

PH.D. THESIS

**A statistical pattern recognition approach to
symbolic music classification**

Pedro J. PONCE DE LEÓN AMADOR

Supervisor:

José M. IÑESTA QUEREDA



Universitat d'Alacant
Universidad de Alicante

Departament de Llenguatges i Sistemes Informàtics
Departamento de Lenguajes y Sistemas Informáticos

September 2011

*Para Alvaro y Merche,
por su paciencia y cariño.*

Agradecimientos

Para llegar hasta esta página he recorrido un larguísimo camino, que empezó probablemente cuando mis padres me regalaron mi primer instrumento musical, y mi hermano mi primera calculadora. Ellos no se imaginaban como iba a terminar la cosa, claro. Es lo que tiene hacer regalos sin pensar en las consecuencias. Por ello, y por todo lo demás, les quiero expresar mi eterno agradecimiento y todo mi amor.

La persona que me ha llevado de la mano en el (también largo) tramo final de este camino es el director de esta tesis, José Manuel Iñesta. Sus contribuciones al desarrollo de este trabajo van mucho más allá de lo acostumbrado. Quiero agradecerle sus numerosas aportaciones y consejos tanto en lo académico y lo profesional como en lo personal, así como las facilidades que siempre me ha brindado para desarrollar mi trabajo de la mejor forma posible.

Los consejos y el apoyo de los miembros del *Grupo de Reconocimiento de Formas e Inteligencia Artificial* del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante han sido también de vital importancia en muchos aspectos de esta investigación. Vaya para todos ellos mi agradecimiento, y en especial a los miembros del Laboratorio de Informática Musical, liderado por Jose Manuel Iñesta: David Rizo, cuya capacidad de trabajo no deja de asombrarme día tras día, Antonio Pertusa, Carlos Pérez-Sancho, Tomás Pérez, José F. Bernabeu, Plácido R. Illescas, María Hontanilla, Tico Agoiz y Javier Sober, quienes de una forma u otra han colaborado en este trabajo.

Quiero también agradecer la cálida acogida que me han brindado otros investigadores para desarrollar mi trabajo en el seno de su grupo: Gérard Assayag, Carlos Agon, Jean Bresson, Moreno Andreatta y Karim Haddad, del *Musical Representations Team*, del IRCAM, y Andreas Rauber, Thomas Lidy y Rudi Mayer, del *Information & Software Engineering Group* de la Vienna University of Technology.

El conjunto de trabajos que soportan esta investigación ha sido financiado a lo largo de los años por los siguientes proyectos:

- DRIMS: Descripción y recuperación de información musical y sonora. Ministerio de Ciencia e Innovación. TIN2009-14247-C02-02
- MIPRCV: Multimodal Interaction in Pattern Recognition and Computer Vision. Ministerio de Educación y Ciencia (Consolider Ingenio 2010). CSD2007-00018

-
- UA-CPS: Desarrollo de un sistema automático para el análisis armónico interactivo de secuencias musicales. Universidad de Alicante. GRE09-32
 - Acc. Int. E-A: Clasificación de géneros musicales combinando descriptores de audio y simbólicos mediante un sistema de transcripción automática. Ministerio de Educación y Ciencia. HU2007-0025
 - PROSEMUS: Procesamiento semántico de música digital. Ministerio de Educación y Ciencia. TIN2006-14932-C02-02.
 - GV-JRRJ: Extracción de características y combinación de clasificadores para el reconocimiento de firmas, estilos musicales, espectros estelares y desambiguación sintáctica. Generalitat Valenciana I+D+i (Investigadores emergentes). GV06/166
 - GV-JMI: Transferencia de tecnologías de categorización de textos a aplicaciones de recuperación de información en música digital. Generalitat Valenciana. GV04B-541
 - TIRIG: Tratamiento, indexación y recuperación de información en grandes volúmenes de datos mediante técnicas de reconocimiento de formas: aplicaciones en descripción y clasificación de datos secuenciales. CICYT. 2003-08496-C04-04

Por último, mi mayor agradecimiento a mi compañera, Merche, y a mi hijo, Alvaro, que hace unos días le preguntó a su madre, “Mamá, pero ¿cuántas tesis ha escrito ya papá?” Gracias por vuestra enorme paciencia y optimismo, vuestro apoyo y vuestro amor. Os quiero.

Pedro J. Ponce de León
Julio, 2011

Contents

1	Introduction	1
1.1	Music information retrieval	1
1.2	Melody and genre	6
1.3	Don't tell me how it sounds...	6
1.4	Melody part selection	9
1.5	Music genre recognition	16
1.6	Brief outline of this thesis	25
2	Technical background	27
2.1	Unsupervised learning	27
2.2	Supervised learning	37
2.3	Information Fusion	49
2.4	Ensemble Methods	51
2.5	Feature selection	56
2.6	Fuzzy systems	60
2.7	Experiment design	65
2.8	Performance evaluation	66
3	Melody part selection	79
3.1	Melody characterization in multi-part files	79
3.2	Human-friendly melody part selection	104
3.3	A fuzzy approach	117
3.4	Conclusions on melody part selection	140
3.5	Contributions in this chapter	143
4	Music genre recognition	145
4.1	Feature space exploration by SOM	146
4.2	Human music genre recognition	159
4.3	Supervised music genre recognition	166
4.4	Ensembles for music genre recognition	189
4.5	Fusion from audio and symbolic domains	199
4.6	Conclusions on music genre recognition	222
4.7	Contributions in this chapter	225
5	Summary and future perspectives	227
5.1	Melody part selection	227
5.2	Music genre recognition	228
5.3	Perspectives on melody part selection	229
5.4	Perspectives on music genre recognition	230

CONTENTS

5.5 Publications	234
Appendices	239
A Software tools used/developed	241
A.1 Third-party tools	241
A.2 GUI for melody part selection	242
A.3 GUI for music genre recognition	243
B Predefined fuzzy sets	245
C Resumen en castellano	249
C.1 Introducción	249
C.2 Selección de partes melódicas	254
C.3 Reconocimiento del género musical	260
C.4 Conclusiones y desarrollo futuro	264
Bibliography	267

List of Figures

2.1	A competitive learning net	28
2.2	Neighborhood and learning rate evolution during SOM training.	32
2.3	Two usual unit configuration in a bidimensional SOM.	34
2.4	Example of U-matrix SOM visualization	35
2.5	Example of Sammon projection of the map shown in Fig. 2.4	36
2.6	Optimal hyperplane in a two-class problem projection	41
2.7	A fully connected three-layer perceptron.	43
2.8	The logistic function	44
2.9	An early fusion scheme for music related features.	50
2.10	A late fusion scheme for music related features.	51
2.11	Different models for giving the authority (a_k) to each classifier	53
2.12	A non-dominated pair example in Pareto-optimal selection.	56
2.13	Representation of fuzzy speeds by fuzzy sets.	61
2.14	Example of fuzzy variables	62
2.15	The fuzzyfication step	64
2.16	Fuzzy inference step example	64
2.17	Defuzzyfication example	65
2.18	9GDB dataset taxonomy.	74
2.19	MIREX2007 audio genre dataset taxonomy.	76
3.1	Occupation and polyphony rates examples.	83
3.2	Distribution of values for some descriptors	85
3.3	Melody vs. non-melody classification success and baseline	90
3.4	Comparison of melody track selection results, based on genre	101
3.5	A snapshot of the melody part detection graphical interface.	103
3.6	Average rule set coverage, as a function of N	112
3.7	TP/FP ratio for average rule coverage.	113
3.8	Fuzzy set example for attribute <i>TrackNormalizedDuration</i>	120
3.9	Fuzzy set shapes utilized and their parameters	123
3.10	Output attribute <i>IsMelody</i> modeled by singleton fuzzy sets	123
3.11	Overlapping fuzzy set partitions.	125
3.12	Representation scheme of fuzzy sets.	125
3.13	Boundaries of a fuzzy set.	126
3.14	fuzzifying a crisp boundary.	128
3.15	Fuzzy set definitions from the best evolved FIS (FISO 1).	132
3.15	Fuzzy set definitions from the best evolved FIS (cont.).	133
3.16	(top) GA fitness by hits. (bottom) GA fitness by F -measure	134
3.16	(cont.) GA fitness by F -measure	135
3.17	Alternative setup for the fuzzy output variable <i>IsMelody</i>	138

LIST OF FIGURES

3.18	Predefined fuzzy sets for attribute <i>AvgAbsInterval</i>	139
4.1	Depiction of the exploration methodology	148
4.2	Contribution to classification by SOM planes	149
4.3	Sammon projection of a 16×8 SOM map trained with unrestricted pitch range random melodies vs. jazz melodies. .	151
4.4	Sammon projection of the 16×8 map of figure 4.5: restricted pitch range random melodies versus real melodies.	152
4.5	SOM map for the same weights as in figure 4.4.	153
4.6	16×8 SOM calibrated with random melodies only.	153
4.7	16×8 SOM calibrated with real melodies only.	153
4.8	Weight planes of most contributive descriptors	154
4.9	Descriptors with lesser influence	154
4.10	SOM after training using two music genres	155
4.11	SOM trained with two genres (mixed labels)	156
4.12	Trajectory of the winning units for the jazz standard <i>Yesterdays</i> . .	157
4.13	Trajectory of the winning units for the <i>Divertimento in D</i> . .	158
4.14	Sammon projection of the SOM map in figure 4.11.	158
4.15	Number of errors as a function of the difficulty of the fragments. .	163
4.16	Number of errors as a function of the age.	164
4.17	Number of errors as a function of level of studies.	165
4.18	Average success rate for $1R$ rules using a single feature.	175
4.18	Average success rate for $1R$ rules using a single feature (cont.).	176
4.19	Average success and standard deviation for $1R$ rules for selected descriptors.	177
4.20	Values for \bar{z} for each descriptor	180
4.21	Recognition percentage in the $\omega\delta$ -space	182
4.22	Bayes recognition results for the different models versus the window width, with a fixed $\delta = 1$	183
4.23	Evolution of k -NN recognition	184
4.24	NN recognition versus window width	185
4.25	Snapshot of a GUI application for music genre recognition. . .	188
4.26	Number of errors made by the different ensembles	197
4.27	Maximum polyphony parameter behaviour	206
4.28	Overview of the cartesian ensemble system	212
5.1	Multi-level music genre recognition system overview.	233
B.1	Predefined fuzzy sets	246
B.1	Predefined fuzzy sets (cont.)	247

List of Tables

1.1	Answers to the question <i>What is melody?</i> in an informal survey.	13
2.1	Confusion matrix for a two class classifier	68
2.2	Symbolic (MIDI) corpora.	72
2.3	Audio music corpora.	73
2.4	JvC1 and JvC2 MIDI file corpora summary.	73
2.5	MIREX 2007 Audio mood classification clusters.	77
3.1	MIDI track descriptors	83
3.2	Melody part selection corpora	88
3.3	Summary of number of tagged melody tracks per song.	88
3.4	Melody versus non-melody classification results.	90
3.5	Melody track categorization error summary.	90
3.6	CL200 classification error examples	91
3.7	JZ200 classification error examples	92
3.8	KR200 classification error examples	93
3.9	Melody track selection success and errors	95
3.10	KR200 melody part selection error examples	96
3.11	Genre specific melody part selection accuracy results	97
3.12	Num. of classification errors in melody part selection	98
3.13	Melody track selection across genres ($p_\epsilon = 0.01$).	99
3.14	Num. of errors in melody part selection across genres	99
3.15	Melody track selection (training w/ all genres)	100
3.16	Num. of errors in melody part selection (training w/ all genres)	100
3.17	Contingency table for a condition X_i	106
3.18	Corpora for rule-based melody track selection.	109
3.19	Rule antecedent pruning results	110
3.20	Rule coverage summary	110
3.21	Melody track categorization. Best results for <i>RWC-G</i>	114
3.22	Melody track categorization. Best results for <i>RWC-P</i>	114
3.23	Melody track categorization. Best results for <i>AJP</i>	115
3.24	Melody track categorization. Best results with $N = 1$	115
3.25	Best melody part selection results for test datasets.	115
3.26	First rules from ALL200, with scoring function s_0	116
3.27	RIPPER setup parameters	118
3.28	RIPPER-ALL200 (crisp) rules.	119
3.29	Average attribute presence frequency in crisp rule systems. . .	121
3.30	Fuzzy linguistic terms	122
3.31	Fuzzy rule system equivalent to the crisp RIPPER-ALL200 . .	129

LIST OF TABLES

3.32	FIS optimization setup parameters	130
3.33	FIS optimization (FISO) experiments	131
3.34	Performance of evolved FIS vs. RIPPER-ALL200 on the ALL200 dataset.	133
3.35	Summary of evolved FIS results on test datasets.	137
3.36	Melody track categorization results.	137
3.37	Melody track categorization. Evolved vs. fixed <i>AvgAbsInterval</i> attribute fuzzy sets.	139
3.38	Track samples correctly classified by the fuzzy system.	141
3.39	Track samples incorrectly classified by the fuzzy system.	142
3.40	Fuzzy rules fired by examples in Table 3.38 and 3.39.	142
4.1	Training parameters for Random vs. Jazz experiment.	150
4.2	SOM parameters for Jazz vs. classical music training.	153
4.3	Statistical profile of the people subjected to the test.	160
4.4	Error percentages in terms of group of people.	162
4.5	Error percentages in terms of the difficulty levels assigned.	163
4.6	Error rates over sliding window lengths from 3 to 16,	166
4.7	Distribution of melody length in bars	169
4.8	Statistical musical descriptors considered.	172
4.9	Best descriptor performances with <i>1R</i> classifier.	174
4.10	Feature selection results	179
4.11	Best success rates	185
4.12	Averages and standard deviations of success rates	186
4.13	Average success rates for whole melody segment length ($\omega = \infty$)	186
4.14	Error rates of ensembles with the UCI/Statlog data sets	191
4.15	Error rates of ensembles with the JvC data sets	193
4.16	Working parameters and accuracy of selected classifiers	196
4.17	Ensemble's performance.	196
4.18	Classifiers ranked by diversity.	198
4.19	Audio datasets.	199
4.20	Audio feature sets used in fusion experiments	201
4.21	Symbolic feature sets used in fusion experiments	202
4.22	Transcriptor maximum polyphony (P) estimation results (sin- gle classifiers)	205
4.23	Transcriptor maximum polyphony (P) estimation results (en- semble)	205
4.24	Accuracy for a SVM trained on different feature sets	207
4.25	Early fusion accuracy for a SVM trained on combined feature sets	208
4.26	MIREX 2007 average raw accuracy summary	208

LIST OF TABLES

4.27	MIREX 2008 average raw accuracy summary	210
4.28	MIREX 2009 ensemble's base classifiers	214
4.29	MIREX 2009 average raw accuracy summary.	214
4.30	Comparison between early and late fusion approaches	216
4.31	Cartesian ensemble accuracy on the LMD dataset	216
4.32	Additional voting methods in the cartesian ensemble.	217
4.33	Best results of base classifiers on different datasets	217
4.34	Results of the ensemble classification on different datasets . . .	219
4.35	Best results on single subspace/classifier combinations with feature selection.	220
4.36	Best results on single subspace/classifier combinations without feature selection.	220
4.37	Feature selection results for the GTZAN corpus.	221
4.38	Ensemble cross-validation execution times	221
4.39	Results of ensemble classification	222
A.1	Model configuration parameter choices available in the genre recognition graphical application.	243

1

Introduction

“A problem is a chance for you to do your best.”
Duke Ellington

1.1 Music information retrieval

Music information retrieval (MIR) is a field of research devoted to the extraction of meaningful information from the content of music sources ([Orio, 2006](#); [Typke et al., 2005](#)). In the application of pattern recognition techniques to MIR, two main folds can be found in the literature: audio information retrieval ([Foote, 1999](#); [Tzanetakis and Cook, 2000b](#)) and symbolic music information retrieval ([Downie, 2003](#)).

In audio information retrieval, the raw digital audio signal is processed. Usually WAV or MP3 files ([Bosi and Goldberg, 2003](#)) are the input to these systems. No explicit information about notes, voices or any musical symbol or tag is encoded in the signal. On the other hand, symbolic MIR is based on processing symbols with direct musical meaning: notes with pitch, duration, dynamics, etc. The most common formats used as input for these systems are ASCII text files like kern, abc, MusicXML ([Good, 2001](#); [Selfridge-Field, 1997](#)), or binary files containing note control information like MIDI files ([Selfridge-Field, 1997](#)). In these formats input data contain information about what and how is to be played, instead of the rendered music itself like in the audio signal. The semantics of both approaches is different, at least at a first stage in information retrieval algorithms. In the case of symbolic processing, information theory or text processing techniques can be applied to process it. On the other hand, raw audio does not contain explicit information about notes or voices, and thus requires to perform signal processing to extract these musical data, a process that almost always introduces noise to the actual musical material found in the audio stream. Currently, some of the most active topics in audio information retrieval have as objective the extraction of such musical information as note onsets,

CHAPTER 1. INTRODUCTION

timbre or voices (Dessein et al., 2010; Holzapfel et al., 2010; Pertusa, 2010). With this preprocessing of the raw audio, many of the work lines found in the symbolic music information retrieval can also be tackled, but with the drawback of having to deal with the possibly noisy musical data extracted.

Other areas of research that deal with different music representations are *visual MIR* and *metadata MIR*. The first is mainly devoted to the task of *optical music recognition*, where digital images of written music are to be converted to some other representation format, often a symbolic one like MIDI or MusicXML (Bellini et al., 2008; Rebelo et al., 2010). The *metadata MIR* has received increasing interest by the MIR community in recent years. The coming up of new ways of interaction between music users, mainly based on metadata such as social tags is the reason behind this (Lamere, 2008a; Levy and Sandler, 2009; Turnbull et al., 2009). Also, the use of lyrics or information about artists, albums, or music pieces found on web pages are sources for metadata MIR approaches (Knees et al., 2004; Mayer et al., 2008b; McKay and Fujinaga, 2007b; Schedl, 2008).

1.1.1 MIR research topics

The following is a non exhaustive list of some important research topics in the MIR community, taken mainly from the call for papers of the *Ninth International Conference on Music Information Retrieval*¹, held in Philadelphia, USA. No distinction is made here between symbolic or audio MIR. References point the reader to sample works on the topic.

Harmony Chord extraction and labeling. Key finding (Gómez, 2006; Lee, 2007; Pardo and Birmingham, 2002).

Melody Melody/motive extraction, selection, segmentation, and similarity (Gomez et al., 2003; Isikhan and Ozcan, 2008; Uitdenbogerd and Zobel, 1998).

Timbre Speech-music separation, instrument identification, sound source separation (Gerhard, 2003; Greco, 2008; Virtanen, 2006).

Music recommendation Playlist generation, music discovering (Celma, 2006; Eck et al., 2007).

Music Visualization, User interfaces intelligent alternative user interfaces to access music collections based on music content or social websites (Lidy and Rauber, 2008; Pampalk et al., 2003).

¹ISMIR 2008: <http://ismir2008.ismir.net/cfp>

1.1. MUSIC INFORMATION RETRIEVAL

Music classification Classification of music based on tags. Tags represent music sets the tagged object pertains to. They often represent broad categories of music, such as mood or genre, but can also refer to something more specific, like composer, performer or even producer. (cf. Content-based similarity) ([Bertin-Mahieux et al., 2010](#); [Conklin, 2009](#); [McKay, 2010](#); [Stamatatos and Kavallieratou, 2004](#)).

Social and music networks listener or artist community building. Music tagging from social data ([Knees et al., 2004](#); [Lamere, 2008a](#); [Levy and Sandler, 2009](#); [Schedl, 2008](#)).

Content-based similarity, categorization and retrieval Music similarity measures, music genre classification, Query-by-humming, Query-by-example, Query-by-rhythm ([Cruz-Alcázar et al., 2003](#); [Selfridge-Field, 1998](#); [Typke, 2007](#)).

Music identification Cover detection, plagiarism detection ([Miotto et al., 2010](#); [Müllensiefen and Pendsch, 2009](#); [Stein et al., 2007](#)).

Computational musicology musicological analysis by means of computers (Schenkerian analysis, Narmour analysis, tonal analysis,...) ([Kirlin and Utgoff, 2008](#); [Leman, 2008](#); [Paulus and Klapuri, 2008](#)) (cf. *Computational music theories, Structural analysis*).

Computational music theories Generative theories (GTTM), Descriptive theories (Schenker, Narmour, Hanson,...) ([De Haas et al., 2009](#); [Gilbert and Conklin, 2007](#); [Hamanaka et al., 2006](#); [Lerdahl et al., 1996](#); [Narmour, 1990a](#))

Optical music recognition (OMR) Extract music data from digital images of written music ([Bainbridge and Bell, 2001](#); [Bellini et al., 2007](#); [Byrd and Schindele, 2006](#)).

Music preservation Preservation and digitalization of old records and printed music ([Ng et al., 2008](#); [Orio et al., 2008](#)).

