$

$$end(CS) = CS[end].end \tag{4.26}$$

Wurde bei der kontinuierlichen Analyse des Signals ein Segment geschlossen, führt die Markierung einer neuen Cry-Unit zur Eröffnung eines neuen Segmentes, dessen Start-Zeitpunkt der Start-Punkt dieser Cry-Unit ist. Die Frage ist, welches Kriterium zum schließen dieses Segments führt. Laut Golub et al [14] ein Cry-Segment „die komplette klangliche Antwort auf einen spezifischen Stimulus “. Eine mögliche und objektiv messbare Interepration dieses Endes ist, dass nach dem Auftreten von Cry-Units eine längere Stille mit einer Abwesenheit von Cry-Units festgestellt wird, da das Baby „aufgehört hat, zu weinen “. Übertragen auf die in 4.3.4 vorgestellte Terminologie heißt das, dass ein Segment beendet und ein neues begonnen wird, wenn die Distanz (Zeitraum der Stille) zwischen zwei benachbarten Cry-Units $\mathrm{d}(CU_i, CU_{i+1})$ einen gewissen Grenzwert $t_{seg-max}$ überschreitet. Gleichung 4.27 formalisiert diesen Zusammenhang. Daraus lässt sich schlussfolgern, dass die Distanzen zwischen allen benachbarten Cry-Unit eines Segmentes unter diesem Grenzwert $t_{seg-max}$ liegen. Gleichung 4.28 formalisiert diese Nebenbedingung an die Cry-Units eines Segmentes.

$$d(cu_i, cu_{i+1}) > t_{seg-max} \rightarrow CS_n = [CS_n, cu_i] \wedge CS_{n+1} = [cu_{i+1}] \tag{4.27}$$

$$\forall i = 1...\mathrm{length}(CS) - 1 : \mathrm{d}(CS[i], CS[i+1]) \leq t_{seg-max} \tag{4.28}$$

Die einfachste Art, $t_{seg-max}$ festzulegen, ist, einen festen Grenzwert von t s zuzuweisen. Abbildung 4.18 visualisiert die so resultierende Segmentierung an einem Beispiel. Jeder Grenzwert mit $t_{seg-max} > 0.5\,\mathrm{s}$ würde zu der gezeigten Segmentierung führen.

Algorithmus 4 zeigt die Segmentierung nach diesem Prinzip in Pseudo-Code. Input des Algorithmus ist die Liste aller Cry-Units $CU_{all} = [cu_1...cu_n]$, die nach dem Decision-Smoothing nach Algorithmus 3 entstanden ist. Das Ergebnis des Algorithmus ist die Liste, die alle gefundene Cry-Segmente $[cs_1...cs_n]$ enthält.

Algorithm 4 Gruppierung von Cry-Units zu Cry-Segments

```
1: function segmentCryUnits(CU_all, t_seg−max)
2:     CS_all ← []
3:     cs_i ← [CU_all[1]]
4:     for i = 2...length(CU_all) do
5:         cu_i ← CU_all[i]
6:         cu_{i−1} ← CU_all[i − 1]
7:         if d(cu_{i−1}, cu_i) < t_seg−max then
8:             cs_i ← [cs_i, cu_i]
9:         else
10:            CS_all ← [CS_all, cs_i]
11:            cs_i ← [cu_i]
12:        end if
13:    end forreturn CS_all
14: end function
```

Algorithmus 4 kann zwar kontinuierlich durchgeführt werden, da er jeweils nur auf die aktuelle gefundene und eine vergangene Cry-Unit zurückgreift, hat in dieser Form jedoch den nachteil, dass das Ende eines Segmentes später als notwendig festgestellt wird. Angenommen, ein Grenzwert von $t_{seg-max} = 20\,\mathrm{s}$ wurde festgelegt

Bei einer kontinuierlichen durchgeführten Segmentierung wird das erste Segment dann eröffnet, sobald die erste Cry-Unit durch die VAD markiert wurde, und diese Cry-Unit dem Segment hinzugefügt. Die Dauer der Stille nach dieser Cry-Unit wird kontinuierlich gemessen. Wird ein nächste Cry-Unit festgestellt, bevor die Stille $d_{seg-max}$ übersteigt, so wird diese Cry-Unit dem Segment hinzugefügt und das Messen der Stille nach dieser Cry-Unit beginnt von vorne. Dieser Prozess wird so lange wiederholt, bis die Dauer der Stille nach einer hinzugefügten Cry-Unit $d_{seg-max}$ übersteigt. Dann wird das Segment beendet und der Endzeitpunt des Segmentes auf den Endzeitpunkt der letzten Cry-Unit gesetzt. Abbildung 4.18 zeigt die resultiderende Segmentierung für Beispielsignal mit $d_{seg-max} = 3\,\mathrm{s}$. Tatsächlich würde in dem Beispiel jeder Grenzwert $d_{seg-max} > 0.5\,\mathrm{s}$ zur gezeigten Segmentierung führen.