Score alignment Score following (automatic accompaniment in real time), matching different interpretations of the same music piece ([Arzt et al., 2008](#); [Cont, 2011](#)).

Performance analysis performer identification, educational aid systems, performance haptics analysis and modeling, musical gesture modeling, expressive performance rendering ([Berdahl et al., 2008](#); [Kirke and Miranda, 2009](#)).

CHAPTER 1. INTRODUCTION

Music Summarization thumbnailing, fingerprinting, water-marking music ([Ellis et al., 2010](#); [Meintanis and Shipman, 2008](#))

Data Exchange, Archiving and Evaluation Digital music libraries, music storage formats, music database design, building corpora for evaluation purposes, efficient and effective metadata archiving (mp3 tags, liner notes, critics, etc.), index optimization for music retrieval. ([Lo et al., 2009](#); [Silla Jr. et al., 2008](#))

Musical meaning Perception and cognition based analysis of music, semantic analysis of music utterances. ([Cook, 2001](#); [Scheirer, 1996](#))

Automatic music analysis and transcription onset detection, pitch detection, music transcription ([Klapuri and Davy, 2006](#); [Pertusa, 2010](#)).

Feature representation low and high level feature extraction and representation ([McEnnis et al., 2005](#); [Tzanetakis and Cook, 2000a](#))

Rhythm and meter beat tracking, rhythm pattern detection, meter detection ([Dixon, 2007](#); [Seyerlehner et al., 2008](#)).

Tonal analysis key finding ([Gómez and Herrera, 2004](#); [Rizo et al., 2006a](#)), automatic tonal harmonic analysis ([Illescas et al., 2007](#)).

Digital rights management Music related rights management (ownership, licenses, watermarking, etc.) ([Sinha et al., 2009](#))

Evaluation of MIR systems Evaluation and benchmarking workshops and contests ([MIREX](#); [Orio and Rizo, 2011](#)).

There are some other areas in computer music research that traditionally have not fitted as MIR categories, but are nevertheless related to MIR technologies. The most evident is perhaps *automatic music composition*, that interestingly was one of the first music topics addressed by means of computers. In the 2010 *International Society for Music Information Retrieval* (ISMIR) conference, some assistants expressed publicly their concerns about such computer music field not being considered as part of MIR. The other one research field that is often not considered as part of the MIR area should be *music signal processing*, that is, the research on fundamental methods for low-level feature extraction, domain transformation, sound synthesis, music post-processing, etc.

1.1.2 Symbolic MIR

The goals of symbolic information retrieval can be said to be closer to the actual music theory or musicological analysis than those of audio information retrieval. Some of the most active work areas in the symbolic approach nowadays are listed below:

- Audio to Score Alignment, Score Following: Either aligning a score to an audio signal, or tracking a score using an audio signal as a reference involves symbolic music processing techniques.
- Genre, artist or other tag-based classification: the objective in these tasks is to tell the musical genre, author/performer or whatever tag or set of tags that are to be associated with a given symbolic musical input (often MIDI files).
- Similarity and retrieval: the final target in this workline is to be able to perform a search in a music database to get the most similar pieces to an input query. One of the most tackled problems is the *Query by humming* problem, where the input query is an audio excerpt of a melody sung by the user.
- Cover song identification: the detection of plagiarisms, variations or versions of the same song is the main goal in this case.
- Chord identification: to identify sequences of chords from a flow of single notes, or from a polyphonic music stream.
- Key finding: to guess tonality and key changes of a score from the notes in it.
- Melody identification: to identify the melody line among several MIDI tracks or music staves, as opposite to those that contain accompaniment
- Melody extraction: to extract a monophonic melody from a polyphonic source. The difference with the previous task is that here the music does not come in separate tracks or parts, but in a single polyphonic stream.
- Motive extraction: to find motives (short note sequences) in a score that are the most recurrent, thus acting as the main themes of the song
- Meter detection: to reconstruct the meter from the stream of notes

CHAPTER 1. INTRODUCTION

- Score segmentation: to split the song in parts like musical phrases or sections like verse, chorus, etc.
- Music analysis: to perform musicological analysis for teaching, automatic or computed assisted composition, automatic expressive performance, or build a musical model for other MIR tasks.

1.2 Melody and genre

These are the two musical concepts targeted in this work. They are some of the most vague, ambiguous concepts found in music theory. Very often, their definition and scope is not clearly stated, at least not in a way that is fully computable. Moreover, the meaning of melody is often not exactly the same depending on the musical genre at hand. For example, in one genre, such as pop music, the melody is typically restricted to be played or sung by a single instrument/voice, which is not the case in classical music, for example.

This work deals with the application of pattern recognition techniques to melody selection and genre recognition of symbolically encoded music, using a statistical music description approach in order to capture the computable part of the melody and genre concepts.

The rest of this chapter explores these musical concepts in greater detail, and explains how the pattern recognition approach is undertaken.

1.3 Don't tell me how it sounds, tell me what it plays

The two symbolic MIR tasks described above, melody part selection and genre recognition, are to be approached in the following way: the problem is to be solved in absence of timbre related information. This way, a side goal of the present work will be to provide some clues on whether this kind of tasks can be approached by using only information extracted from the symbolic musical stream, without taking advantage of any metadata available².

As this section title suggests, this work focuses on what the music plays, rather than how it sounds. The quality of sound has much to do with timbre, that is, the instruments actually playing the music, while what is played has to do mainly with the music itself, the note stream, leaving timbre aside.

This quality of the music, timbre, is in general a valuable descriptor when it comes to discriminate musical genre. Suppose we have two melodies

²..., except, of course, the tags needed to apply supervised learning methods.

1.3. DON'T TELL ME HOW IT SOUNDS...

and we know one of them is played by a viola and the other one by an electric guitar with distortion effect. For the first one, we could discard some genres, including those related to modern rock or pop music, with a low probability of getting wrong. For the second one, we could discard, with a similar low error probability, genres related to classical or folk music, for example. Purposely avoiding information about timbre allows for studying intrinsic properties of melodic content, independently of which instrument, at the very end, would be used to perform the music. This is specially relevant to symbolic music formats like MIDI, where switching from one instrument to another does not alter music content, but some control parameters. Thus the symbolic music content remains the same (the score is unaltered), but the final rendering of the encoded music can vary substantially. Related to this, see the work by McKay and Fujinaga ([McKay and Fujinaga, 2005](#)) about the importance of timbre and instrumentation related features for music genre classification of MIDI files. They report up to a 46% relative weight of instrument related features when used in combination with features derived from musical properties such as pitch, rhythm or dynamics.

McKay takes an engineering approach to music genre recognition that takes advantage of all the information usually found in MIDI encoded music, trying to obtain a ‘perfect’, i.e., 100% accurate, system. This has two important differences with respect to the approach in this thesis: first, from the musicological point of view, genre is not associated with musical content only, but rather with the actual information found in a specific format like MIDI, which includes metadata like, e.g., program changes³. Such systems are very sensitive to mere instrumentation or orchestration changes. Second, all available MIDI information (from all tracks) is used to predict the genre of a MIDI encoded piece, which differs from my approach, where mainly melody tracks are sampled for data extraction.

The source materials that are used as symbolic music containers in the present work are also MIDI files. A key property of the so called *Format 1* MIDI files (the most widely used MIDI storage format), is that music is encoded in separate *tracks*. Usually, each instrument taking a role in the music piece being encoded is assigned a different track. However, this is in no way mandatory, so one track could contain music to be played by different instruments at different points in time, or several tracks could be assigned to the same instrument.

³From the MIDI standard viewpoint, *program change* events are considered as *control* events immersed within the normal music stream, rather than metadata. Counter-intuitively, the MIDI standard defines such things like time or key signature, which clearly have a musical meaning, as metaevents. I will not follow this distinction in this work, but I will consider as metadata all kind of MIDI events except note events.

CHAPTER 1. INTRODUCTION

In this dissertation a scientific perspective is taken that, rather than trying to build perfect classification systems, using all the information at hand, investigates whether the musical stream, encoded in a score-like format, conveys enough information to characterize its melodic nature, or its genre, regardless of instrumentation. So, in this work this kind of information is purposely ignored. A situation where a trained individual tries to recognize the genre of several pieces played exclusively on piano could serve as a metaphor for this goal.

The approach to music genre recognition is to rely on one of the most relevant components in a music piece: *the melody*. So a first task to accomplish is the identification of the melody within the content of a music piece. As the music containers are MIDI files, that means to be able to automatically distinguish which track contains the melody, being the rest of the tracks considered as accompaniment. The necessary assumptions to be made will be discussed in chapter 3.

Once the music is separated into melody and accompaniment, the approach to music genre recognition in this work is to segment melody track content in fixed length fragments (named *windows*), and then investigate genre recognition of such fragments, at different lengths, using statistical pattern recognition techniques. The research is then extended to audio genre recognition, by means of automatic transcription of the audio content, where such separation between melody and accompaniment in the resulting symbolic representation does not exist.

The advantage of this approach is that, information on the actual content of the piece with respect to genre can be known in detail. It also makes possible to use such a system online, in a context where the musical input occurs in real time, i.e., it is not previously stored in any place. The level of detail on music content or, put in another way, the amount of music data to be input in real time for the system to respond suggesting genre or style of playing can be tuned adjusting the length of fragments that will be used for genre recognition. Also, such a system should be easily adaptable to different tagging taxonomies, such as artist or social tag based classification. It is also possible to use genre recognition models in genre-oriented automatic composition systems, such as in (Cope, 2001; Cruz-Alcázar and Vidal, 2008; Espí et al., 2007).

The next sections present the task of automatic melody identification and music genre recognition in greater detail.

1.4 Melody part selection

Most of the MIDI or XML music formats available contain music organized in such a way that each voice, instrument, or playing role—in particular the melody—is in some way stored separately from each other. Let's name this kind of music storage formats as *multi-part music files*. Most parts, apart from the melody, play an accompaniment role. In particular, a standard MIDI file is usually structured as a number of tracks, one for each voice⁴ in a music piece. At least one of them usually contains a *melody*.

One of the tasks faced in this thesis is to automatically find that melody track in a multi-part music file using statistical properties of the musical content and pattern recognition techniques. The information used to take decisions is based on how the notes are arranged within each voice of a digital score, making the solution independent of specific storage formats. Only the feature extraction front-end would need to be adapted for dealing with different formats.

The identification of the melody track is very useful for a number of applications. For example, in melody matching, when the query is either in symbolic format (Uitdenbogaerd and Zobel, 1999) or in audio format (Ghies et al., 1995). The process can be speeded up if the melody track is known or if there is a way to know which tracks most likely contain the melody, because the query is almost always a melody fragment. Another useful application can be helping motif extraction systems to build music thumbnails of digital scores for music collection indexing.

In the framework of this thesis, the purpose of devising a melody characterization model is to help a music genre recognition system to automatically separate the melody from the rest of the music content in a multi-part music file. This way, separate genre models for melody and accompaniment can be built and combined when convenient.

1.4.1 What is melody?

Before focusing on the machine learning methodology used to automatically extract the characterization of a *melody*, the musical concept of melody itself needs to be reviewed. I shall begin with a definition attempt:

Melody is a somewhat elusive musical term that often refers to a central part of a music piece that catches most of the listener's attention, and which the rest of music parts are subordinated to.

⁴In this thesis the terms *part*, *track* and *voice* are used interchangeably when referring to multi-part music file content.

CHAPTER 1. INTRODUCTION

This is one of many definitions that can be found in many places, particularly in music theory manuals. Some references are provided below. The variability of these descriptions of the concept can give an idea on the difficulty of the task at hand.

From the music theory point of view, [Toch \(1997\)](#) defines *melody* as

“A succession of different pitch sounds brighten up by the rhythm.”

The verb *brighten* suggests here that this succession of pitches is somehow put ‘in front’ of the music performance taking place. This give some insight on where to look at: pitches and rhythm (note onsets and durations). He also writes

“A melody is a sound sequence with different pitches, in opposition to its simultaneous audition that constitutes what is named as chord”,

which is a more technical definition, but doesn’t help to discriminate between accompaniment and melodic sequences. An interesting property of a melody described by the same author in ([Toch, 1977](#)) is that most melodies have a *wave* shape:

With the combination of ascending and descending scale-segments melody approaches its real nature: the wave line.

The author further elaborates on this, stating the idea that melodies have local maxima and, building on previous higher tones, the melodic sequence arrives at the highest of these maxima, called the *climax*, that often appears near the end of the melody. Therefore, note intervals are suggested as a valuable source of information to characterize melody.

A music dictionary ([Sadie and Grove, 1980](#)) defines melody as

A combination of a pitch series and a rhythm having a clearly defined shape.

However, this is a definition that covers all sorts of monophonic note sequences, not only melodies. Also, it doesn’t put a melody in a context where multiple musical parts play simultaneously, as neither of Toch definitions do.

The music theory literature lacks works about melody in favor of works about counterpoint, harmony, or *form* ([Selfridge-Field, 1998](#)). Besides, the

1.4. MELODY PART SELECTION

concept of melody is dependent on the genre or the cultural convention, an idea that will be incorporated in this research. The most interesting studies about melody have appeared in recent years, mainly influenced by new emerging models like generative grammars ([M. Baroni, 1978](#)), artificial intelligence ([Cope, 1996](#)), and Gestalt and cognitive psychology ([Narmour, 1990b](#)). All these works place effort on understanding the melody in order to generate it automatically.

The types of tracks (or parts) and descriptions of *melody* versus *accompaniment* are posed in ([Selfridge-Field, 1998](#)). The author distinguishes:

- *compound* melodies where there is only a melodic line where some notes are principal, and others tend to accompany, being this case the most frequent in unaccompanied string music.
- *self-accompanying* melodies, where some pitches pertain both to the thematic idea and to the harmonic (or rhythmic) support.
- *submerged* melodies confined to inner voices.
- *roving* melodies, in which the theme migrates from part to part.
- *distributed* melodies, in which the defining notes are divided between parts and the prototype cannot be isolated in a single part.

From the audio processing community, several definitions can be found about what a melody is. Maybe, the most general definition is that of [Levitin \(2002\)](#)⁵:

“melody is an auditory object that emerges from a series of transformations... along the six dimensions: pitch, tempo, timbre, loudness, spatial location, and reverberant environment; sometimes with changes in rhythm; but rarely with changes in contour”.

[Gomez et al. \(2003\)](#) gave a list of mid- and low-level features to describe melodies:

- Melodic attributes derived from numerical analysis of pitch information: number of notes, pitch range, interval distribution, melodic profile, melodic density.
- Melodic attributes derived from musical analysis of the pitch data: key information, scale type information, cadence information.

⁵as cited by ([Kim et al., 2000](#))

CHAPTER 1. INTRODUCTION

- Melodic attributes derived from a structural analysis: motive analysis, repetitions, pattern location, phrase segmentation.

Another attempt to describe a melody can be found in ([Temperley, 2004b](#)). In that book, the author proposes a model of melody perception based on three principles:

- Melodies tend to remain within a narrow pitch range.
- Note-to-note intervals within a melody tend to be small.
- Notes tend to conform to a key profile (a distribution) that depends on the key.

In order to gain more insight on the concept, an informal survey was carried out where the subjects were asked to answer the question *What is a melody?*. Both musicians and non-musicians took part in the survey. Table [1.1](#) shows some of the answers gathered. The following list is a *compendium* of shared melody traits found in those answers:

- (finite) succession of notes
- *cantabile* pitch range
- monophonic
- lead part
- identifies/characterizes the piece, song
- unity
- diversity
- contains repeating patterns
- often linked to text
- done by humans
- understandable, catchy, memorizable by humans

As the reader would have noticed, most of these traits are also present in academic definitions provided above. All these different properties that a melody should have can be used as a reference to build an automatic melody identification system.

In ([Hewlett and Selfridge-Field, 1998](#)), the authors identify melody as hard to model by computers:

1.4. MELODY PART SELECTION

Musicians
<i>The succession of characteristic sounds that make a song, tune or piece easily recognizable.</i>
<i>A succession of sounds (pitches) constituting their own sound plane, which is of principal importance in a musical piece.</i>
<i>A finite set of notes that has its own meaning and character, being accepted and assimilated in an intuitive and natural way by the human being, allowing to distinguish, understand and memorize it in an easy way.</i>
<i>A succession of monodic music figures (with or without sound, that is, including silences) made by humans for humans, with its own meaning and unity. This definition distinguish a melody from animal singing (e.g., a songbird singing), which has no musical meaning.</i>
<i>A succession of sounds distributed in time with its own identity and coherence.</i>
<i>A note sequence, usually ‘cantabile’ (even if transposition is needed), and therefore monophonic, that appropriates the leading role in a music piece, consigning the rest of the music to something called ‘accompaniment’.</i>
Non-musicians
<i>A note sequence played by a principal instrument that stands out among the rest of notes played by other instruments, allowing to identify a music piece.</i>
<i>Set of sounds that make up songs and tunes, easily recognizable by the human being.</i>
<i>The part of a song that a person can remember and reproduce, being thereby monophonic. (Note: unless you come from a very exotic tribe).</i>

Table 1.1: Answers to the question *What is melody?* in an informal survey.

“Melody concept is an amalgamation of what is within the music and what is within our minds. Computers can only address the first.”

The kind of statistical description of music used in this work, tries to capture part of what is computable in a melody, i.e., “...*what is within the music*”, which I believe is a substantial part of what makes melody such a recognizable, yet hard to define, musical entelechy.

1.4.2 State of the art in melody part selection

The automatic selection of melody parts has not been tackled as a main objective in the literature until recently. A related problem that has received great attention in the past is the extraction of melody lines from a polyphonic source. This problem has been approached from at least three different points of view with different understandings of what a melody is. The first approach is melody extraction from a polyphonic *audio* source. For this task it is important to describe the melody in order to leave out those notes that are

CHAPTER 1. INTRODUCTION

not candidates to belong to the melody line (Eggink and Brown, 2004). In the second approach, a melody line (mainly monophonic) must be extracted from a *symbolic* polyphonic source where no notion of *track* is used (Karydis et al., 2007). With this approach, Uitdenbogerd and Zobel (1998) developed four algorithms for detecting the melodic line in polyphonic MIDI files, assuming that a melodic line is a monophonic sequence of notes. These algorithms are based mainly on note pitches; for example, selecting at every time step only the note with the highest pitch (*skyline* algorithm). The third approach focus on how to split a polyphonic source into a number of monophonic sequences by partitioning it into a set of melodies (Marsden, 1992). In general, these works are called monophonic reduction techniques (Lemström and Tarhio, 2000).

There were some early works in the literature addressing the melody part selection problem. Ghias et al. (1995) built a system to process MIDI files extracting a sort of melodic line using simple heuristics. Tang et al. (2000) presented a work where the aim was to propose candidate melody tracks, given a MIDI file. They take decisions based on single features derived from informal assumptions about what a melody track may be.

More recently, in Madsen and Widmer (2007b), the authors propose several methods for measuring complexity in music, based on pitch and duration related features. They use them as a basis for building melody track prediction models. The idea is based on the observation that there seems to be a connection between the complexity of a musical line, and the amount of attention that will be devoted to it on the part of a listener. The assumption is that the melody (or lead instrument) track in a MIDI file will contain the largest amount of information. Global and local measures based on entropy, as well as compression ratio of entire tracks are used as complexity metrics. Prior to computing these complexity measures, tracks are reduced to a monophonic sequence of notes. Local measures are obtained by means of a sliding window that extracts overlapping track segments. The complexity of these segments is computed and the track with the highest complexity is chosen as the ‘winner’ for that window. The track containing the most complex voice for the longest time is predicted as the melody track. Global measures refer to the entire track. The third model for melody track prediction is based on the LZW compression algorithm. The track that is less compressed is predicted as the melody track.

These models are evaluated on two data sets of popular music (*Traditional* and *Modern*), as this genre of music is expected to have the melody constrained to a single track. Baseline for melody track prediction by random guessing is about 15% for the *Traditional* dataset and about 11% for the *Modern* one. The authors report a 52% prediction success on the *Traditional*

1.4. MELODY PART SELECTION

dataset using a local complexity measure based only on inter onset interval information and a window 12 s long. Results for the Modern dataset are reported to be as good as a 62% success using the same measure with a window 6 s long. They also report that comparable results on the Traditional dataset have been achieved also by just selecting the track with the highest average pitch. When dealing with the entire track as a whole, compression based models outperform complexity measure based models, that in turn outperform selecting just the highest pitched voice. Given this results on popular music, there is evidence that a somewhat strong correlation exists between rhythmic complexity and melody perception, as the best results were obtained using measures based on note inter onset interval information.

Based on entropy measures, as in the previous work, (Madsen and Widmer, 2007a) presented an algorithm for predicting the most likely melody note at any point in a piece, using a fixed length sliding window (one to four seconds long) to compute local complexity values for each voice (MIDI channel). Music from Haydn and Mozart –manually annotated– is used as a test corpus. The work showed evidence that melody in classical music is indeed partly definable by complexity. Such measures can therefore help melody understanding in conjunction with other structural and gestalt based measures. Window length is a key parameter of the system presented here. A comprehensive study on these measures as a function of the window length is desirable but were not found in the work of Madsen.

In the context of melody retrieval from polyphonic music, Suyoto and Uitdenbogerd (2008) found evidence that defining duration information as inter-onset interval improves retrieval effectiveness significantly when combined with pitch-based pattern matching, which is not the case when note duration is used instead of inter-onset interval (IOI).

In (Gomez et al., 2003), several melody definitions are gathered from the literature. In these definitions melody is considered an auditory object, a sequence of pitches or a set of attributes. Melody is also associated with the concept of unity and theme, being somewhat merged with the concept of *phrase* or *motive*. This work also presents several methods to represent melody such as using note frequency, pitch, note duration, intervals or melodic contour. Also, the MPEG-7 *Melody Description Scheme* (ISO/IEC, 2005) is outlined, which uses, among others, a 5-step contour representation, in which intervals are quantized to values from -2 to $+2$. It is identified as the only standard that proposes an all-purpose melody description scheme. When dealing with melody as a set of attributes, the authors identify a list of features suitable to characterize melody, such as harmonic features (key, scale, cadences,...), rhythmic features, pitch-based features (number of notes, *tessitura*, interval distribution, melodic profile or melodic density, for

CHAPTER 1. INTRODUCTION

example) and features derived from a structural analysis, such as motive or pattern location and indexing, and phrase segmentation. Perceptual or subjective features such as emotional/textual melodic descriptors are also considered as suitable for melody characterization.

1.5 Music genre recognition

Music genre recognition has been approached by many authors in the MIR community. The task could be defined as the problem of automatically assigning one or more genre tags to a piece or a fragment of music. The common working hypothesis is that music pieces from the same genre should share some common attributes that would permit to correctly classify them into genres. The problem is usually approached by supervised learning techniques, although unsupervised methods have also been used, avoiding the need to formulate a computable definition of what genre is. Genre modeling is a versatile tool that not only allows to perform classification, but also to use automatic composition algorithms to generate new songs in the learnt genres ([Cope, 1996](#); [Cruz-Alcázar and Vidal, 2008](#)).

Most of the research on this topic is based on audio music corpora, and has been taken as a benchmarking problem to demonstrate a wide range of audio feature extraction methods and classification paradigms ([Ahrendt, 2006](#); [Lidy et al., 2010b](#); [Scaringella et al., 2006](#); [Silla and Freitas, 2009](#); [Tzanetakis and Cook, 2002a](#)). Nowadays a number of audio corpora exist that authors regularly use for testing their approaches (c.f. sec. 2.8.4). On the other hand, the field of symbolic music genre recognition has been explored in a more limited way. See section 1.5.3 for some references in this domain. Other approaches to genre recognition are text-oriented, based on lyrics, and using text mining techniques to classify a song according to their lyrics, i.e., without exploring the actual musical content ([Mayer et al., 2008b](#)). Another approach to the problem is to use community meta-data (such as social tags) using web-mining techniques to extract information about an author or song from different websites or forums ([Lamere, 2008b](#); [Levy and Sandler, 2009](#)). Recently, hybrid systems have emerged by combining more than one of the previous approaches ([Lidy et al., 2010a](#); [Mayer et al., 2008a](#); [McKay et al., 2010](#)).

The genre concept naturally induces hierarchies in the organization of music archives. While most of the research done in MIR deals with flat genre taxonomies, some works consider the use of hierarchical genre taxonomies: ([Burred and Lerch, 2003](#)), where audio signals are classified into a 4-level taxonomy that differentiates speech and background noise from music, which

1.5. MUSIC GENRE RECOGNITION

includes a 3-level music genre taxonomy. In (McKay, 2010), the author built a symbolic dataset with 38 leaf node classes and 3 levels of depth.

The main source of symbolic music for genre recognition are MIDI files, usually collected from the Internet. They are by far the more abundant kind of symbolic music format objects available, though some collections in another formats exist, such as the RISM database (Tsou, 2011), which contains metadata and incipits of European music manuscripts from 1600. Other databases exist in the Humdrum *kern* format (Huron, 2002), like the *Essen folksong Collection* (Schaffrath, 1995), or in GUIDO format (Hoos et al., 1998). This kind of symbolic music files are mainly found on academic sites devoted to the cataloging or preservation of classical music. MusicXML (Good and Actor, 2003) is a score-oriented format for encoding symbolic music, with a number of internet sites offering downloadable music in this format.

The choice of MIDI as the source music format in this thesis is due to the above mentioned wide availability of music pieces in this format, ranging from early classical music to modern pop tunes. Rather than a score-oriented format, MIDI was initially devised as a communication protocol between synthesizers, thus being able to represent not only musical notes, but also control events needed for real time interpretation. The standard MIDI file format (MMA, 1990) was created to facilitate the storage of such music interpretations. For this reasons it is considered by some authors as a pseudo-symbolic format. Despite of that, it is widely used for symbolic MIR research, as it is straightforward to separate note events from control events.

The definition of music genre pose several problems for computational approaches (Aucouturier and Pachet, 2003; Lippens et al., 2004). There is a lack of consensus in the definition of genre labels, and important inconsistencies can be found both in the number of labels used and in the way they are used. These problems have recently led research in the field towards the use of user tags to classify music. With the advent of the social network and music recommendation sites in Internet, user tags have almost overtaken the role of the genre as the primary dimension for music classification. Nevertheless, musical genre classifiers are useful tools for the automatic organization of music databases, because genre is still the very first selector used by users to browse present-day music catalogs.

There have been some claims that systems working with symbolic information do not really reflect user's needs, since real-world systems should work with audio databases. This is partially true, because many people store their music files in digital audio format, as well as music retailers do, but, on the other hand, music scholars mainly work with scores, seldom with audio files. Nevertheless, recent advances in polyphonic transcription and chord

CHAPTER 1. INTRODUCTION

recognition from audio files suggest that it is possible to use such algorithms to obtain a symbolic representation of music. Symbolic systems can be then used as a back-end, working with the symbolic sequences obtained from the audio files.

One of the goals in this thesis is to investigate systems able to recognize the musical genre of symbolic music encoded as MIDI files, using statistical features derived from the melodic, harmonic, and rhythmic aspects of music.

1.5.1 What is genre?

Musical genre is a quality of music that most people can perceive intuitively. It is probably the most popular music descriptor, as it is often used to describe, categorize, and even compare songs, albums, or authors. There is not a formal definition of what a musical genre is. Moreover, the terms *genre* and *style* are sometimes treated as synonyms. In (Fabbri, 1999), the author usefully defined genre while distinguishing it from style:

- Genre is “*a kind of music, as it is acknowledged by a community for any reason or purpose or criteria, i.e., a set of musical events whose course is governed by rules (of any kind) accepted by a community*”.
- Style is “*a recurring arrangement of features in musical events which is typical of an individual (composer, performer), a group of musicians, a genre, a place, a period of time*”.

These definitions suggest that genre is a broader concept than style, that involves in its definition the concurrence of an audience that somehow agree in putting under the same term a collection of musical pieces. The boundaries between different genres are imprecise, and the existence of verified ground-truth corpora, i.e., definition by extension, seems the only way to approximatively define these boundaries. It has been suggested that only limited agreement can be achieved among human annotators when classifying music by genre, and that such limits impose an unavoidable ceiling on automatic genre classification performance (Aucouturier and Pachet, 2003; Lippens et al., 2004). In (Sordo et al., 2008) the authors compare genre taxonomies created by experts’ consensus to folksonomies created by the collaborative effort of a community (social tagging). They conclude that experts, wisdom of crowds, and machines agree in the classification and cohesion of some genres (e.g. Blues, Hip-Hop), and clearly disagree in others (e.g. Rock).

Despite the above, in (McKay and Fujinaga, 2006b) the authors explain some arguments in favor of using genre in music classification:

1.5. MUSIC GENRE RECOGNITION

- Genre involves a special emphasis on culturally predetermined classes that makes it worthy of separate attention.
- Genre classification is useful to end users, who are already accustomed to browsing both physical and on-line music collections by genre.
- Genre labels provide a vocabulary that can be used to discuss musical categories.
- Research in automatic genre classification can also provide valuable empirical contributions to the fields of musicology and music theory.

The authors also recommend the development of genre recognition systems able to label different sections of a music piece differently, where windows should be classified individually.

One of the arguments against using genre is that they do change over time. New genres and subgenres appear regularly in the music market, while pre-established genres tend to grow in size and, at the same time, change their meaning. It is commonly thought that as novel musical ideas within a genre become overused, and gradually become part of that genre concept, certain pressure exists for new ideas to develop, leading to the raising of new music genres. This ‘pressure’ comes from different sources. For example, in (Temperley, 2004a), the communicative-pressure principle is discussed and proposed as one of the forces that foster the evolution of musical genres. Temperley argues that *“music functions, at least in part, to convey certain structures to the listener via a surface of notes. The communicative process relies on mutual understanding between producers (composers and performers) and listeners as to how surfaces and structures are related”*. The author of the present dissertation, as a practicing musician, has informally identified at least two other ones. The first one exists on musically demanding genres such as classical music or jazz: as skilled musicians and composers get familiar with the ‘standard’ musical ideas from a genre, they start getting bored with them. This combination of dexterity and boredom leads their activity to innovation based on intellectual challenge, from where this cutting-edge musicians and composers get more fun. The second source of pressure for change exists in not so ‘musically demanding’ genres, the ones where the entertainment industry, notably recording companies and mass-media corporations are the driving force. As the music in these kind of genres is not intrinsically complex or elaborated, at least from the strictly musical point of view –though certainly it is in the more technological aspects of the business as recording, production and selling– the innovation comes from the need to ‘be different’ (as an equivalent to *be new!*) imposed by the

CHAPTER 1. INTRODUCTION

media business. In this line, in (McLeod, 2001), the author suggests that in electronic music, *“the naming of new subgenres can be linked to a variety of influences, such as the rapidly evolving nature of the music, accelerated consumer culture, and the synergy created by record company marketing strategies and music magazine hype”*.

Nevertheless, research on music genre recognition is worth the value, as it provides a workbench where several aspects of MIR can be tested and evaluated. Also, despite all the criticism, musical genres remain a very effective way to describe and tag artists and their music.

1.5.2 Application of music genre recognition systems

Immediate applications are the classification, indexation and content-based search of digital music libraries, where digitized (MP3), sequenced (MIDI) or structurally represented (XML) music can be found. In general, every tag-based organization of music objects is a field of application for such recognition systems.

Other applications include the modeling of user musical taste in order to look for that kind of music over large musical databases. Such a model could be used in cooperation with automatic composition procedures to guide this latter process according to some stylistic profile provided by the user.

Music genre recognition systems can help in the task of authorship attribution. System designed to classify fragments of music pieces can also help in musicological analysis to guide in the search of specific stylistic devices inside digital scores.

1.5.3 State of the art in music genre recognition

In the music genre recognition, genre is not limited to broad categories of music, such as Jazz or Classical music. It can also mean, for example, the identification of particular composers or performers. Also, music classification based on mood tags is a task closely related to genre recognition. Works on music genre recognition can be broadly divided in audio encoding and symbolic encoding based applications. In each of this areas, further categories can be established, based on the kind of features used for classification, or the computational approach taken to perform recognition. Audio based genre classification is not reviewed here, but a good review on these works, although a bit outdated, can be found in (Ahrendt, 2006; Scaringella et al., 2006). The reader can also refer to the ISMIR conference

1.5. MUSIC GENRE RECOGNITION

proceedings⁶, one of the most important conferences in the MIR field, where a fair amount of these works can be found.

Music classification based on symbolic representations include the classification of composers, or authorship attribution (Backer and Kranenburg, 2005; Hontanilla et al., 2011; Margulis et al., 2008; Stein et al., 2007), recognition of mood (Hu et al., 2008), or performance style (Dannenberg et al., 1997; Stamatatos and Kavallieratou, 2004). Works focusing on genre recognition in the symbolic domain are reviewed below, in a more or less chronological order.

In Carpinteiro (1998) the author presents a hierarchical self-organising map (SOM) able to analyze time series of musical events. The model can recognize instances of a reference sequence (a fugue by J.S. Bach) in presence of noise, and even discriminate those instances in a different musical context. In this work, the SOM is used as sequence recogniser, using a time integration mechanism in the input layer of two SOM, arranged one on top of the other, to represent the reference monophonic melodic sequence in order to provide the SOM with the ability of processing time sequences.

In the work by Thom (2000) pitch histograms are used to describe blues fragments of the saxophonist Charlie Parker. The author developed a probabilistic mixture model of variable-sized multinomials, and a procedure that uses this model to learn how to perceive/generate variable-sized histograms. The approach is able to discover powerful musical abstractions when trained on saxophonist Charlie Parker.

(Tzanetakis et al., 2003) perform genre classification based on pitch histograms extracted from both audio and MIDI files. Results from symbolic and audio data are compared. A set of 100 musical pieces in MIDI format, distributed in five genres, and an additional 500 audio pieces generated from MIDI files, plus 500 general audio pieces were used for comparison and evaluation. For classifiers trained on symbolic data, a 50% accuracy is reported for a 5-genre classifier, and a 80% accuracy is obtained when using pair-wise classifiers.

The authors in Shan and Kuo (2003) investigate the mining and classification of music genre by melody from a collection of MIDI music, including New Age songs from Enya, a Beatles album and Chinese folk songs. In their approach they obtain a melody representation by deriving chord n-grams and sequences from the previously extracted melody using several *ad-hoc* heuristics. The method used is associative classification that relate frequent itemsets to classes. They report an average accuracy from

⁶<http://www.ismir.net>

CHAPTER 1. INTRODUCTION

70% to 84% for 2-way classification using derived-chord sequences as melody representations.

A work on melody based genre recognition is found in (Li and Sleep, 2004). The authors empirically demonstrate how dimensionality reduction on combined feature sets can improve classification accuracy. Melodies are extracted from a 4-genre corpus of MIDI files and represented as pitch sequences. Feature sets based on bi-gram statistics and pitch and interval properties are utilized. For feature selection, a PCA reduction followed by an optimal transformation (Duchene and Leclercq, 1988) is applied, and classification is performed by means of SVM. An average 5% improvement on classification accuracy is obtained by combining the original feature sets, reporting an average 73% of success when using a flat genre model.

In (McKay and Fujinaga, 2004), the authors present a system for classification of MIDI files by genre, using a 2-level taxonomy with 3 top genres and 9 leaf genres (the Bodhidharma dataset). They use feedforward neural networks and k-nearest neighbour classifiers, and a genetic algorithm for selecting models through feature selection. They report a 90% success at leaf level and a 98% success at top level, using a five-fold crossvalidation scheme.

(Basili et al., 2004) investigated the impact of different musical features derived from MIDI on the accuracy of a genre recognition system using different machine learning algorithms. Features and classification methods were evaluated on a corpus of 300 MIDI files distributed in six genres.

In (Ruppin and Yeshurun, 2006), the authors combined techniques of selection and extraction of musically invariant features with classification using a compression distance similarity metric, as an approximation of the Kolmogorov complexity. The approach was evaluated on a corpus of 50 MIDI files distributed in 3 genres, reporting up to 85% success using a pitch/time second derivative feature.

(Karydis, 2006) presented a music genre classification system that relies on note pitch and duration features, derived from histograms. They concluded that adding features based on note duration improve the classification success of a system relying solely on pitch derived features.

In (DeCoro et al., 2007), the authors applied a Bayesian framework to combine, or aggregate, a hierarchy of multiple binary classifiers, and consequently improve performance over previous results on classifying the Bodhidharma 2-level genre taxonomy.

The work presented in (Cataltepe et al., 2007) reported findings on using MIDI files and audio features from MIDI, separately and combined together, for MIDI music genre classification. They computed distances between MIDI files using the normalized compression distance (NCD), an approximation to

1.5. MUSIC GENRE RECOGNITION

Kolmogorov complexity. Audio features were extracted from rendered MIDI files. They evaluated their approach on the Bodhidharma dataset, getting a 93% success rate when combining NCD and classification based on audio features.

In (Pérez-Sancho et al., 2008), the authors presented a genre classification framework for audio music based on a symbolic classification system. Audio signals are transformed to a symbolic representation of harmony using a chord transcription algorithm, by computing Harmonic Pitch Class Profiles. Then, language models built from a ground-truth of chord progressions for each genre were used to perform classification. A related work (Perez-Sancho et al., 2009) modeled chord progressions as n -grams and strings, and then applied perplexity and Naive Bayes classifiers in order to model how often those structures are found in the target genres. The approach was evaluated on a 2-level corpus consisting of 761 MIDI files distributed in 8 leaf sub-genres and 3 top genres, using different textual chord representations. They reported a 85% success at top level and a 50% success at leaf level, and indicated that errors at leaf level occur mostly within top level genres.

In (Cruz-Alcázar and Vidal, 2008), three different classification methods were used: two grammatical inference algorithms (ECGI and k-TSI), and language modeling using n -gram models. These methods were tested with two corpora of musical genres, containing three and four different styles respectively, and several encoding formats. The best results in classification were obtained with the n -gram models, reaching a 98.3% classification rate with the three-classes corpus, and a 99.5% with the four-classes corpus.

The use of entropy based measures for the discrimination of different classical music styles was investigated in Margulis et al. (2008). There, the authors considered the use of entropy as a plausible measure that correlates with the *arousal potential* –the psychological strength of a stimulus pattern– as evidenced in some referred works. The entropy of eight musical parameters in different classical music styles was examined. The music corpus used for evaluation is composed of eight different repertoires of classical music. Some interesting findings were reported, such as that average first-order and normalized entropy for each repertoire from the 17th and 18th centuries across all parameters was significantly correlated with the order of composer birth dates.

In (Conklin, 2009) a method for symbolic music genre classification based on a feature set pattern representation is proposed. An ensemble of sequential patterns is used to classify unseen pieces using a decision list method. Results on a small corpus of 195 folk song melodies are reported.

CHAPTER 1. INTRODUCTION

Recently, some works have explored the combination of features derived from the symbolic, audio, text, and metadata domains to improve performance in genre classification. For example, in (McKay et al., 2010) the authors investigated the genre classification utility of combining features extracted from lyrical, audio, symbolic and cultural sources of musical information. They found that cultural features extracted from web data and listener tags were particularly effective, while they also found that features derived from lyrics were less effective than the rest.

It must be noted that it is rather difficult to compare these works because most of them use a different data set for their experiments. There are many reasons for this situation. First, the different conceptions of musical genre make it impossible to compare works dealing with different tasks, such as genre or composer classification. However, even when the same task is carried out, most authors have gathered their own data sets in different file formats, because there is not any standardized, symbolic corpus for testing genre classification algorithms. Music files are usually copyrighted material, and this has prevented the MIR community from the creation of any publicly available data set, although some attempts have been done in this direction, such as the RWC database (Goto, 2004) or the Music Information Retrieval Evaluation eXchange (MIREX)⁷.

From all the works dealing with symbolic music sequences, two differentiated groups can be found according to the way music is described. In the first group we can find those works that use a shallow description of musical content, describing music using statistical features computed from the notes. In the other group are the works that focus on the local relationships of notes, using the music sequence itself as the input to the classifier. Both approaches are useful to analyze music at different levels – global and local – and are thus complementary. (Zanette, 2008) claims that music perception takes into account statistical properties of music, as the outcome of the composition process is an ordered sequence of events conveying information, like a message. So statistical tools can be used to study musical qualities. Thus, it seems reasonable to think that statistical pattern recognition techniques can find enough information in music objects to model the genre of a set of musical pieces.

⁷http://www.music-ir.org/mirex/wiki/MIREX_HOME

1.6 Brief outline of this thesis

As described in sections 1.2 and 1.3, this thesis tackles two MIR related problems: melody part selection and music genre recognition using symbolic music sources. These tasks are to be approached relying only on music content (mainly notes) found on symbolic music objects like MIDI files. This implies that information related to timbre is not taken into account, which is an important characteristic of this work.

The rest of this manuscript is structured as follows:

Chapter 2: Technical background. Pattern recognition methods, experiment design techniques and performance evaluation methods and materials used in this thesis are reviewed.

Chapter 3: Melody part selection. Approaches using decision trees, rule systems and fuzzy systems are discussed, and experimental results are evaluated.

Chapter 4: Music genre recognition. Exploration of the statistical feature space by SOMs is presented. Results from a human recognition survey are discussed. Supervised learning experiments, by single classifiers, ensembles, and combination of audio and symbolic data are presented and discussed.

Chapter 5: Summary and future perspectives Conclusions and contributions of this thesis are summarized here. Future research lines are outlined.