Es gibt verschiedene Möglichkeiten, die höchst mögliche Pause $\mathrm{d}_{seg-max}$ zu definieren. Der Einfachste Fall, der auch in Abbildung 4.18 angenommen wurde, ist das Setzen eines global festlegten Grenzwertes. Weitere Möglichkeiten sind, $\mathrm{d}_{seg-max}(CS)$ als Funktion des Segmentes selber zu gestalten. So könnte beispielsweise ein längeres Segment eine höhere maximal-Pause erzeugen.

Schlussendlich konnten in der Fachliteratur keine konkreten Hinweise zur Bestimmung von $\mathrm{d}_{seg-max}$ gefunden werden. Daher wurde entschieden, dem Arzt, der das System benutzt, selber einstellen zu lassen. Es wird mit einem Festen

4.5 Feature-Extraction

4.6 Ableitung der Schmerz-Scores

4.7 Visualisierung

5 Zusammenfassung

Literaturverzeichnis

[1] K J S Anand. *Pain in Neonates and Infants*. Elsevier, 2007.

[2] Zachariah Boukydis Barry Lester. *Infant Crying: Theoretical and Research Perspectives*. Springer, 1985.

[3] Judy Bildner. *CRIES Instrument Assessment Tool of Pain in Neonates*. City of Hope Pain, 1997. Online unter `http://prc.coh.org/pdf/CRIES.pdf`.

[4] R Sisto & Giuseppe Buonocore Carlo Bellieni, Franco Bagnoli. Cry features reflect pain intensity in term newborns: An alarm threshold. *Pediatric Research*, 5:142–146, 1. Online unter `https://www.researchgate.net/publication/297827342_Cry_features_reflect_pain_intensity_in_term_newborns_An_alarm_threshold`.

[5] Douglas Dankel. The ID3 Algorithm , 1997. Online unter `http://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-pa/2.htm`.

[6] Alin Dobra. Introduction to classification and regression, 2005. Online erhältlich unter: `https://www.cise.ufl.edu/~adobra/datamining/classif-intro.pdf`.

[7] H. Hollien & T Murry E Müller. Perceptual responses to infant crying: identification of cry types. *Journal of Child Language*, 1(1):89–95, 1974. Online unter `https://www.cambridge.org/core/journals/journal-of-child-language/article/perceptual-responses-to-infant-crying-identification-of-cry-types/4F0F8088116FCE381851D8D560697A5F`.

[8] B. Simak E. Verteletskaya. Performance Evaluation of Pitch Detection Algorithms, 2009. Online unter `http://access.feld.cvut.cz/view.php?cisloclanku=2009060001`.

[9] Jan Hamers & Peter Gessler Eva Cignac, Romano Mueller. Pain assessment in the neonate using the Bernese Pain Scale for Neonates. *Early Human Development*, 78(2):125–131, 2004. Online unter `http://www.sciencedirect.com/science/article/pii/S0378378204000337`.

[10] Trevor Hastie Gareth James, Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 2013.

[11] Ricardo Gutierrez-Osuna. Introduction to Speech Processing. Online unter `http://courses.cs.tamu.edu/rgutier/csce689_s11/`.

[12] Health Facts For You. *Using Pediatric Pain Scales Neonatal Infant Pain Scale (NIPS)*, 2014. Online unter `https://www.uwhealth.org/healthfacts/parenting/7711.pdf`.

[13] Hodgkinson. Neonatal Pain Assessment Tool , 2012. Online unter `http://www.rch.org.au/uploadedFiles/Main/Content/rchcpg/hospital_clinical_guideline_index/PAT%20score%20update.pdf`.

[14] Michael J Corwin Howard L Golub. A Physioacoustic Model of the Infant Cry. In *Infant Crying - Theoretical and Research Perspectives*, chapter 3, pages 59 – 82. Plenung,

1985.

[15] Giorgio Ingargiola. Building Classification Models: ID3 and C4.5. Online unter http://cis-linux1.temple.edu/~giorgio/cis587/readings/id3-c45.html.

[16] Donna Geiss Laura Wozniak & Charles Hall Ivan Hand, Lawrence Noble. COVERS Neonatal Pain Scale: Development and Validation. *International Journal of Pediatrics*, 2010, 2010. Online unter https://www.hindawi.com/journals/ijpedi/2010/496719/.