2

Technical background

“All musicians are subconsciously mathematicians.”
Thelonious Monk

During the development of this work, many machine learning techniques have been used or tested, from feature selection to performance evaluation. Learning schemes from different paradigms have been applied, both as single models or as ensembles, in order to discover which techniques are best suited to deal with symbolic music information described by statistical features. In particular, classifier ensembles have been targeted that use a variety of base models and feature sets. The classical train-and-test learning methodology has been used throughout the research, including feature selection and performance evaluation techniques like crossvalidation or leave-one-out estimation methods. This chapter describes all the schemes, methods, and metrics used elsewhere. Also, experiment design is addressed briefly, and pre-existing materials used in this research are described. This section can be largely overlooked by those familiar with machine learning.

2.1 Unsupervised learning

Unsupervised learning (or clustering) is the process of learning from unlabeled data. There are some reasons for interest in unsupervised learning. For example, in some problems, collecting and labeling large sets of samples can be very costly. Also, sometimes it is interesting to train first a model with unlabeled data, then use supervision to label the groupings found, as we shall do when applying self-organising maps to a music genre recognition task. Finally, in early stages of a research, it may be valuable to do some exploratory data analysis to gain some insight into the structure of the data at hand. We shall discuss here the unsupervised learning techniques used in this thesis, mainly the self-organising map or SOM.

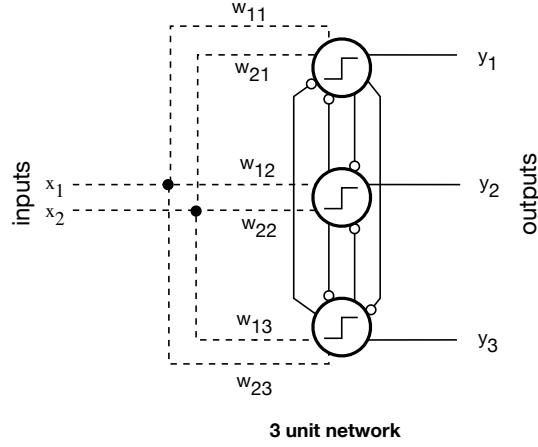


Figure 2.1: A competitive net for processing data grouped in three clusters.

2.1.1 Unsupervised competitive learning

Competitive learning, a type of unsupervised learning, is a method used by some kinds of neural networks (Freeman and Skapura, 1991). In the presence of input signals, the units in the output layer ‘compete’ for being the winning unit, the one with the highest excitation level. Once the output value of each unit is calculated, only the one with the highest value would be activated. This is known as the *winner-takes-all* learning strategy.

An n -dimensional input vector $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ is processed by as many units as clusters in the input space to be identified. Only the unit with the highest excitation value would be activated with a value equal to one, thus assigning a particular cluster to the input data:

$$y_i = \begin{cases} 1 & \text{if } \sum w_{ij}x_j \geq \sum w_{kj}x_j, \forall k \\ 0 & \text{otherwise} \end{cases} \quad (2.1)$$

where w_{ij} is the weight for the connection between input x_j and unit i .

Figure 2.1 illustrates the design of a simple net of three competitive units able to identify clusters in a two-dimensional input space. Deciding which is the winning unit relies on global information about each unit state. This is represented in the figure as links between output units. A signal on one of these links stronger than the unit activation value will make the unit become inhibited, being its activation value equal to zero, as equation 2.1 states.

Competitive learning algorithm

Suppose that a set of n -dimensional normalized vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ is going to be classified in k clusters. The network consists of k units, n inputs, k weight vectors fully connecting input with units.

The competitive learning algorithm works by attracting weight vectors \mathbf{w}_i towards the clusters. It iterates randomly on \mathcal{X} , selecting an input sample \mathbf{x}_i at each iteration. As the input vectors and weight vectors are normalized, their dot product $\mathbf{w}_i \mathbf{x}_j$ equals the cosine of the angle between the vectors:

$$\mathbf{w}_i \mathbf{x}_j = \|\mathbf{w}_i\| \|\mathbf{x}_j\| \cos \alpha = \cos \alpha$$

A *test* step chooses \mathbf{w}_m , the weight vector to update, to be the one whose dot product is the maximum. This is the weight vector closest to the input sample \mathbf{x}_j .

An *update* step moves \mathbf{w}_m towards the direction of \mathbf{x}_j . Several alternative update rules can be used:

- *Learning coefficient update*: $\Delta \mathbf{w}_m = \eta \mathbf{x}_j$, $\eta \in [0..1]$. η decreases toward zero at each iteration.
- *Update by difference*: $\Delta \mathbf{w}_m = \eta(\mathbf{x}_j - \mathbf{w}_m)$
- *Batch update*: Weight updates are accumulated. Weights are modified only at certain epochs.

Finally, the algorithm uses a stop criterion to end learning. This criterion could be set in several ways, typically setting a maximum number of update iterations to be performed.

Convergence analysis

For the competitive learning method to converge it is necessary that the target clusters satisfy certain conditions. A solution is stable if weight vectors remain ‘inside’ the clusters they represent. The solution is unstable if the clusters overlap or they are very large, preventing the algorithm to find a stable solution.

Suppose $P = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m\}$ is a set of vectors in an n -dimensional space, located in the same semi-space. Then the set of vectors $\mathbf{x} = \alpha_1 \mathbf{p}_1 + \alpha_2 \mathbf{p}_2 + \dots + \alpha_m \mathbf{p}_m$ with $\alpha_i > 0$ is the **cone** defined by P .

A cone defined by a cluster contains all the vectors falling ‘inside’ the cluster. The angular diameter of a cone defined by normalized vectors is proportional to the wider angle found between two vectors in the cluster. A

sufficient condition to reach a stable solution is that the angular diameter of the cluster cones must be smaller than the distance between them. In this case a stable solution exists where weight vectors ‘fall’ inside each cluster cone and remain there.

2.1.2 Self-organising maps (Kohonen networks)

A neural network whose inputs are real values computes a function

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m.$$

The domain of f can be represented by a self-organising map (SOM), or Kohonen network, (Freeman and Skapura, 1991; Kohonen, 1990), a particular kind of unsupervised competitive net. When an input vector in a particular region is selected, only one SOM unit is activated. This one is known as the *Best Matching Unit*, *BMU*, the one with the maximum activation value.

SOMs are neural methods able to obtain approximate projections of high-dimensional data distributions in low-dimensional spaces, usually bidimensional. Different clusters in the input data can be located in the built neural map. These clusters can be semantically labelled to characterize the training data and also hopefully future new inputs.

Given the problem of mapping an n -dimensional space using an unidimensional vector of Kohonen units, the goal is for each unit to specialize in a particular area of the input space. When the network is fed with inputs from such an area, the corresponding *BMU* will compute the highest excitation value.

A Kohonen unit computes the distance between an input vector \mathbf{x} and its weight vector \mathbf{w} . Any metric can be used to compute such distance. The euclidean distance is the most commonly used.

In its simpler configuration, a SOM is organized as an unidimensional array of units. All units located r positions to the right or to the left of an unit make up the *neighbourhood* of radius r of this unit.

The SOM learning algorithm uses a neighborhood function $\phi(i, k, t)$. It represents the coupling strength between *BMU* i and unit k at iteration t of the training process. Only weight vectors of units in the neighborhood of the *BMU* are updated at each new input. The neighborhood function value is usually a function of the topological distance between the units:

$$\phi(i, k, t) = \phi(\|r_i - r_k\|, t)$$

where $r_i, r_k \in \mathbb{N}^D$ are the coordinates of units i and k in the map, respectively. Thanks to the competitive learning rules and the neighborhood

2.1. UNSUPERVISED LEARNING

function, the SOM behaves as an elastic surface that adapts itself to the input data domain. The *stiffness* of this surface is determined by the average radius and form of the ϕ function.

A simple choice is to define $\phi(i, k, t) = 1$ for all i units whose distance to k is less or equal to r and $\phi(i, k, t) = 0$ for other units. As it is desirable for the amount of adaptation of weight vectors to decrease with time, it is usual to select $\phi(i, k, t) = \alpha(t)$ for units within radius r , where $\alpha(t)$ is a monotonic decreasing function ($0 < \alpha(t) < 1$). This type of neighborhood function is known as *bubble* neighborhood.

Another type of commonly used neighborhood function is defined in terms of the Gaussian function:

$$\phi(i, k, t) = \alpha(t) \exp\left(-\frac{\|r_i - r_k\|^2}{2\sigma^2(t)}\right)$$

where $\alpha(t)$ is the same coefficient as in the bubble neighborhood. The standard deviation of the gaussian function, $\sigma(t)$, defines the width of the neighborhood, similar to the bubble radius r . Here, $\alpha(t)$ and $\sigma(t)$ are both monotonic decreasing functions. Their precise form is not important. They can be linear respect to the number of iterations, defined as

$$\alpha(t) = \alpha(0)(1 - t/T)$$

where T is the maximum number of iterations for the training process. They can also be defined as functions of *inverse time*:

$$\alpha(t) = A/(B + t)$$

where A and B are constants. This kind of time-decreasing function is adequate when working with very large maps and long training cycles, because it allows for a fine adjustment of the weight vectors. The correct values for this functions and their parameters are defined empirically.

The neighborhood radius value for a bubble-like neighborhood function usually decreases linearly to one during training:

$$r(t) = 1 + (r(0) - 1)(1 - t/T)$$

Figure 2.2 shows the influence area of the neighborhood function at initial time t_0 , at $t > t_0$ and at last iteration when $t = T$. The xy plane represents a bidimensional SOM. The neighborhood radius decreases as t increases, and so does the learning coefficient, represented as the distance between the top discs representing neighborhood area and the map.

Next, the SOM training algorithm is presented.

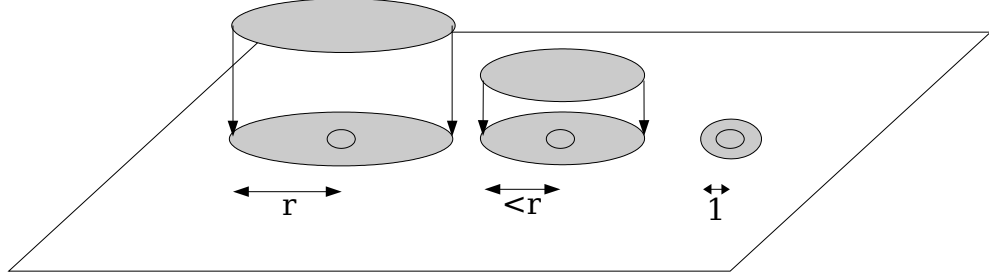


Figure 2.2: Neighborhood and learning rate evolution during SOM training.

General SOM training algorithm.

start Select initial values for $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_m$ randomly. Define the initial radius r , and a neighborhood ϕ with a learning rate α .

step1 Select an input vector \mathbf{x} according to a given probability distribution.

step2 Select the *BMU* m (i.e., the unit closer to \mathbf{x}).

step3 Update the weight vectors using the neighborhood function and the following update rule:

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \phi(m, k, t)(\mathbf{x}(t) - \mathbf{w}_k(t)), \quad \forall 1 \leq k \leq M$$

where M is the number of units in the map.

step4 If the maximum number of iterations T has been reached then **stop**; else modify ϕ as defined and return to **step1**.

The update rule in step 3 attracts the weight vector of unit m in the direction of \mathbf{x} . The weight vectors in its neighborhood are also attracted in the same direction, though not quite as much as for unit m . The euclidian distance is used here to measure the distance between \mathbf{x} and the weight vectors. During learning, the neighborhood size and the value of ϕ gradually decrease, reducing the influence of a unit on its neighborhood.

The convergence of the learning process is influenced by many factors, such as the form of the neighborhood function, the neighborhood size and the way it is modified during the process. This could lead to specific problems like the formation of knots of units with virtually the same weight vectors or units that are excited with similar input vectors in different areas of the

2.1. UNSUPERVISED LEARNING

map. It becomes difficult to ‘undo’ these knots if the network plasticity has decreased to a very low value. As there exists no general convergence proof for multidimensional SOMs, best practice recommends the use of neighborhood functions and learning parameters that are known to lead the learning process to convergence.

SOM convergence analysis

A neighborhood function like

$$\phi(i, k) = \begin{cases} 1 & \text{if } i = k \\ 1/2 & \text{if } i = k \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

produces a concentration of units around the center of the input distribution. The hard coupling of the units attracts them towards this center. The correct learning strategy consists of starting the learning process with a hard coupling, and reduce it gradually as the learning progresses.

In the bidimensional SOM case, this strategy makes the network units to concentrate at the center. However, the distribution periphery also attracts the units, unfolding the network and helping to achieve convergence. If it is empirically known that training data are centered in the coordinates origin, initializing network weights with small values helps backing this learning process.

The parameters for the neighborhood function are searched in a way that a threshold is obtained (unidimensional case), from which stable states of the network can be reached. In the n -dimensional case an adequate neighborhood function, a good initialization and an adjusting process for the α and r parameters are chosen in order to avoid the network to freeze¹. In case the convergence process gets stuck, it is advisable to reset training with initial random weights, and also try either a greater neighborhood radius or a lesser decreasing ratio for the learning coefficient.

Bidimensional self-organising maps

Self-organising maps can be arranged as multidimensional lattices. A very useful configuration for visualization purposes consists in organising the units in the map in a bidimensional plane, in such a way that a neighborhood of radius r of a unit includes all units within a distance r from the unit.

A specific advantage of bidimensional SOMs is its competence to produce effective visualization of the clustering they produced. The units can be

¹A networks freezes when its weight vectors do not update, or they do it too soon

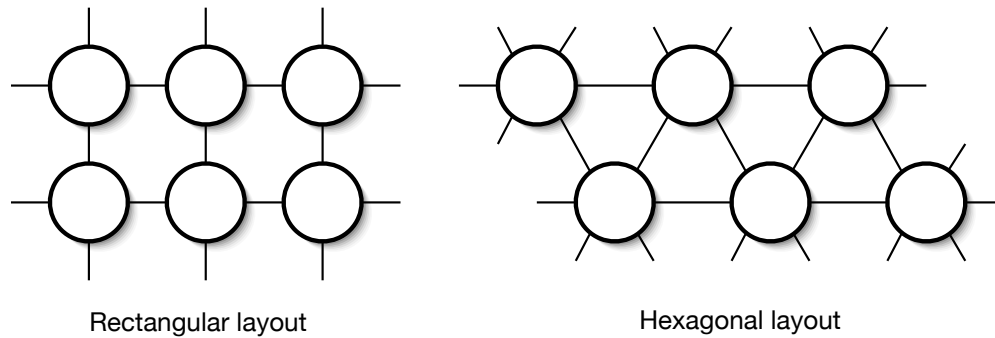


Figure 2.3: Two usual unit configuration in a bidimensional SOM.

interconnected in a rectangular or hexagonal layout (Figure 2.3), being the hexagonal one more visually appealing. The visualization shows a non-linear projection of the probability density function of n -dimensional input data into a bidimensional space, that is the result of the SOM learning process.

After the training stage, a new input vector is assigned to the map unit whose weight vector is closer to it. Any metrics can be used to measure the distance between input vectors and weight vectors. In this work the euclidean distance is used to select the winning unit, as described in the general algorithm presented in section 2.1.2. An input vector activates only the unit whose weight vector is closer to it.

An optimal correspondence between the bidimensional map and the probability density function of the input data $p(x)$ that preserve the local structures of $p(x)$ is highly desirable –the map projection of $p(x)$ can be thought as a pressed flower. In most situations the exact form of $p(x)$ need not to be known, especially when x is many-dimensional. It is more important to automatically find those dimensions and subspaces of the input domain where a significant number of input values concentrates.

Map calibration

A map is calibrated by labeling map areas with labels corresponding to certain input classes. This is achieved feeding the trained map with a set of labeled input samples. Units being activated by certain samples are labeled with these samples' class. Each unit can be tagged with several labels, being usually one of them the most frequent label. Ties can happen, however. Once calibrated, the SOM can be used for classification purposes. The most frequent label will be the one assigned to input samples firing the unit.

Map visualization

SOMs are usually visualized using the U-map representation, where the units are displayed as hexagons with a dot or label in their centre. The grey level of unlabeled hexagons between units represents the distance between neighbor units (the clearer the closer they are). For labelled units, grey level is an average of the neighbor distances to this unit. This way, clear zones are clusters of units, sharing similar weight vectors. The labels are a result of calibrating the map with a series of test samples and indicate the class of samples that activates more frequently each unit. Fig. 2.4 shows an example of a U-matrix visualization of a SOM.

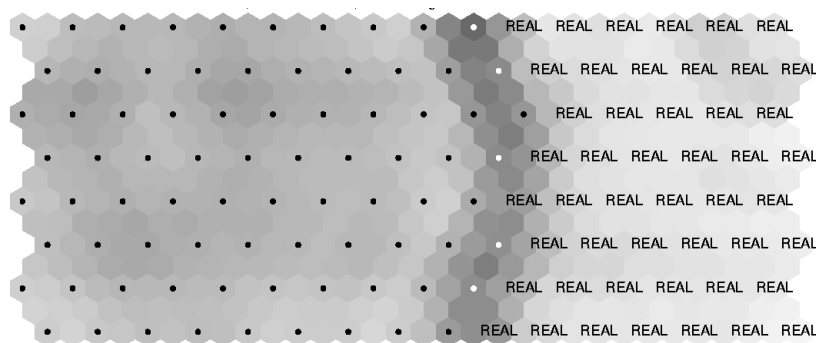


Figure 2.4: Example of U-matrix SOM visualization. The map has been calibrated using samples tagged REAL. They form a cluster on the right of the map, while units on the left are left untagged.

Another form of map visualization is the Sammon projection ([Sammon, 1969](#)), able to map n -dimensional data to a lower- (usually two-) dimensional space in such a way that the inherent data‘structure’ is approximately preserved. Fig. 2.5 shows the same map as in Fig. 2.4 using the Sammon projection.

SOM parameter selection

Map shape An hexagonal lattice is to be preferred for visual inspection. The edges of the map ought to be rather rectangular than square, because the network of weight vectors must be roughly oriented along the major dimensions of the input distribution $p(x)$, in order to stabilize during the learning process. Therefore, visual inspection of the input data distribution, e.g. by Sammon mapping, can help finding suitable map dimensions.

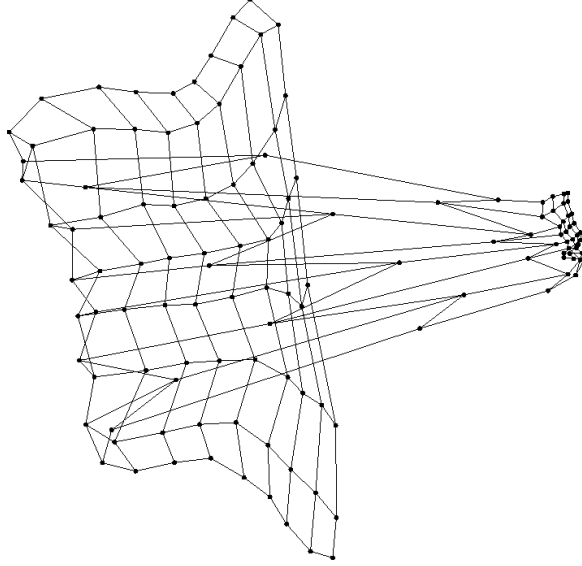


Figure 2.5: Example of Sammon projection of the map shown in Fig. 2.4. Lattice nodes correspond to map units. The more compact node cluster corresponds to the area tagged REAL in that figure.

Learning with few training samples When the number of training samples is rather small, a cyclic training can be performed, where each training sample is used several times. Several alternatives exist: samples may be applied cyclically or in a randomly permuted order, or randomly selecting input samples. In practice, cyclic application order is not significantly worse than the other, mathematically better justifiable, alternatives.

Quality of learning It may be obvious that an optimum map configuration should exist for a given input dataset. When it comes to compare maps trained with the same input data and using the same neighborhood function, the best one (that closer to the optimum) would be the one whose *average quantization error* is smaller. The *average quantization error*, or *Q error*, for short, is the mean of $\|\mathbf{x} - \mathbf{w}_i\|$ for all training samples, where \mathbf{x} is the input sample and \mathbf{w}_i its BMU weight vector. This measure is obtained inputting the training data once again to the already trained network. Therefore, an appreciable number of random initializations of the initial weight values, or *trials*, ought to be tried, then selecting the map with the lower Q error.

It is non-sense to compare the Q error of maps with different neighborhood functions, as this value is minimum when $\phi(i, k) = \delta(i, k)$ (Kronecker delta function). However, the map would have not auto-organising power with such a function. In general, the Q error depends strongly on the neighborhood function.

Missing input sample components It is still possible to train a SOM with incomplete data. The SOM implementation used in this work can deal with missing input sample components by computing distance calculation and weight vector modification steps using available input components.

SOM implementation The SOM implementation used in this work is the *SOMPAK* package (Kohonen et al., 1995). It comes with utilities for training and calibrating maps, then use them to cluster or classify new data. It also includes U-mat and Sammon visualization utilities, as well as tools for visualizing selected weight vectors components (called *planes*).

2.2 Supervised learning

2.2.1 Bayesian classifier

The Bayesian classifier is a supervised parametric classifier (Duda et al., 2000). For a problem of c classes it computes a set of discriminant functions $g_i(\mathbf{x}), i = 1, \dots, c$. It assigns a feature vector \mathbf{x} to class ω_i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \text{for all } j \neq i. \quad (2.2)$$

$$(2.3)$$

For minimum-error-rate classification, $g_i(\mathbf{x}) = P(\omega_i|\mathbf{x})$ is taken. Applying the *Bayes rule*

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)\pi_i}{\sum_{j=1}^c p(\mathbf{x}|\omega_j)\pi_j} \quad (2.4)$$

where π_i are the priors of each class. Taking logarithms, and discarding the denominator, since it is the same for all classes, we can write

$$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln \pi_i \quad (2.5)$$

which is a convenient form of the discriminant function. Often, the densities $p(\mathbf{x}|\omega_i)$ are assumed to be multivariate normal—that is, $p(\mathbf{x}|\omega_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$.

CHAPTER 2. TECHNICAL BACKGROUND

In the general case, where the covariance matrices Σ_i are different for each class, the resulting discriminant functions are quadratic:

$$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + w_{i0}, \quad (2.6)$$

where

$$\mathbf{W}_i = -\frac{1}{2} \Sigma_i^{-1}, \quad (2.7)$$

$$\mathbf{w}_i = \Sigma_i^{-1} \boldsymbol{\mu}_i, \quad (2.8)$$

and an independent term

$$w_{i0} = -\frac{1}{2} \boldsymbol{\mu}_i^t \Sigma_i^{-1} \boldsymbol{\mu}_i - \frac{1}{2} \ln |\Sigma_i| + \ln \pi_i. \quad (2.9)$$

Covariance matrix, mean vectors, and priors can be estimated by maximum likelihood as follows:

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{N} \sum_{j=1}^N \mathbf{x}_j \quad (2.10)$$

$$\hat{\Sigma}_i = \frac{1}{N} \sum_{j=1}^N (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \quad (2.11)$$

$$\hat{\pi}_i = \frac{n_i}{N} \quad (2.12)$$

where N is the number of samples in a conveniently tagged training dataset, and n_i is the number of samples from class ω_i .

Naïve Bayes classifier

The above assumptions lead to a classifier with up to $(n^2 + 4)c/2$ parameters to estimate: Σ_i , $\boldsymbol{\mu}_i$, and π_i , for all $i = 1 \dots c$. Moreover, in multivariate classification situations, with different covariance matrices, problems may occur in the quadratic Bayesian classifier when any of the matrices Σ_i is singular. This usually happens when there are not enough data to obtain efficient estimative for the covariance matrices $\Sigma_i, i = 1, 2, \dots, c$. A straight forward modification of the general case, often used in classification problems, is to assume that features are independent of each other, so that Σ_i is diagonal. This both reduces the number of parameters to estimate, making the classifier more robust, while eluding the potential singular matrix problem. In this case, a so called *Naïve Bayes* classifier can be constructed, where

$$p(\mathbf{x}|\omega_i) = \prod_j p(x_j|\omega_i). \quad (2.13)$$

This assumption doesn't hold for most problems. However, despite its simplicity, it has been shown that such a classifier can obtain near optimal classification errors ([Domingos and Pazzani, 1997](#)).

2.2.2 Nearest neighbour classifier

Also called the k -nearest neighbour, or k -NN classifier, this is one of the most frequently used non-parametric (or lazy) learning algorithms ([Aha et al., 1991](#)). It is trained by just storing training instances, and optionally do some pre-computation that usually involves computing distance matrices. A test instance may then be classified by calculating the distance to surrounding training instances, and tagging it with the most frequent class label from the k nearest training instances. Others strategies for assigning a class label to a test instance could be used, like performing a weighted vote based on distances to each of the neighbours.

In order to compute distances between points in feature space, several choices exist. The Euclidean distance is the most commonly used. Other alternatives such as Manhattan distance, or tangent distance ([Duda et al., 2000](#)) can be applied, depending on the training data. In order to prevent features with large values dominate distance measurements relative to features with small values, features are often normalized or standardized before storing them.

The choice of the number of neighbours k is important. It is often chosen to be odd in order to prevent the risk of ties. When the number of training samples is high, a higher value of k can be appropriate. An often used value is the square root of the number of training samples. If k is too large, the probability of including points of different classes increases. If it is too small, outliers or noisy training samples can easily distort the classifier outcomes. When $k = 1$, the classifier is called simply a nearest neighbour classifier

While its training phase is usually very fast, the disadvantage of k -NN is that, like other lazy learners, classifications can be computationally expensive to perform. A straightforward implementation would have a $O(n^2)$ time complexity, where n is the number of training instances, as the distances from the test point to all training points must be computed. However, a variety of more efficient implementations are widespread available ([Gómez-Ballester et al., 2006](#); [Moreno-Seco et al., 2000, 2003](#)).

2.2.3 Support Vector Machines

Support vector machines (SVM) are a type of discriminant-based classifiers that rely on preprocessing input data to represent samples in a higher dimension, where they will be linearly separable. With an appropriate non-linear mapping to a sufficiently high dimension, data from two classes can always be separated by an hyperplane. SVMs are designed specifically for two-class problems, but they can be generalized to K -class problems by defining K two-class problems, each one separating one class from all others.

Each pattern \mathbf{x}_k can be transformed by a nonlinear mapping ϕ to a higher dimension space:

$$\mathbf{y}_k = \phi(\mathbf{x}_k) \quad (2.14)$$

where $\phi()$ is also called a *basis* or *kernel* function. The first step in building a SVM is to choose an appropriate kernel for the problem at hand. In many cases, polynomials, Gaussians, or other basis functions like sigmoids, are used. The dimensionality of the mapped space can be arbitrarily high. A linear discriminant in this space is

$$g(\mathbf{y}) = \mathbf{w}^t \mathbf{y} + b. \quad (2.15)$$

Let $z_k = \pm 1$, according to whether sample k is in ω_1 or ω_2 . Thus a separating hyperplane ensures

$$z_k g(\mathbf{y}_k) > 0, \quad (2.16)$$

as shown in Figure 2.6. The distance ρ from the hyperplane to the instances closest to it on either side is called the *margin*, and the optimal separating hyperplane is the one that maximizes the margin. The goal of training a SVM is to find this hyperplane. It is expected that the larger the margin, the better generalization of the classifier. Assuming that a positive margin ρ exists, then

$$\frac{z_k g(\mathbf{y}_k)}{\|\mathbf{w}\|} \geq \rho, \quad k = 1, \dots, n, \quad (2.17)$$

and the goal is to find the weight vector \mathbf{w} that maximizes ρ . To ensure uniqueness of the solution, the additional constraint $\rho \|\mathbf{w}\| = 1$ is imposed, which means that the solution should also minimize $\|\mathbf{w}^2\|$. The common formulation of this optimization problem is to fix ρ and solve

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (2.18)$$

$$\text{subject to } z_k(\mathbf{w}^t \mathbf{y}_k + b) \geq 1 \quad \forall k = 1, \dots, n. \quad (2.19)$$

The *support vectors* are the training samples for which Eq. 2.17 is an equality. That is, the support vectors are the training samples lying on the margin, and so they constraint its width, as they are the closest points to the hyperplane.

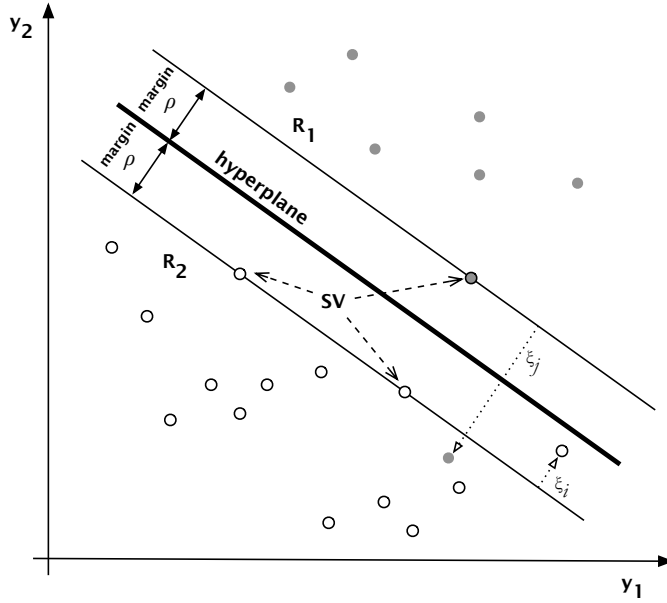


Figure 2.6: Optimal hyperplane in a two-class problem projection. SVs are support vectors lying on the maximum margin hyperplanes.

Suppose that the data are not separable in the mapped space, which depends on the mapping function used, and the presence of outliers. Then Eq. 2.18 has no solution. The constraints can be relaxed by introducing positive *slack variables* associated to certain training samples \mathbf{y}_i :

$$z_i(\mathbf{w}^t \mathbf{y}_i + b) \geq 1 - \xi_i. \quad (2.20)$$

If $0 < \xi_i < 1$ the margin is not satisfied, but \mathbf{y}_i is still correctly classified, as in Figure 2.6. If $\xi_j > 1$, then the sample is misclassified. Slack variables are introduced as a *margin error* term that penalizes the objective function:

$$\min_{\mathbf{w}, b, \xi_i} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i. \quad (2.21)$$

where the parameter C is a positive constant balancing the objective of maximizing the margin and minimizing the margin error.

Several alternatives exist for training a SVM. One of the most popular, implemented in the WEKA toolkit, and used in this work is the *sequential minimal optimization* (SMO) algorithm (Platt, 1999). From the abstract of this reference, we can read

Training a Support Vector Machine (SVM) requires the solution of a very large quadratic programming (QP) optimization problem. SMO breaks this large QP problem into a series of smallest possible QP problems. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop.

An important benefit of the SVM approach is that the complexity of the resulting classifier is characterized by the number of support vectors rather than the dimensionality of the transformed space. As a result, SVMs tend to be less prone to problems of overfitting than other methods.

2.2.4 Multilayer Perceptron

A multilayer perceptron is a kind of *feed-forward neural network*. This is a class of classifiers inspired by the model of a neuron from the organic brain. They are composed of units called *nodes* or *neurons*, that are interconnected by weighted links, as depicted in Figure 2.7. Concepts are learned by modifying the values of these weights. The nodes are organized into layers. The nodes x_i in the *input layer* are provided with input data. The z_k nodes in the *output layer* provide the output of the network when an input instance is present at the input layer. Usually, when a neural network is used for classification purposes, each output node is associated with each possible class. This kind of two-layer networks are called *perceptrons*. They are only capable of learning linearly separable classes. However, if one or more *hidden layers* of nodes are added in between the input and output layers, the resulting *multilayer perceptron* is capable of learning non-linear relations.

Each node in the hidden and output layers typically computes a function f of its inputs that produces an *activation* output. Such function is chosen as to be a continuous non-linear function. Often, a ‘S’ shaped, or *sigmoid* function is used, like the hyperbolic tangent, or the *logistic* function,

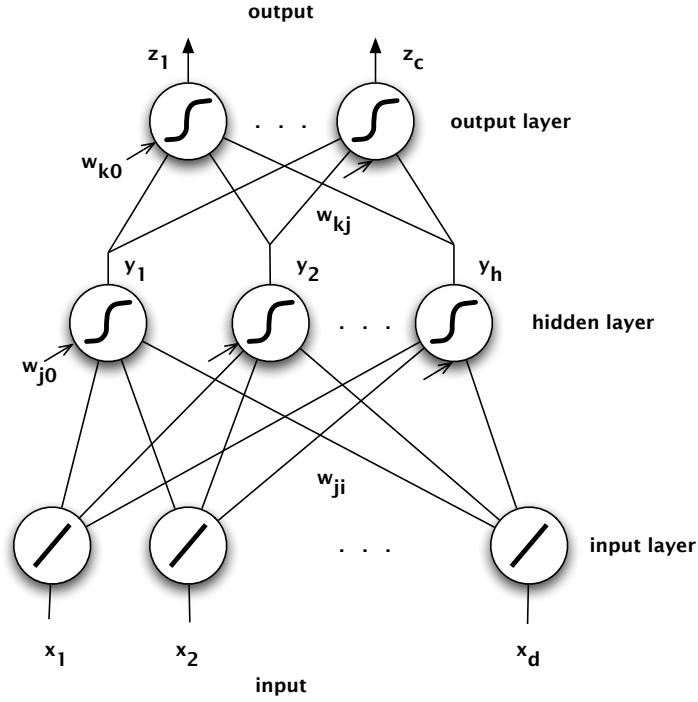


Figure 2.7: A fully connected three-layer perceptron.

$$\text{logistic}(x) = \frac{1}{1 + e^{-x}}, \quad (2.22)$$

shown in Figure 2.8. Some networks also include a bias input to hidden or output units through a weighted link, w_{j0} and w_{k0} . The signal from each output unit is a non-linear discriminant function $g_k(\mathbf{x})$, expressed as

$$g_k(\mathbf{x}) \equiv z_k = f \left(\sum_{j=1}^h w_{kj} f \left(\sum_{i=1}^d w_{ji} x_i + w_{j0} \right) + w_{k0} \right). \quad (2.23)$$

Backpropagation learning algorithm

The method most frequently used to learn the weights of the network is the *backpropagation algorithm*. It is a supervised method, that aims at minimizing the error in the output units, by adjusting the link weights between layers. The basic learning process is to start with an untrained network, with possibly random weights, present a training sample at a time, and compare the network output vector \mathbf{z} with a target output vector \mathbf{t} . For

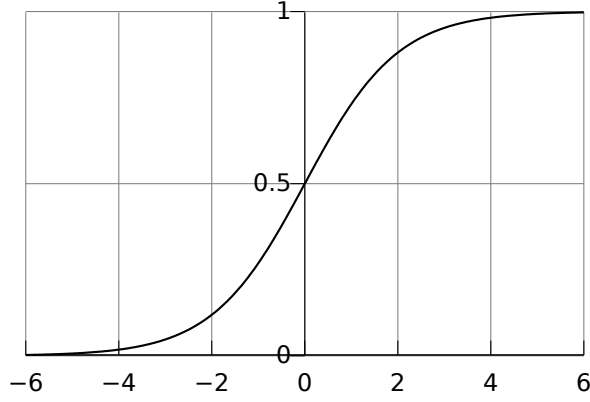


Figure 2.8: The logistic function, often used as the non-linear activation function of a perceptron unit.

the sake of simplicity, consider a three-layer perceptron without bias inputs. The *training error* is computed as

$$J(\mathbf{w}) = \frac{1}{2} \|\mathbf{t} - \mathbf{z}\|^2. \quad (2.24)$$

The backpropagation learning rule is based on gradient descent. The network weights are updated as to reduce the training error:

$$\Delta \mathbf{w} = -\eta \frac{\partial J}{\partial \mathbf{w}}, \quad (2.25)$$

where η is the *learning rate*. It controls how large the adjustments to the weights are during each training iteration. Weight vectors are therefore updated at each sample input, or iteration m

$$\mathbf{w}(m+1) = \mathbf{w}(m) + \Delta \mathbf{w}(m). \quad (2.26)$$

Applying Eq. 2.24, the learning rule for hidden-to-output weights is

$$\Delta w_{kj} = \eta \delta_k y_j, \quad (2.27)$$

where δ_k is called the *sensitivity* of output unit k , and it is defined as

$$\delta_k = (t_k - z_k) f'(net_k) \quad (2.28)$$

where $f(net_k) = z_k$. Similarly, the sensitivity for a hidden unit y_j is defined as

$$\delta_j = f'(net_j) \sum_{k=1}^c w_{kj} \delta_k \quad (2.29)$$

where $f(\text{net}_j) = y_j$. This propagates the training error in the output back to the hidden layer, and implicitly computes an effective target activation value for each hidden unit. Thus, the learning rule for the input-to-hidden weights is

$$\Delta w_{ji} = \eta \delta_j x_i. \quad (2.30)$$

There is a variety of training protocols for applying the backpropagation algorithm. Given a multilayer network structure, they usually involve an initialization step, setting up how training samples are presented to the network, and specifying some stopping criterion. The latter usually involves setting up a threshold θ on the training error, and stop training when the change in the criterion function $J(\mathbf{w})$ is smaller than θ .

A three-layer network can learn any continuous function from input to output, given sufficient number of hidden units, proper non-linearities and weights. In particular, any posterior probabilities can be represented by a three-layer net.

2.2.5 Decision trees and rule learning methods

Decision trees

Decision trees, when used as classification tools, represent discrete-valued functions. They can be re-represented as sets of if-then rules to improve human readability. They are very popular inductive inference algorithms, applied to a wide variety of tasks.

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides a discrete value for the instance, thus effectively classifying it. At each tree node, a test on some attribute of the instance is performed, each branch descending from that node corresponding to one of the possible values of the attribute. If the attribute is not discrete, the test is often an inequality splitting the attribute's range in two descending branches. In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the root to a leaf is a conjunction of attribute tests, being the tree itself a disjunction of these conjunctions.

Several algorithms for learning decision trees exist. Here we describe the *CART* (Classification And Regression Trees) ([Duda et al., 2000](#)) methodology, which is the one used in the random forest classifier described in section 2.2.5. CART provides a general framework that can be instantiated in various ways to produce different decision tree learning algorithms.

CHAPTER 2. TECHNICAL BACKGROUND

Every decision tree learning algorithm progressively splits the training set of samples into smaller and smaller subsets. Each splitting correspond to a new node in the tree. In the ideal case, a perfect split will left samples from different classes in different subsets. This kind of subsets are called *pure* subsets. In practice, mixed classes are found in a subset, thus a decision tree learning algorithm should use a criterion on whether it should stop splitting at a given point or select another attribute and grow the tree further. The CART approach considers a number of decisions to be made in order to instantiate a decision tree learning algorithm:

How many splits will there be at a node? It strongly depends on the attribute used for splitting. Often, a node is split in two branches, resulting in binary decision trees.

Which attribute(s) should be tested at a given node? Fundamentally, the attribute and the test performed on it should produce immediate descendent nodes as pure as possible. Several different mathematical measures of node impurity exist that have basically the same behavior: node impurity should be 0 when all samples reaching that node are from the same class. On the other side, it should be maximum when all classes are equally represented in the sample subset reaching the node. *Entropy impurity* is the most popular measure:

$$i(N) = \sum_j P(\omega_j) \log_2 P(\omega_j),$$

where N is a node and $P(\omega_i)$ is the proportion of samples from class ω_i at node N . The drop in node impurity in a binary tree is defined by

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R), \quad (2.31)$$

where N_L and N_R are the left and right descendent nodes and P_L is the proportion of samples at node N that will go to N_L . The best test is the one maximizing $\Delta i(N)$. The impurity reduction (when using entropy impurity) is the information gain obtained when splitting node N . Single attribute based tests are often used, as they are easier to compute. This way, splitting hyperplanes are perpendicular to axes in the input space, creating *regions* that are assigned to particular classes. The optimization of Eq. 2.31 is performed locally. So, the impurity reduction is a *greedy* method and reaching a global optimum is not guaranteed.

When to stop splitting Growing a binary decision tree until each leaf has the lowest possible impurity typically leads to overfitting the training data. On the other hand, stopping splitting too early will degrade classification power. Several different stop criteria can be used: maximum tree depth is established prior to training, or a node is declared a leaf when the classification error in a validation set is minimized. Another criteria for stop splitting could be that the number of samples reaching a node is below a given threshold. Likewise, a small threshold can be set on node impurity reduction. A node is declared a leaf if its best candidate split reduces node impurity by less than this threshold.

Pruning the tree Pruning is the alternative to stopped splitting. In pruning, trees are grown fully, until node impurity at leaves is minimum. All sibling leaves are then considered for *merging*, that is, if their elimination produces a small increase in impurity, they are deleted. The benefits of pruning is that it avoids the *horizon effect* present in every greedy method of search.

Labeling leaf nodes Given that a leaf node is pure, it gets the class label of the training samples reaching this node. However, when a leaf node is not pure, it is labeled by the class that has most training samples represented at the leaf.

Dealing with missing attributes Samples with missing attributes are called *deficient samples*. A straightforward strategy is to simply delete them from the training data, but this is quite wasteful. Another technique is to ignore samples with missing values when computing node impurity on a given attribute. When it comes to classifying deficient test samples, one approach is to use the test at a node whenever possible but to use an alternative test whenever a test sample is missing that attribute. For this to be possible, during training each non-leaf node is assigned an ordered set of *surrogate tests*, in such a way that the sample's missing value is replaced by the value of the non-missing attribute most strongly correlated with it. Another technique consists of assigning most likely values (*virtual values*) to missing attributes.