[17] J Gorriz & J Segura J Ramorez. Voice Activity Detection. Fundamentals and Speech Recognition System Robustness. *Robust Speech Recongnition and Understanding*, page 460, 2007. Online unter http://cdn.intechopen.com/pdfs/104/InTech-Voice_activity_detection_fundamentals_and_speech_recognition_system_robustness.pdf.

[18] Jeih-weih Hung & Lin-shan Lee Jia-lin Shen. Robust Entropy-based Endpoint Detection for Speech Recognition in Noisy Environments. 1998. Online unter https://www.researchgate.net/publication/221489354_Robust_entropy-based_endpoint_detection_for_speech_recognition_in_noisy_environments.

[19] Carol Espy-Wilson & Tarun Pruthi Jonathan Kola. Voice Activity Detection. *MERIT BIEN*, 2011. Online unter http://www.ece.umd.edu/merit/archives/merit2011/merit_fair11_reports/report_Kola.pdf.

[20] Kim Weaver & Fathi M. Salam Khurram Waheed. A robust Algorithm for detecting speech segments using an entropic contrast. *IEEE*, 2003. Online unter http://ieeexplore.ieee.org/document/1187039/.

[21] Miroslav Kubat. *An Introduction to Machine Learning*. Springer, 2015.

[22] Wei-Yin Loh. A Comparative Performance Study of Several Pitch Detection Algorithms. *WIREs Data Mining Knowl Discovery*, 1:14–23, 2011.

[23] Tze-Wey Loong. Understanding sensitivity and specificity with the right side of the brain. *BMJ*, 327(7417), 2003. Online unter https://www.ncbi.nlm.nih.gov/pmc/articles/PMC200804/.

[24] M M Homayounpour M H Moattar. A simple but efficient real-time Voice Activity Detection Algorithm. Signal Processing Conference, IEEE, August 2009. Online unter http://ieeexplore.ieee.org/document/7077834/?arnumber=7077834&tag=1.

[25] Stephen Marsland. *Machine Learning - An Algorithmic Perspective*. Chapman & Hall / CRC, 2009.

[26] Tom M Mitchell. *Machine Learning*. WCB McGraw-Hill, 1997.

[27] Hans M Koot Dick Tibboel Jan Passchier & Hugo Duivenvoorden Monique van Dijk, Josien de Boer. The reliability and validity of the COMFORT scale as a postoperative pain instrument in 0 to 3-year-old infants. *Pain*, 84(2):367—-377, 2000. Online unter http://www.sciencedirect.com/science/article/pii/S0304395999002390.

[28] Taddio Nulman. A revised measure of acute pain in infants. *J Pain Symptom Manage*, 10:456–463, 1995. Online unter http://geriatricphysio.yolasite.com/resources/Modified%20Behavioral%20Pain%20Scale%20(MBPS)%20in%20infants.pdf.

[29] J L Mathew P J Mathew. Assessment and management of pain in infants. *Postgrad Med J*, 79:438–443, 2003. Online unter http://pmj.bmj.com/content/79/934/438.full.

[30] Dr. Iain Pardoe. Stat 501 - regression methods. Online erhältlich unter: https://onlinecourses.science.psu.edu/stat501/node/251, 2017.

[31] Steven Creech & Marc Weiss. Patricia Hummel, Mary Puchalski. N-PASS: Neonatal Pain, Agitation and Sedation Scale – Reliability and Validity. *Pediatrics/Neonatology*, 2(6), 2004. Online unter http://www.anestesiarianimazione.com/2004/06c.asp.

[32] Susan Parker-Price & Ronald Barr Philip Zeskind. Rythmic organization of the Sound of Infant Cry. *Dev Psychobiol*, 26(6):321–333, 1993. Online unter https://www.ncbi.nlm.nih.gov/pubmed/8119482.

[33] R Ward & C Laszlo Qiaobing Xie. Automatic Assessment of Infants' Levels-of-Distress from the Cry Signals. *IEEE Transanctions on Speech and Audio Processing*, 4(4):253–265, 1996. Online unter http://ieeexplore.ieee.org/document/506929/.

[34] Yizhar Lavner Rami Cohen. Infant Cry Analysis and Detection. IEEE 27-th Convention of Electrical and Electronics Engineers in Israel, 2012. Online unter http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6376996.

[35] Brian Hopkins & James Green Ronald Barr. *Crying as a Sign, a Symptom, and a Signal*. Mac Keith Press, 2000.

[36] J R Shayevitz & Shobha Malviya Sandra Merkel, Terri Voepel-Lewis. The FLACC: A Behavioral Scale for Scoring Postoperative Pain in Young Children. *Pediatric Nursing*, 23(3):293–7, 1996. Online unter https://www.researchgate.net/publication/13998379_The_FLACC_A_Behavioral_Scale_for_Scoring_Postoperative_Pain_in_Young_Children.