The random forest classifier

Random forests (RF) ([Breiman, 2001](#)) are weighed combinations of decision trees that use a random selection of features to build the decision taken

CHAPTER 2. TECHNICAL BACKGROUND

at each node. This classifier has shown good performance compared to other classifier ensembles with a high robustness with respect to noise. One forest consists of K trees. Each tree is built to maximum size using CART methodology without pruning. Therefore, each leaf on the tree corresponds to a single class. The number F of randomly selected features to split on the training set at each node is fixed for all the trees. After the trees have grown, a new sample \mathbf{x} is classified by each tree and their results are combined., giving as a result a membership probability for each class ω_i . Given a RF with K trees, where each tree T_j outputs decision d_j on an input sample \mathbf{x} , this probability $p(\omega_i|\mathbf{x})$

$$p(\omega_i|\mathbf{x}) = \frac{\sum_j \delta(\omega_i, d_j)}{K} \quad (2.32)$$

where

$$\delta(\omega_i, d_j) = \begin{cases} 1 & \text{if } d_j = \omega_i \\ 0 & \text{otherwise} \end{cases} \quad (2.33)$$

RIPPER rule induction algorithm

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) (Cohen, 1995) is a propositional rule learner. It builds a ruleset by repeatedly adding rules to an empty ruleset until all positive examples are covered. Rules are formed by greedily adding conditions to the antecedent of a rule (starting with empty antecedent) until no negative examples are covered. After a ruleset is constructed, an optimization post-pass manipulates the ruleset so as to reduce its size and improve its fit to the training data. A combination of cross-validation and minimum-description length techniques is used to prevent overfitting.

RIPPER is an improvement on a technique called *reduced error pruning* (REP), initially devised to prune decision trees (Quinlan, 1987), later adapted to rule learning systems (Cohen, 1993). It is an *overfit-and-simplify* learning strategy to handle noisy data, that first grows a model that overfits the training data, then simplifies or *prunes* the model. This usually leads to improved error rates on unseen data. Being REP a computationally expensive algorithm, some improvements on it were devised. *Incremental reduced error pruning* (IREP) improves on REP by achieving comparable results much faster. RIPPER is a modification of IREP to increase its generalization power without greatly affecting its computational efficiency. It does so by three modifications: a new metric for guiding the pruning phase,

a new stopping condition for adding rules to a rule set, and a technique for optimizing the rules learned by IREP.

The fact that RIPPER starts considering less prevalent classes first, makes it suitable for some kind of problems with imbalanced data, such as the melody characterization problem studied in section 3.

2.2.6 The *1R* classifier

The *1R* rule learner is a very simple algorithm that proves surprisingly effective on the standard datasets commonly used for evaluation (Holte, 1993). Its aim is to learn a single rule from a training set, that classifies an object on the basis of a single attribute (like a one-level decision tree). The *1R* algorithm chooses the most informative single attribute and bases the rule on this attribute alone. The algorithm assumes that the attributes are discrete. If not, then they must be discretized. Missing values are handled in the algorithm by treating them as a separate value in the enumeration of an attribute.

This is an sketch of the algorithm:

- For each attribute a , form a rule as follows:
 - For each value v from the domain of a ,
 - * Select the set of instances where a has value v .
 - * Let c be the most frequent class in that set.
 - * Add the following clause to the rule for a :
if a has value v then the class is c
 - Calculate the classification accuracy of this rule.
- Use the rule with the highest classification accuracy.

In this thesis, *1R* is used for testing the discrimination power of single features for music genre recognition, so it is used on one-dimensional datasets.

2.3 Information Fusion

The existence of multiple sources of information related to a concept, or multiple viewpoints of a single object, is common place in the multimedia world, where music is not an exception. This naturally leads to the existence of multiple feature spaces conceived to describe the same type of objects. Here comes a set of techniques known as *information fusion*, or more specifically, *data fusion* defined in Wikipedia as

CHAPTER 2. TECHNICAL BACKGROUND

... the use of techniques that combine data from multiple sources and gather that information in order to achieve inferences, which will be more efficient and potentially more accurate than if they were achieved by means of a single source.

Below, two subsets of such techniques are described, *early fusion* and *late fusion*.

2.3.1 Early fusion

A common approach to use multiple feature spaces in a machine learning system is to fuse all them in a single, multimodal feature space. This approach is called *early fusion* and it is defined in (Snoek et al., 2005) as

A fusion scheme that integrates unimodal features before learning concepts.

Figure 2.9 shows a scheme of the method. An advantage of early fusion techniques is that correlation between features extracted from different modalities can be exploited to improve a classification system performance. Also, only one (multimodal) model need to be learned. A disadvantage of the approach is that difficulties to combine features into a common representation could sometimes arise.

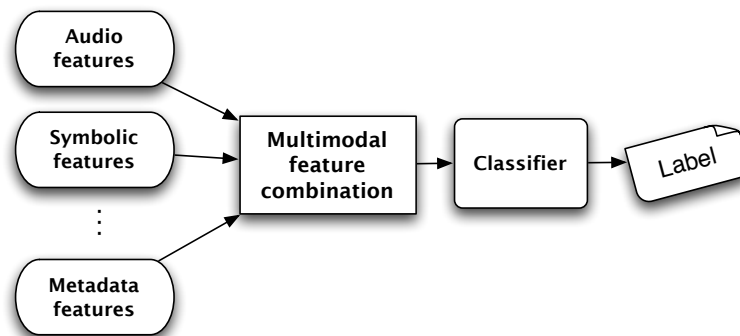


Figure 2.9: An early fusion scheme for music related features.

2.3.2 Late Fusion

The problem arises naturally as a need of improvement of classification rates obtained from individual classifiers. As the limits of existing individual

2.4. ENSEMBLE METHODS

methods are approached and it is hard to develop better ones, the solution of the problem might be just to combine existing well performing methods, hoping that better results will be achieved. This kind of information fusion is called *late fusion* or *decision fusion* and it is defined in (Snoek et al., 2005) as

A fusion scheme that first reduces unimodal features to separately learned concept scores, then these scores are integrated to learn concepts.

i.e., first a set of models is trained (maybe on the same or different feature spaces), and then their individual decisions on new input data are combined in some way to produce a *consensus* decision. Late fusion focuses on the individual strength of single modalities. As several classification schemes can be used to learn from data, there is no need to rely on a particular one, yielding *a priori* more robust systems. A disadvantage of these methods is the potential loss of correlation in a mixed feature space. Also, in general, it requires a greater learning effort than the early fusion approach, where only one model is trained.

Figure 2.10 shows a late fusion scheme. The next section discusses some strategies used in this work to perform the fusion of classifier outputs in a single ensemble outcome.

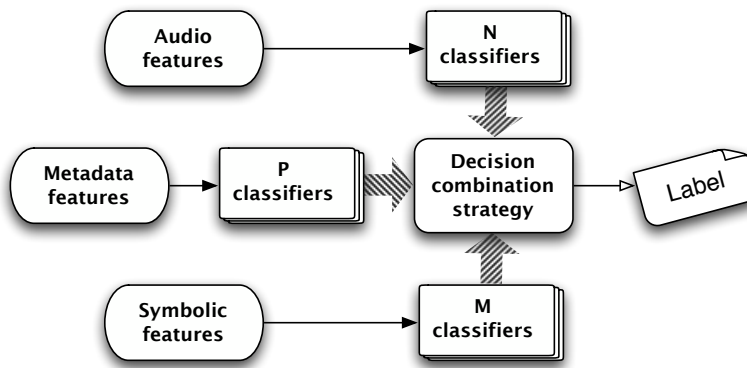


Figure 2.10: A late fusion scheme for music related features.

2.4 Ensemble Methods

Ensemble methods combine decisions from a set of base classifiers, in order to improve both the accuracy and robustness of single classifiers. Works on

this subject point out the importance of the concept of *diversity* in classifier ensembles, with respect to both classifier outputs and structure (Cunningham and Carney, 2000; Dietterich, 2000; Kuncheva, 2003; Partridge and Griffith, 2002). The ensemble methods presented here could be regarded as committees of ‘experts’ (Blum, 1997), in which the decisions of individual classifiers are considered as opinions supported by a measure of confidence, usually related to the accuracy of each classifier. The final classification decision is taken either by majority vote or by a weighing system.

2.4.1 Voting schemes

Designing a suitable method of decision combinations is a key point for the ensemble’s performance. Different possibilities have been explored and compared in this work. In particular, several weighted voting methods, along with the unweighted plurality vote (the most frequent class is the winner class). In the discussion that follows, N stands for the number of samples, contained in the training set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, M is the number of classes in a set $\mathcal{C} = \{c_j\}_{j=1}^M$, and K classifiers, D_k , are utilized.

Several choices are presented below. They are numbered for further reference.

Unweighted methods

1. Plurality vote (PV). Is the simplest method. Just count the number of decisions for each class and assign the sample \mathbf{x}_i to the class c_j that obtained the highest number of votes. The problem here is that all the classifiers have the same ‘authority’ regardless of their respective abilities to classify properly. In terms of weights it can be considered that $w_k = 1/K \forall k$.

Weighted methods

2. Simple weighted vote (SWV). The decision of each classifier, D_k , is weighted according to its estimated accuracy (the proportion of successful classifications, α_k) on the training set (Opitz and Shavlik, 1996). This way, the authority for D_k is just $a_k = \alpha_k$. Then, its weight w_k is:

$$w_k = \frac{a_k}{\sum_l a_l} \quad (2.34)$$

Also for the rest of weighting schemes presented here (except the last one), the weights are the normalized values for a_k , as shown in this equation.

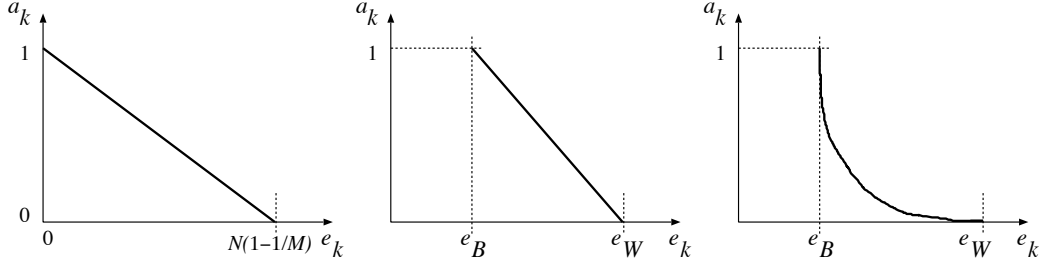


Figure 2.11: Different models for giving the authority (a_k) to each classifier in the ensemble as a function of the number of errors (e_k) made on the training set.

The weak point of this scheme is that an accuracy of 0.5 in a two-class problem still has a fair weight although the classifier is actually unable to predict anything useful. This scheme has been used in other works ([Stamatatos and Widmer, 2002](#)) where the number of classes is rather high. In those conditions this drawback may not be evident. The following methods (numbers 3 to 5) were first proposed in ([Ponce de León et al., 2006](#)), to overcome this limitations. A comparison study between all methods presented here is discussed in section 4.4.

3. Re-scaled weighted vote (RSWV). The idea is to assign a zero weight to classifiers that only give N/M or less correct decisions on the training set, and scale the weight values proportionally. As a consequence, classifiers with an estimated accuracy $\alpha_k \leq 1/M$ are actually removed from the ensemble. The values for the authority are computed according to the line displayed in figure 2.11-left. Thus, if e_k is the number of errors made by D_k , then

$$a_k = \max\left\{0, 1 - \frac{M \cdot e_k}{N \cdot (M - 1)}\right\}$$

4. Best-worst weighted vote (BWWV). In this ensemble, the best and the worst classifiers in the ensemble are identified using their estimated accuracy. A maximum authority, $a_k = 1$, is assigned to the former and a null one, $a_k = 0$, to the latter, being equivalent to remove this classifier from the ensemble. The rest of classifiers are rated linearly between these extremes (see figure 2.11-center). The values for a_k are calculated as follows:

$$a_k = 1 - \frac{e_k - e_B}{e_W - e_B} \quad ,$$

CHAPTER 2. TECHNICAL BACKGROUND

where

$$e_B = \min_k \{e_k\} \quad \text{and} \quad e_W = \max_k \{e_k\}$$

5. Quadratic best-worst weighted vote (QBWWV). In order to give more authority to the opinions given by the most accurate classifiers, the values obtained by the former approach are squared (see figure 2.11-right). This way,

$$a_k = \left(\frac{e_W - e_k}{e_W - e_B} \right)^2 .$$

6. Weighted majority vote (WMV) The theorem 4.1 in (Kuncheva, 2004) states that accuracy of an ensemble of *conditionally independent* classifiers is maximized by assigning weights

$$w_k \propto \log \frac{\alpha_k}{1 - \alpha_k}$$

where α_k is the individual accuracy of the classifier.

Given $\mathbf{s} = [s_1, \dots, s_K]^T$ the vector with the label output of the ensemble, where s_k is the label suggested for a sample \mathbf{x} by classifier D_k , the classifiers in the ensemble are said to have conditional independence if $P(\mathbf{s}|\omega_j) = \prod_{i=1}^K P(s_i|\omega_j)$.

In order to use a voting method of this type as a reference for the previously proposed methods (numbers 3 to 5), in this case the weight of each classifier is computed as:

$$w_k = \log \frac{\alpha_k}{1 - \alpha_k}$$

In order to guarantee minimum classification errors, the prior probabilities for the classes have to be taken into account too. See (Kuncheva, 2004) for a proof. The key conclusion of this theorem is that the optimal weights do not take into account the performance of other members of the ensemble but only magnify the relevance of the individual classifier based on its accuracy.

Classification by weighted methods.

Once the weights for each classifier decision have been computed, the class receiving the highest score in the voting is the final class prediction. If $\hat{c}_k(\mathbf{x})$ is the prediction of D_k for the sample \mathbf{x} , then the prediction of the ensemble can be computed as

$$\hat{c}(\mathbf{x}) = \arg \max_{c_j \in \mathcal{C}} \sum_k w_k \delta(\hat{c}_k(\mathbf{x}), c_j) \quad , \quad (2.35)$$

being $\delta(a, b) = 1$ if $a = b$ and 0 otherwise.

Since the weights represent the normalized authority of each classifier, it follows that $\sum_{k=1}^M w_k = 1$. This makes possible to interpret the sum in Eq. 2.35 as $P(\mathbf{x}|c_j)$, the probability that \mathbf{x} is classified into c_j , and $\hat{c}(\mathbf{x})$ as the class for which this probability is maximum.

2.4.2 Ensemble diversity

Model diversity is a key design factor for building effective classifier ensembles. This has been empirically shown to improve the accuracy of an ensemble over its base models when they are numerous enough. As an informal definition of diversity in an ensemble of classifiers, let us cite Kuncheva in (Kuncheva, 2004):

Intuitively, we want the ensemble members to be as correct as possible, and in case they make errors, these errors should be on different objects.

This method of producing a pool of classifiers followed by a selection procedure to pick the classifiers that are most diverse and accurate is known as *overproduce-and-select* method for ensemble building. For selecting the most diverse models within the ensemble the *Pareto-optimal* selection strategy can be applied in order to discard models not diverse or not accurate enough.

Pareto-optimal selection

Let $A = \{a_1, \dots, a_m\}$ be a set of alternatives characterized by a set of criteria $C = \{C_1, \dots, C_M\}$. The *Pareto-optimal set* $A^* \subseteq A$ contains all non-dominated alternatives. An alternative a_i is non-dominated iff there is no other alternative $a_j \in A, j \neq i$, so that a_j is better than a_i on all criteria.

This strategy for selecting the best set of models is based on rating them in pairs, according to two measures (Kuncheva, 2004). So, in our case the alternatives a_i are pairs of classifiers (or models). The first measure is the *inter-rater agreement* diversity measure κ , defined on the coincidence matrix M of the two models in a pair. The entry $m_{r,s}$ is the proportion of the dataset, which model h_i labels as c_r and model h_j labels as c_s . The agreement between both classifiers is given by

$$\kappa_{ij} = \frac{\sum_k m_{kk} - \text{ABC}}{1 - \text{ABC}} \quad (2.36)$$

where ABC is *agreement-by-chance*

$$ABC = \sum_r \left(\sum_s m_{r,s} \right) \left(\sum_s m_{s,r} \right) \quad (2.37)$$

The second one is the pair average error, computed by

$$e_{ij} = 1 - \frac{\alpha_i + \alpha_j}{2} \quad (2.38)$$

where α_i and α_j are the estimated accuracies of both models. This way, classifiers are selected in pairs. Those with low κ and low average error will be candidates for selection. Figure 2.12 depicts a sketch of a non-dominated pair. The pairs lying inside the grey area would dominate pair $\langle 2, 3 \rangle$, as it would be better than it on all criteria (κ and e , in this case). As there is no such pair, $\langle 2, 3 \rangle$ is in the Pareto-optimal set of selected pairs. The solid line represents the Pareto-optimal set of pairs, while the dashed line represents the kappa-error convex hull set, also used to prune classifier ensembles (Margineantu and Dietterich, 1997). The convex hull is a subset of the Pareto-optimal set.

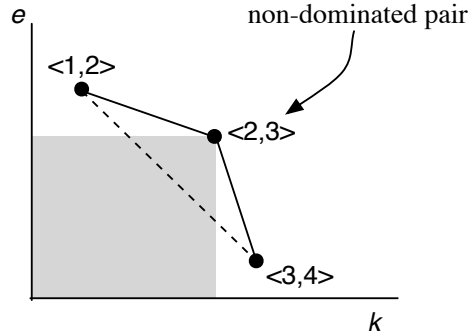


Figure 2.12: A non-dominated pair example in Pareto-optimal selection.

The number of classifiers in the pruned ensemble cannot be specified in advance with Pareto-optimal selection. This lack of control on the ensemble size is seen as a defect of the method.

2.5 Feature selection

Given an *ad hoc* feature set, it is often desirable to know which features are relevant for a particular classification problem, in terms of the classifier's performance. Also, the demand for a large number of samples grows exponentially with the dimensionality of the feature space, a limitation

known as the *curse of dimensionality*. The fundamental reason for this is that high-dimensional functions have the potential to be much more complex than low-dimensional ones, hence harder to learn. Feature selection techniques exist that reduce such large feature spaces by keeping most informative features. Another way to achieve such dimensionality reduction is to apply *feature extraction* techniques, consisting of deriving a smaller set of new features from the original ones. However, usually these techniques do not discard original features as they are still needed to compute the new ones. In this section, feature selection and extraction techniques used elsewhere in this work are presented.

2.5.1 A feature ranking technique

This method is a simple feature ranking technique for two-class problems. It assumes feature independence, that obviously do not hold in most cases, but can help selecting the most *a priori* relevant features. As it doesn't check for correlations, feature redundancy is not detected. The method tests the separability between classes provided by each descriptor independently, and uses this separability to obtain a feature ranking.

Consider that the M features at hand are random variables $\{X_j\}_{j=1}^M$ whose N sample values are those in some dataset. We drop the subindex j for clarity, because all the discussion applies to each feature. The set of N feature values are split into two subsets: $\{X_{1,i}\}_{i=1}^{N_1}$ are the descriptor values for samples from class ω_1 , and $\{X_{2,i}\}_{i=1}^{N_2}$ are those for samples from class ω_2 , X_1 and X_2 are assumed to be independent random variables, since both classes are assumed to follow independent population distributions. We want to know whether these random variables do indeed belong to different distributions. Considering that both sets of values hold normality conditions, and assuming that the variances for X_1 and X_2 are different in general, the test contrasts the null hypothesis $H_0 \equiv \mu_1 = \mu_2$ against $H_1 \equiv \mu_1 \neq \mu_2$. If H_1 is concluded, it is an indication that there is a clear separation between the values of this descriptor for the two classes, so it is an *a priori* good feature for classification. Otherwise, we must conclude that both X_1 and X_2 follow the same data distribution, and this particular feature, taken alone, will not help to discriminate between classes ω_1 and ω_2 .

The Welch's test is used for contrasting the hypothesis. It defines the following statistic:

$$z = \frac{|\bar{X}_1 - \bar{X}_2|}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \quad , \quad (2.39)$$

where \bar{X}_1 and \bar{X}_2 are estimated means, and s_1^2 and s_2^2 are estimated variances for the feature values in a given dataset. The greater the z value is, the wider the separation between both sets of values is for that descriptor. A threshold to decide when H_0 is more likely than H_1 , that is to say, the descriptor passes the test for the given dataset, must be established. This threshold, computed from a t-student distribution with infinite degrees of freedom and a 99.7% confidence interval, is $z = 2.97$. Features with a z value greater than the threshold are kept, and the rest are discarded. Furthermore, the z value permits to rank features tested on the same dataset according to their separation ability.

2.5.2 Fast Correlation-Based Filter

The *fast correlation-based filter (FCBF)* described in (Yu and Liu, 2003) is a feature search method that uses a *symmetrical uncertainty (SU)* correlation-based measure to evaluate features. This measure indicates how much of a feature can be predicted given the information in another feature. The method finds a set of *predominant* features in two steps. First, *relevant* features are ranked according to their *SU* value with respect to the class (SU_c). A threshold δ on SU_c can be established to discard features not enough relevant. In the second step, redundant features are further discarded. A feature F_q with rank q is considered redundant if its *SU* with respect to any feature F_p such that $p < q$ is greater than its SU_c . FCBF has been shown to efficiently achieve high degree of dimensionality reduction for high-dimensional data, while enhancing or maintaining predictive accuracy with selected features.

The correlation measure *SU* between two variables X and Y is defined as

$$SU(X, Y) = 2 \left[\frac{IG(X|Y)}{H(X) + H(Y)} \right], \quad (2.40)$$

where

$$IG(X|Y) = H(X) - H(X|Y), \quad (2.41)$$

where $H(X)$ is the entropy of variable X and $H(X|Y)$ is the entropy of X after observing values of another variable Y , known as *conditional entropy*. They are defined as

$$H(X) = - \sum_i P(x_i) \log_2(P(x_i)), \quad (2.42)$$

$$H(X|Y) = - \sum_j P(y_j) \sum_i P(x_i|y_j) \log_2(P(x_i|y_j)). \quad (2.43)$$

The SU measure is in the range $[0, 1]$, with the value 1 indicating that knowledge of the value of either one completely predicts the value of the other, and the value 0 indicating that X and Y are independent. In experiments reported by the authors in (Yu and Liu, 2004), the relevance threshold δ is heuristically set to be the SU_c value of the $\lfloor N/\log N \rfloor$ -th ranked feature for a given data set. However, when no prior knowledge about the data is available, the authors recommend using $\delta = 0$.

2.5.3 Principal component analysis

A technique that deals with the correlation between features to obtain new, uncorrelated features is Principal Component Analysis (PCA). This is achieved transforming the original feature space of d dimensions into a new space where features are uncorrelated and calculated as linear combinations of the original ones. PCA finds the orthogonal axes of the new space. The first axis is the one where the variability of the original data is maximum, that is, its direction is that of a line in the original feature space where the projection of sample points gives the maximum dispersion. The second axis is one orthogonal to the first where dispersion is maximum, and so on. The transformation can be expressed as

$$Y = W^T X \quad (2.44)$$

where Y are the transformed samples, W is a $d \times d$ orthogonal transformation matrix, and X are the original samples. It can be proven that

$$\mu_Y = W^T \mu_X \quad (2.45)$$

$$\Sigma_Y = W^T \Sigma_X W \quad (2.46)$$

where μ_X, μ_Y are the sample means, and Σ_X, Σ_Y the covariance matrices in the original and transformed space, respectively. Σ_Y is a $d \times d$ diagonal matrix, showing that new space features are uncorrelated. The values λ_i in the diagonal of this matrix are the eigenvalues of Σ_X , and the columns of W are its eigenvectors. Σ_X is a symmetric positive semidefinite matrix, with real eigenvalues. Furthermore, these values are ordered:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \quad (2.47)$$

corresponding to the variance of the transformed samples in each dimension. Thus, the dimension associated with the first eigenvalue has the greater variance, and the first column of W —the eigenvector associated with λ_1 —defines the direction for this dimension. This is the first principal component—the first axis in the transformed space—and it explains the maximum variance direction of the original samples. The same applies to the other transformed dimensions. The axes in the transformed space are ordered by eigenvalue, so selecting only the first k axes that provide a large value of the total original variance, the dimensionality of the original space can be reduced at a low cost. Often, there will be just a few k large eigenvalues, and this implies that k is the inherent dimensionality of the data, while the remaining $d - k$ dimensions generally contain noise.

An interesting point in PCA is that the coefficients of W can be interpreted to understand how original features contribute to the total variance of the samples. Features corresponding to coefficients with larger values are considered to contribute more to the variance explained by the principal component under study.

A drawback of this technique is that, despite possibly $k < d$, in general all the original features still need to be computed, as features in the transformed space are linear combinations of them.

2.6 Fuzzy systems

Fuzzy systems are reasoning systems based on fuzzy logic, a form of many-valued logic derived from the fuzzy set theory. A good introduction to fuzzy logic can be found in (Lee, 1990). In contrast with *crisp logic*, fuzzy logic truth values ranges in degree between 0 and 1. In most fuzzy systems, facts are expressed by *fuzzy variables*, often called *linguistic variables*. Each variable x has a *term set* $T(x)$ over its universe of discourse U , from which it can take values up to a certain degree. For example, if *speed* is a linguistic variable, its term set could be

$$T(\text{speed}) = \{\text{slow}, \text{moderate}, \text{fast}\}$$

where each term is characterized by a *fuzzy set* or *membership function*. Fuzzy sets are sets whose elements have degrees of membership. Thus, for a term t , $\mu_t : U \Rightarrow [0, 1]$ denotes its membership function. This function has often a bell-like, triangular or trapezoidal shape, as in Figure 2.13. In this

example, a speed value of 80 km/h would be considered as *moderate* rather than *fast*.

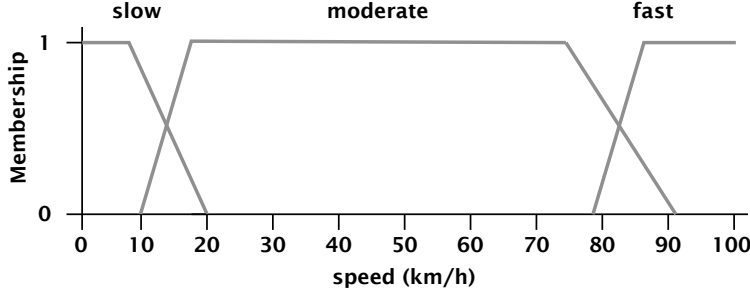


Figure 2.13: Representation of fuzzy speeds by fuzzy sets.

This way, a *fuzzy expression* like *speed IS fast* has a truth value between 0 and 1, depending on the actual numerical value of speed. Expressions are connected by logical operators like \wedge (*AND*), and \vee (*OR*), to express more complex facts. These operations are often implemented as

$$\begin{aligned}\mu_x \wedge \mu_y &= \min(\mu_x, \mu_y) \\ \mu_x \vee \mu_y &= \max(\mu_x, \mu_y)\end{aligned}$$

Other implementations include bounded product, bounded sum, etc. See (Lee, 1990) for a detailed description.

The knowledge about the problem domain is implemented as *fuzzy rules*, where fuzzy expressions connected by logical operators form the antecedent part of the rule. A fuzzy system is said to be a *Mamdani-type* fuzzy system when its rules' consequent is a fuzzy expression itself. Another kind of fuzzy systems are *Takagi-Sugeno* systems, where the consequents are real-valued functions. In this work only Mamdani-type fuzzy systems are used. A fuzzy system contains four modules:

- a *knowledge base* (KB), comprised of a *data base* (DB) and a *rule base* (RB). The DB are the linguistic variables and their associated terms and membership functions. The RB is the set of fuzzy rules defined in the system.
- a *fuzzy inference engine*. The core module that applies fuzzy inference methods to derive a fuzzy consequence from fuzzy inputs.

CHAPTER 2. TECHNICAL BACKGROUND

- an *input fuzzyfication interface*. It converts crisp input values into fuzzy values.
- an *output defuzzyfication interface*. It converts the system fuzzy outcomes into crisp output values.

Let's define a simple problem in order to illustrate how a fuzzy system operates. Consider we want to evaluate the risk of driving a car². Two possible input variables could be the age of the driver and the power of the car engine. The linguistic variables and their terms and associated membership functions are represented in Figure 2.14. Our domain expert has provided us with two fuzzy rules:

```
R1: IF age IS young AND car-power IS high
    THEN risk IS high
R2: IF age IS normal AND car-power IS medium
    THEN risk IS medium
```

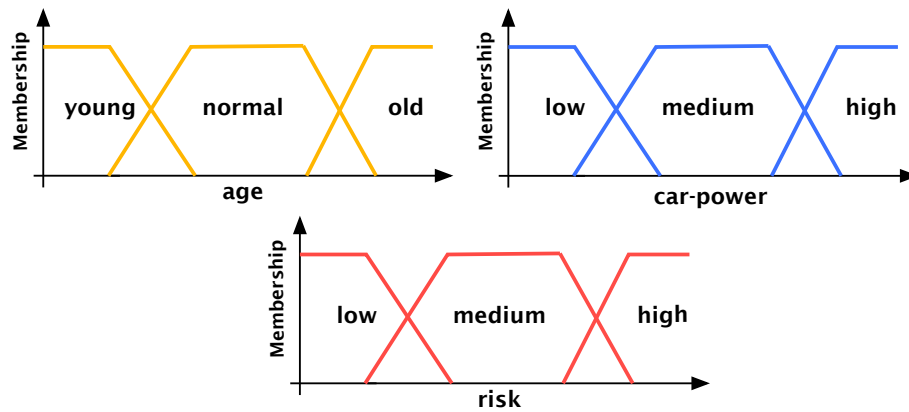


Figure 2.14: Example of fuzzy variables, their linguistic terms and respective membership functions.

The fuzzy system then operates in three steps:

1. *Fuzzyfication step* (Figure 2.15). Measurements of the relevant input variables are converted into appropriate fuzzy sets to express measurement uncertainties by linguistic terms, converting the original crisp variable into a fuzzy variable. For each fuzzy expression (x IS a), $\mu_a(x)$ is computed.

²This example was borrowed from a tutorial in fuzzy logic, part of the *Advanced Course on Knowledge Discovery*, Ljubljana, 2005, by Michael R. Berthold. This tutorial can be found here: http://videlectures.net/acai05_berthold_fl/.

2. *Fuzzy inference* (Figure 2.16). The fuzzyfied measurements are used by the inference engine to evaluate the rules stored in the fuzzy rule base. The result of this evaluation is one or several fuzzy sets defined on the universe of possible outputs. Applying the operators that connect fuzzy expressions in a fuzzy rule antecedent, we obtain the *degree of fulfillment* of the rule. For example, in Figure 2.16, for rule R_1 the AND operator has been implemented by taking $\mu_{R_1} = \min(\mu_{young}(x), \mu_{high}(y))$.

Rule *activation* is the process by which the degree of fulfillment of a rule acts on an output fuzzy set. Activation methods include taking the product or the minimum of the degree of fulfillment and the output membership function. The figure illustrates the minimum as the activation function.

Finally, results from each rule are accumulated. In our example, the maximum of the activations level for both rules is taken as the final fuzzy output value. Other accumulation operators are the bounded sum, or the normalized sum.

3. *Defuzzification* (Figure 2.17). The fuzzy outputs are finally converted into crisp numbers (either a single value or a vector) that, in some sense, is the best representative of the fuzzy output sets. The most commonly used method of defuzzification is the *Center Of Gravity* (COG) that computes the centroid of the area of aggregated output membership functions.

Fuzzy systems have been criticized because in their classical definition they don't make use of training data (Duda et al., 2000), and rely uniquely on human expert's knowledge. Neural networks and genetic algorithms have been used to allow fuzzy systems to learn their KB from training data. In the next section fuzzy systems extended by genetic algorithms are discussed.

2.6.1 Genetic fuzzy systems

A fuzzy system making use of a genetic algorithm (GA) in its design process is called generically a *genetic fuzzy system* (GFS). In this work the use of a GA in a fuzzy system is limited to the definition of the data base. In particular, a GA is used to learn fuzzy membership functions for predefined linguistic variables from training data. The common approach is that an individual in the GA population represents a different DB definition, that is, each chromosome contain a coding of all membership functions giving meaning to the linguistic terms. The degree of adaptation of an individual is

CHAPTER 2. TECHNICAL BACKGROUND

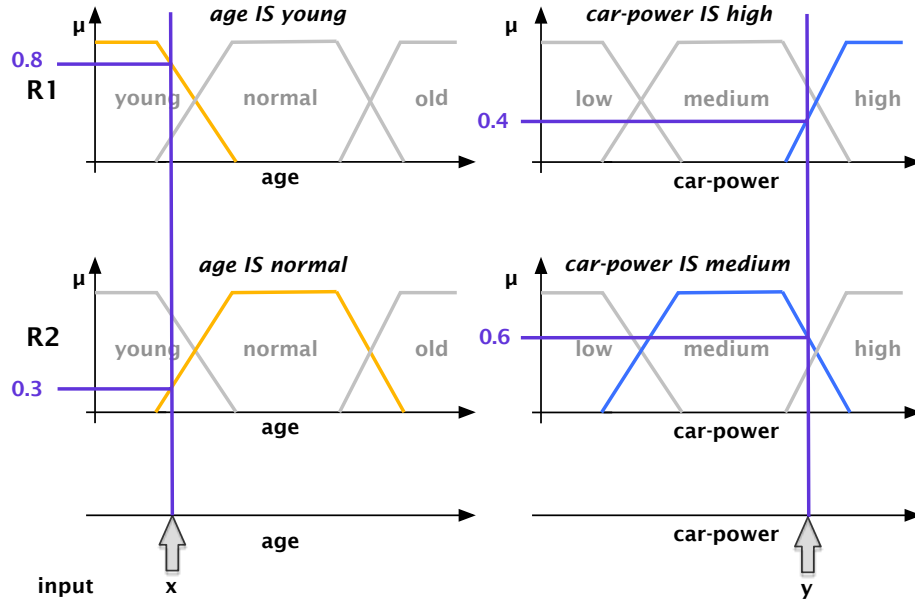


Figure 2.15: The fuzzyfication step. For each fuzzy expression in a rule, the corresponding linguistic term membership function is applied to crisp input values x and y , and a probability for the fuzzy term to be true is obtained.

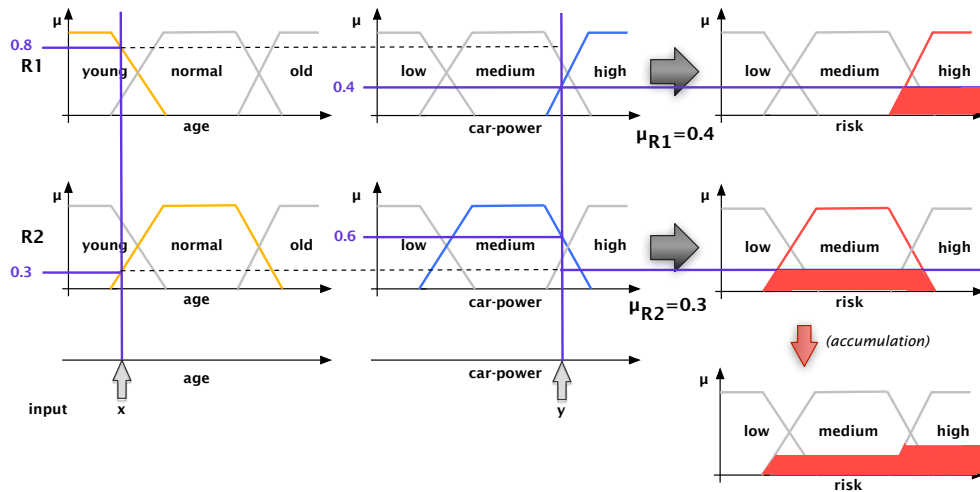


Figure 2.16: Fuzzy inference step example. The minimum (associated to the AND operator) of fuzzy term probabilities is taken as the probability of the rule. Membership functions of linguistic terms in output fuzzy terms are truncated at the rule probability level. The final accumulation step aggregates areas under the truncated output fuzzy sets.

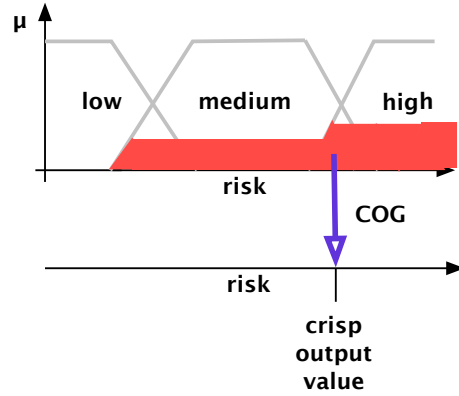


Figure 2.17: Defuzzification example. The *Centre Of Gravity* method is used to obtain a crisp output value.

measured using a fitness function that usually is based on the application of a fuzzy system to a test dataset, using a KB formed by a predefined RB and the DB encoded in the individual's chromosome (Cordón and Herrera, 1995). A method for properly encoding membership functions into a chromosome and decoding them is explained in section 3.3.2.

2.7 Experiment design

2.7.1 Orthogonal arrays

Orthogonal arrays (OA) (Hedayat et al., 1999) are a device often used for experiment design. Frequently, experiments have some free parameters to configure (for example, the number of runs in a GA). The combination of parameter values can lead to a huge number of experiments to be performed, which is often not practical. In this situation, a reduced number of experiments is desirable, while one would like to be sure that the free parameter space is explored in a consistent way. An orthogonal array of m runs, n variables, strength s and level l , denoted $OA(m, n, l, s)$, ensures that, given any $s < n$ parameters with l values each, all their respective values will appear in combination in an equal number of experiments. This avoids testing all possible combinations, while remaining confident that every combination of s parameters appears at least once in some experiment. Here is an example, denoted $OA(8, 5, 2, 2)$:

CHAPTER 2. TECHNICAL BACKGROUND

```
00000
10011
01010
00101
11001
10110
01111
11100
```

This example express the configuration of 8 experiments with 5 free parameters, each of which can take 2 different values (0 and 1, mapping to whatever values make sense for the parameters). Each row corresponds to a different experiment setup, and each column to a different parameter. The strength of the array (2) says that taking any two columns, any combination of 0's and 1's would be tested and equal number of times (two times in this example).

2.8 Performance evaluation

This section reviews which are the evaluation metrics and methods used in this work. It also presents some of the music collections previously available in the literature.

2.8.1 Evaluation metrics

The most widely used metrics to assess the accuracy of a classification method is the *correct classification ratio* or its inverse, the classification *error rate*. This accuracy is usually estimated using different datasets for training and testing the classifier.

Information retrieval metrics

When assessing the performance of a two-way classification method, it becomes relevant the use of information retrieval (IR) metrics. There exist a huge variety of such metrics, aimed at measuring the effectiveness of information retrieval methods, notably those working with text. Such systems are often devoted to the retrieval of relevant documents from a document database, given a query. In the case of a two-way classifier, these metrics can be applied to each class separately, measuring the effectiveness of the classifier in successfully classifying samples of that class. The problem can then be posed in this terms: ‘Retrieve all samples of class X from the

2.8. PERFORMANCE EVALUATION

database'. In this case, the test data are the 'database' from which relevant samples from the given class shall be extracted.

Precision and *Recall* are the most widely used metrics to assess a system's effectiveness in the IR world. They are also fairly well understood quantities. These are their classic definitions:

Precision The proportion of retrieved material that is actually relevant.

Recall The proportion of relevant material actually retrieved in answer to a query.

In terms of measuring the output given by a classifier for a given query class A , we can redefine *precision* as

the proportion of test samples that actually belong to class A
among those predicted as A by the classifier.

Recall can as well be redefined as

the proportion of test samples predicted as A with respect to all
 A samples in the test set.

Precision and recall can also be interpreted as probabilities:

- *Precision* as an estimate of $P(A/C_A)$, the conditional probability that an item will actually belong to class A given that it was predicted to be A .
- *Recall* as an estimate of $P(C_A/A)$, the conditional probability that an item will be labeled as class A given that it actually belongs to class A .

where C_A is the classifier answer and A represent the actual class of the sample.

Both precision and recall can be computed in terms of error and success. Given a two class problem, a confusion matrix can be build. Such a matrix contains information about actual and predicted classifications done by a classifier. Table 2.1 shows the confusion matrix for a two class (A and B) classifier.

The entries in the confusion matrix have the following meaning with respect to class A :

CHAPTER 2. TECHNICAL BACKGROUND

Table 2.1: Confusion matrix for a two class classifier

		predicted	
		A	B
actual	A	TP	FN
	B	FP	TN

true positive (TP) number of *correct* predictions for class A samples.

true negative (TN) number of *correct* predictions for class B samples.

false positive (FP) number of *incorrect* predictions for class B samples.

false negative (FN) number of *incorrect* predictions for class A samples.

Here, we identify class *A* with *positive* samples and class *B* with *negative* ones.

Given the confusion matrix, the precision (P), recall (R), and accuracy (Ac) measures are calculated as in eqs. 2.48 to 2.50:

$$P = \frac{TP}{TP + FP} \quad (2.48)$$

$$R = \frac{TP}{TP + FN} \quad (2.49)$$

$$Ac = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.50)$$

Note that both P and R are a per-class measure, while Ac is the accuracy of the model being evaluated.