[37] Andreas Spanias Sassan Ahmadi. Cepstrum-Based Pitch Detection Using a New Statistical V/UV Classification Algorithm. *IEEE Transactions on Speech and Audio Detection*, 7(3):333–338, 1999. Online unter http://ieeexplore.ieee.org/document/759042/.

[38] Julius Smith. *Spectral Audio Signal Processing*. Center for Computer Research in Music and Acoustics (CCRMA), 1993. Online unter https://www.dsprelated.com/freebooks/sasp/Short_Time_Fourier_Transform.html.

[39] Sabine Deligne & Peder Olsen Trausti Kristjansson. Voicing Features for Robust Speech Detection. Interspeech Lisboa, September 2005. Online unter http://papers.traustikristjansson.info/wp-content/uploads/2011/07/KristjanssonRobustVoicingEurospeech2005.pdf.

[40] P H Wolff. The role of biological rhythms in early psychological development. *Bulletin of the Menninger Clinic*, 31:197–218, 1967.

Appendices

Tabelle .1: Accuracy-Werte der Grenzwertfindung mit REPTree

$S_{Training}$	3 dB				50 dB				50+3 dB			
A_{Test}	3 dB	50 dB	7 dB*	Mean	3 dB	50 dB	7 dB*	Mean	3 dB	50 dB	7 dB*	Mean
Zeit	77.81%	79.02%	86.04%	80,96%	49.33%	94.70%	48.66%	64,23%	77.54%	92.47%	84.38%	84,80%
Freq	82.05%	89.28%	82.71%	84,68%	70.52%	94.37%	55.06%	73,31%	81.75%	91.22%	74.90%	82,62%
Ceps	88.98%	94.72%	92.96%	**92,22%**	86.83%	94.68%	92.83%	**91,45%**	88.98%	94.72%	92.96%	**92,22%**
Corr	80.45%	73.47%	84.89%	79,60%	73.07%	87.14%	77.98%	79,39%	77.90%	84.88%	82.84%	81,87%
Zeit+Freq	82.05%	89.28%	82.71%	84,68%	70.52%	94.37%	55.06%	73,31%	81.75%	91.22%	74.90%	82,62%
Zeit+Ceps	88.98%	94.72%	92.96%	**92,22%**	86.83%	94.68%	92.83%	**91,45%**	88.98%	94.72%	92.96%	**92,22%**
Zeit+Corr	80.45%	73.47%	84.89%	79,60%	49.33%	94.70%	48.66%	64,23%	80.32%	92.35%	88.22%	86,96%
Freq+Ceps	88.98%	94.72%	92.96%	**92,22%**	70.65%	94.75%	55.06%	73,49%	88.98%	94.72%	92.96%	**92,22%**
Freq+Corr	82.05%	89.28%	82.71%	84,68%	70.52%	95.60%	95.60%	87,24%	81.75%	94.42%	74.90%	83,69%

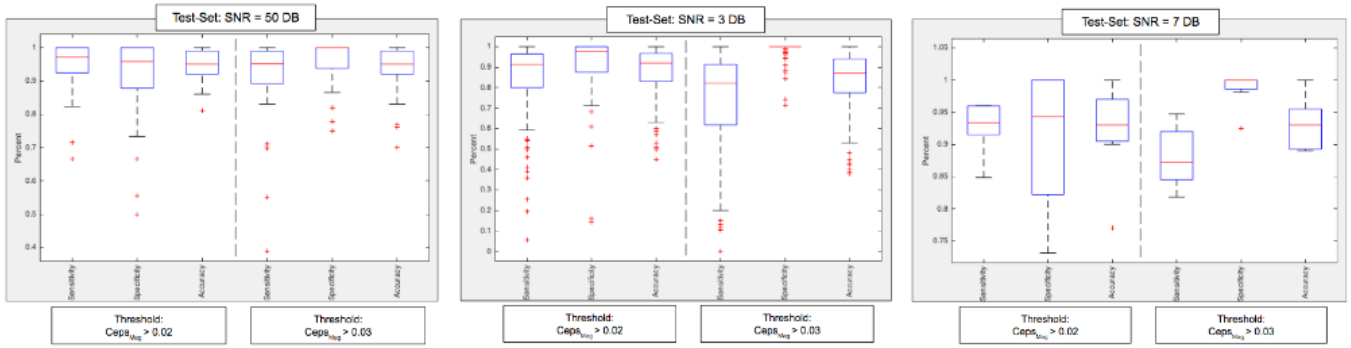

Abbildung .1: Boxplot-Auswertung über Sensitivity, Specificity und Accuracy der beiden VAD-Modelle