The effectiveness of the system determined by P , R , and Ac may not be an adequate performance measure when the number of negative instances is much greater than the number of positive instances. Suppose there are 1000 instances, 995 of which are negative instances and 5 of which are positive ones. If the system classifies them all as negative, the accuracy, as well as the precision for the negative class, would be 99.5%, even though the classifier missed all positive cases. Other performance measures account for this by including TP in a product. One of the most widely used is the F -measure:

$$F = \frac{2 \times R \times P}{R + P}$$

that is actually the harmonic mean of precision and recall. This measure will be used to assess classifier performance whenever the problem has one class that is underrepresented with respect to the other one.

2.8.2 Evaluation methods

In order to estimate the generalization error of a classifier, datasets are typically split in training and test (or validation) sets. The classifier is built using trained data, and its performance is evaluated on test data. A key question is to avoid *overfitting* training data, as this normally is a sign of poor performance on unseen data.

Cross-validation This is one common method that is used to evaluate how well a classifier is expected to perform. This involves first randomly dividing the ground-truth instances into k equally sized parts. The classification algorithm is then used to train k models that are each validated once. The training and validation of each of the k models is called a *fold*. Each fold involves using one of the k data partitions for validation and the other $k - 1$ partitions for training. Each of the k partitions is thus used for validation exactly once, and none of the k trained models is ever validated on instances that were used to train it. The average error rate and standard deviation of the error rate is then computed across all folds. Usually a choice of k between 3 and 10 is made, depending on the size of the dataset and the number of classes in it. As a rule of thumb, one would like to have a sufficient number of training samples from all classes, in order to build an accurate model, as well as enough test samples that will allow robust error estimation. Also, It is usually best to make sure that each class is represented in comparable proportions in each data subset. *Stratified crossvalidation* is a form of crossvalidation that keep class proportions across folds. Whenever possible, a *balanced dataset*, where all classes are represented more or less equally, is desirable for evaluating a classifier. However, this depends on the nature of the problem to solve. For example, in melody part selection, a problem addressed in Chapter 3, when building ground-truth sets from real data, melody samples are underrepresented, compared to non-melody ones.

Leave-one-out When only a small ground-truth dataset is available for training, a *leave-one-out* error estimation scheme could be appropriate. This is merely a crossvalidation where $k = n$, where n is the number of instances in the dataset. The classifier is trained n times on $n - 1$ instances, and tested in the remaining instance. This technique generally gives good estimates, as the trained models been evaluated are almost identical to the classifier been tested, when trained with the whole dataset). However, it can be a very time consuming task, depending on the time complexity of the classification scheme been used.

CHAPTER 2. TECHNICAL BACKGROUND

File-level crossvalidation In some of the experiments in music genre recognition performed here, a number of samples can be extracted from a single MIDI file. This means that, using a simple crossvalidation scheme, samples from the same MIDI file (indeed, from the same MIDI track) could be found in both the train and test partitions. In order to avoid that, in those experiments a MIDI file-level crossvalidation process is used, where samples from the same MIDI file are always assigned to the same partition.

Classifier ensemble crossvalidation Evaluating classifier ensembles require three different datasets: one for training the base classifiers, one for testing them and estimating their performance (and therefore their weight) in the ensemble, and a third one to estimate the ensemble performance. When only a dataset is available for estimating performance, a crossvalidation strategy can be used. In general, the dataset is divided in k parts, as usually. At least one of them is used for testing base classifiers, and another one for validating the ensemble. The rest of partitions is usually used for training base models. This experiment set up can be rotated and results averaged, choosing each time different partitions for testing/validation, so each sample is at least used for validation once.

Another variant of this evaluation technique for classifier ensembles is to perform a nested crossvalidation within the ensemble. An *outer* crossvalidation is performed, reserving one partition for validating the ensemble, the rest of them to ‘train’ it. Training the kind of ensembles used in this work (based on weighted and unweighted voting rules), means to train base models, and to estimate their weights in the ensemble. For this, an *inner* crossvalidation is performed. The ensemble training data is divided in a number of partitions, and base classifiers are crossvalidated with them. Results from this ‘internal’ evaluation are used to set up base model weights. Then the base models are trained using the whole ensemble training data, in order to perform an *outer* crossvalidation iteration.

Significance tests

Statistical hypothesis testing and significance testing tools based on interval estimation can help to conclude whether a given classifier is better than another based on their relative cross-validation performances. Such statistical hypothesis testing makes it possible to judge the likeliness that a given average classification accuracy across cross-validation folds, for example, will be within a certain range of the true performance of the classifier on new data.

2.8. PERFORMANCE EVALUATION

There are a number of different hypothesis testing techniques that can be used. *Student's paired t test* can be used to examine the results of k folds in order to, with a chosen confidence level, accept or reject the hypothesis that the true error rate of a classifier is at or below some value.

A contingency table may be constructed in order to indicate how well two classifiers perform on the same validation set after being trained on the same training set. Such a table specifies the number of instances correctly classified by both classifiers, the number incorrectly classified by both, the number correctly classified by one but incorrectly by the other and vice versa. *McNemar's test* makes it possible to use a contingency table to test, with a given confidence level, the hypothesis that the two classifiers have statistically the same error rate.

In practice, one often wishes to test more than two classifiers in order to find which is truly the most accurate. For example, consider the case where l candidate algorithms are each trained on k datasets, such that there are k trained classifiers for each of the l algorithms, and one wishes to test the l algorithms for statistically significant differences in performance. *Analysis of variance* (ANOVA) provides a means for doing this, by testing the hypothesis that the mean error rates of the l groups are equal. If this hypothesis is rejected, then a multiple comparisons test should be applied. The *Tukey-Kramer Honestly Significant Difference* (TK-HSD) is one of the methods available. It is a single-step multiple comparison procedure and statistical test generally used in conjunction with an ANOVA to find which means are significantly different from one another. The TK-HSD test is applied simultaneously to the set of all pairwise comparisons. It identifies where the difference between two means is greater than the standard error would be expected to allow. It assumes equal variance in the compared distributions. It has been used in several MIREX evaluation contest editions, starting from 2008.

2.8.3 Ranking aggregation

In pattern recognition applications, there are situations where making a ranking is desirable. For example, in feature selection, a ranking of relevant features could be helpful. Also, in classifier comparison, a ranking of classifiers based on their estimated accuracy can help decide which do perform best. Sometimes, a number of rankings are made by different rating methods, and a *ranking aggregation* technique should be used. The *Borda's count* method is a simple yet effective ranking aggregation technique originally devised for dealing with elections. This method assigns a weight

CHAPTER 2. TECHNICAL BACKGROUND

to each candidate. Several alternatives exist for computing such weight. For the k -th candidate, its weight $w(k)$ can be defined as:

$$w(k) = \sum_{i=1}^R n - r_i(k), \quad \text{or} \quad (2.51)$$

$$w(k) = \sum_{i=1}^R 1/r_i(k) \quad (\text{Nauru variant}) \quad (2.52)$$

where n is the number of candidates, $r_i(k)$ is the position in the i -th ranking for candidate k , and R is the number of rankings to aggregate.

Once the weights are computed, a new ranking where candidates are ordered by their weight in descending order, or simply the candidate with greater weight is declared as winner.

2.8.4 Evaluation materials

Both symbolic and audio music corpora used in this thesis are described here. Some of them were built by the author's research group, where indicated. Tables 2.2 and 2.3 show a summary of both kinds of datasets. In the case of symbolic corpora, the number of non-empty tracks is reported.

Corpus	files	tracks	genres	tags	remarks
JvC1	110	439	2	genre	(*)
JvC2	42	139	2	genre	(*)
9GDB	856	4143	3/9	genre, melody	2-level hierarchy (*)
CL200	200	688	Classical	melody	(*)
JZ200	200	795	Jazz	melody	(*)
KR200	200	1657	Popular	melody	(*)
CLA	511	2258	Classical	melody	(*)
JAZ	856	4370	Jazz	melody	(*)
KAR	1360	13119	Popular	melody	(*)
RWC-G	48	311	18	genre, melody	
RWC-P	75	801	Popular	melody	Western/Japanese pop

(*) developed by the author's research group.

Table 2.2: Symbolic (MIDI) corpora.

Symbolic music corpora

JvC1 and JvC2 These are two-class corpora made up of MIDI files from Classical and Jazz genres, used for music genre classification evaluation. In

2.8. PERFORMANCE EVALUATION

Corpus	files	length	format	classes	references
GTZAN	1000	30 sec.	mp3	10 genres	(Tzanetakis, 2002)
ISMIRgenre	1458	full		6 genres	(Cano et al., 2006)
ISMIRrhythm	698	30 sec.	WAV	8 dances	(Cano et al., 2006)
LMD	3225	full	mp3	10 genres	(Silla Jr. et al., 2008)
Africa	1024	full		var.	(Cornelis et al., 2005)
MIREX2007genre	7000	30 sec.	WAV	10 genres	(MIREX, 2007a)
MIREX2007artist	3150	30 sec.	WAV	105 artists	(MIREX, 2007a)
MIREX2007classical	2772	30 sec.	WAV	11 composers	(MIREX, 2007a)
MIREX2007mood	600	30 sec.	WAV	5 clusters	(MIREX, 2007a)

Table 2.3: Audio music corpora.

some setups, JvC1 is used as a training dataset, and JvC2 as a validation test. In other setups, they are merged to form the JvC1+2 corpus. Table 2.4 presents a summary of these corpora. A characteristic of the corpora is that each file has just one monophonic melody track in it. Classical music was chosen from works by Mozart, Bach, Schubert, Chopin, Grieg, Vivaldi, Schumann, Brahms, Beethoven, Dvorak, Haendel, Pagannini, Mendhelson, Wagner, and Listz. Jazz music consist of jazz standard tunes from a variety of authors like Charlie Parker, Duke Ellington, Bill Evans, Miles Davis, etc. This dataset is available for research purposes upon request to the author.

		JvC1	JvC2	JvC1+2
Jazz	files	65	21	86
	tracks	237	147	384
	melodies	65	21	86
Classical	files	45	21	66
	tracks	202	162	364
	melodies	45	21	66

Table 2.4: JvC1 and JvC2 MIDI file corpora summary.

9GDB/AJP 9GDB is a MIDI file collection developed within my research group at the University of Alicante. It was fully described in (Pérez-Sancho, 2009b) (as *Perez-9-genres*). It consists of 856 full length songs in MIDI file format, organized as a two-level genre taxonomy, as shown in Figure 2.18. For experiments on information fusion from the audio and symbolic domains (sec. 4.5), the corpus was synthesized to WAV for

CHAPTER 2. TECHNICAL BACKGROUND

extracting audio descriptors, and then re-transcribed to monotimbral MIDI sequences to obtain symbolic features from the resulting transcription.

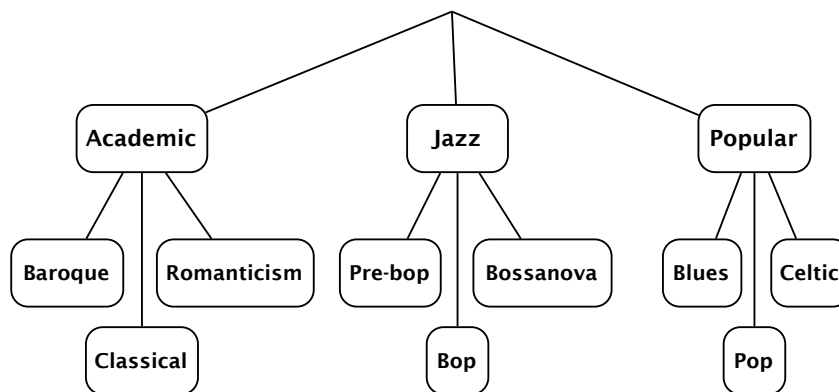


Figure 2.18: 9GDB dataset taxonomy.

A variant of this dataset, called AJP, where individual tracks are tagged as melody or accompaniment, is used for melody track selection research. It consists of the same three top level genres, and the same leaf genres for academic and jazz music, but the popular top level genre has only two subgenres: country and pop-rock. It is made up of 762 full length songs.

RWC-G and RWC-P These two datasets are part of the RWC (Real World Computing) Music Database (Goto, 2004). This was originally an audio music database, but latter the authors provided MIDI versions of the recorded pieces. The RWC database is divided in several corpora. The *Music Genre Database* (Goto et al., 2003), which is one of these corpora, consists of 100 musical pieces in all with three pieces prepared for each of 33 genres and one for a cappella. The database is divided into 10 main categories of genres (popular, rock, dance, jazz, latin, classical, marches, world, vocals, and traditional Japanese music) and 33 subcategories. All 100 pieces are original recordings with 73 being original compositions and 27 being existing public-domain pieces. The RWC-G corpus is a subset of this database, consisting only of MIDI files containing at least a melody track and an accompaniment track. This corpus has both a genre tag per file and a melody tag per track.

The RWC-P corpus is a subset of the *Popular Music Database* (Goto et al., 2002), which consists of 100 songs: 20 songs with English lyrics performed in the style of popular music typical of songs on the American hit charts in the 1980s, and 80 songs with Japanese lyrics performed in the style of modern Japanese popular music typical of songs on the Japanese

2.8. PERFORMANCE EVALUATION

hit charts in the 1990s. All 100 songs with vocals were originally produced (composed, arranged, performed, and recorded). Again, only MIDI files with at least one melody track and one accompaniment track were included in RWC-P. This corpus has each track tagged as melody or non-melody.

Melody tagged corpora Six corpora (named JZ200, CL200, KR200, JAZ, CLA, and KAR, in Table 2.2) were constructed for evaluating melody identification approaches discussed in Chapter 3. A detailed discussion about them can be found in section 3.1.6.

Audio music corpora

ISMIRgenre and ISMIRrhythm The ISMIRgenre and ISMIRrhythm collections were compiled for the genre and rhythm classification tasks, respectively, of the ISMIR 2004 Audio Description contest (Cano et al., 2006) and used frequently thereafter by Music IR researchers. ISMIRgenre consists of the train and development datasets of the genre classification task. They are made up of 6 popular music genres: Classical, Electronic, Jazz/Blues, Metal/Punk, Rock/Pop, and World. All the songs were courtesy from the Magnatune³ website where they are available under a Creative Commons⁴ license for ‘non-commercial use’.

The ISMIRrhythm dataset comprises 8 Latin and Ballroom dances. It is made up of the train and test datasets of the Rhythm Classification contest, whose goal was to compare algorithms for automatic classification of rhythm classes (Samba, Slow Waltz, Viennese Waltz, Tango, Cha Cha, Rumba, Jive, Quickstep) from audio data. Music files are 30-s instances of raw audio at 44100 Hz, 16 bits, mono.

MIREX2007genre This is the dataset used in the MIREX 2007 edition of the audio genre classification and audio similarity contests. Collection statistics: 7000 30-second audio clips in 22,050 Hz mono wav format drawn from 10 genres (700 clips from each genre). Genres are organized in a two level taxonomy, shown in figure 2.19.

MIREX 2007: Audio artist identification dataset Collection statistics: 3150 30-second 22,050 Hz mono wav audio clips drawn from 105 artists (30 clips per artist drawn from 3 albums).

³<http://magnatune.com>

⁴<http://creativecommons.org>

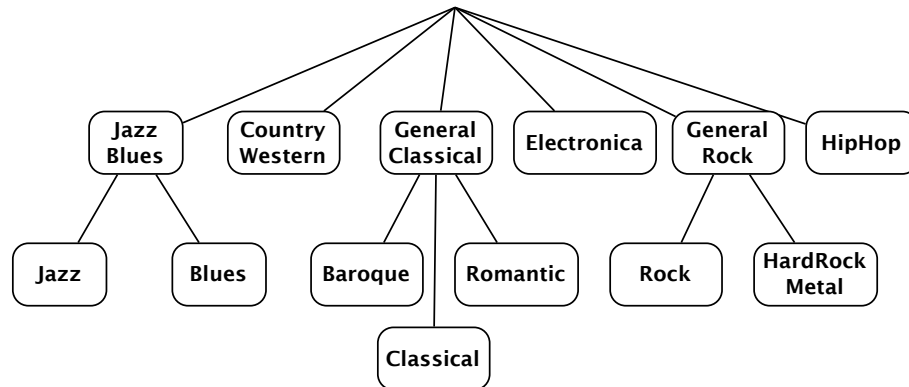


Figure 2.19: MIREX2007 audio genre dataset taxonomy.

MIREX 2007: Audio classical composer identification dataset 2772 30-second 22,050 Hz mono wav clips organised into 11 “classical” composers (252 clips per composer). The database contains tracks for Bach, Beethoven, Brahms, Chopin, Dvorak, Handel, Haydn, Mendelssohn, Mozart, Schubert, and Vivaldi.

30 second clips, 22,050 Hz, mono, 16bit, WAV files.

MIREX 2007: Audio mood classification dataset The ground-truth data set consisted of 600 30-second 22,050 Hz, mono, 16 bit, WAV files evenly divided into 5 mood clusters, according to metadata provided by APM⁵. It covers a variety of genres: each category covers about 7 major genres and a few minor genres. The assignment of moods to files was done by human evaluators with a file accepted as a representative of a given mood when at least 2 out of 3 graders concurred in the assignment. To make the problem more interesting, the distribution among major genres within each category is made as even as possible. Table 2.5 shows the mood tags in each cluster.

Latin Music Database The *Latin Music Database* (Silla Jr. et al., 2008) was used for the first time in the MIREX 2008 audio genre classification task. It has 3,227 audio files from 10 latin music genres: Axá, Bachata, Bolero, Forró, Gaúcha, Merengue, Pagode, Sertaneja, and Tango. Contrary to popular Western music genres, each of these genres has a very specific cultural background and is associated with a different region and/or ethnic and/or social group. Nevertheless, it is important to note that in some aspects the Latin Music Database is musically similar to Western music databases as it

⁵<http://www.apmmusic.com>

2.8. PERFORMANCE EVALUATION

Cluster	mood tags
A	passionate, rousing, confident, boisterous, rowdy
B	rollicking, cheerful, fun, sweet, amiable/good natured
C	literate, poignant, wistful, bittersweet, autumnal, brooding
D	humorous, silly, campy, quirky, whimsical, witty, wry
E	aggressive, fiery, tense/anxious, intense, volatile, visceral

Table 2.5: MIREX 2007 Audio mood classification clusters.

makes use of modern recording and post-processing techniques. By contrast to the MIREX 2007 audio genre collection, it contains at least 300 songs per music genre, which allows for balanced experiments.

GTZAN The GTZAN Genre Collection was used for the well known paper in genre classification ([Tzanetakis and Cook, 2002a](#)). After the author,

Unfortunately the database was collected gradually and very early on in my research so I have no titles (and obviously no copyright permission, etc.). The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions. Nevertheless I have been providing it to researchers upon request mainly for comparison purposes, etc.

The dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22,050 Hz Mono 16 bit audio files in WAV format.

Africa The African collection is a subset of 1024 instances of the audio archive of the Royal Museum of Central-African Belgium, digitized in the course of the DEKKMMA project ([Cornelis et al., 2005](#)). Various metadata categories are available for this set, including 27 different functions, 11 different instrument families, 11 different countries and 40 ethnic groups ([Lidy et al., 2010b](#)). The number of files varies according to number of meta-data available in each category.

3

Melody part selection

*“We all do ‘do, re, mi,’ but you have got
to find the other notes yourself.”*

Louis Armstrong

The goal in this chapter is to pose the general question *What is a melody?* under a statistical approach.

3.1 Melody characterization in multi-part music files

As stated in chapter 1, melody characterization, as approached in this research, refers to the task of deciding which of the tracks in a multi-part music file contains the main melody. For this, we need to assume that the melody is indeed contained in a single voice or track. This assumption is also taken by others authors (Friberg and Ahlbäck, 2008; Madsen and Widmer, 2007b; Tang et al., 2000), as there is empirical evidence that it is the case for much of today’s symbolically encoded western music. An equally important side goal is to obtain objective but human understandable characterization of melodies. The objectivity of such a characterization comes from the fact that these melody descriptions are automatically extracted from sample melodies. This is achieved applying a classical machine learning and pattern recognition methodology to the input data, using minimum *a priori* knowledge. The only such knowledge that is introduced in the system is the discourse universe, that is, the input domain definition, used in building melody representations.

There are some features in a melody part that, at first sight, seem to be valid enough for identifying it, like the presence of relatively higher pitches, as suggested by some results in (Uitdenbogerd and Zobel, 1999), or being monophonic (Uitdenbogerd and Zobel, 1998). Unfortunately, empirical analysis will show that these hypotheses do not hold in general, as discussed

CHAPTER 3. MELODY PART SELECTION

in Section 3.1.3, and more sophisticated criteria need to be devised in order to take accurate decisions.

Another strategy, based on metadata information found in multi-part music formats like MIDI files could also be used. For example, a dictionary based-classifier could be built gathering track names from MIDI file collections, associating some of them with the class *melody* and the rest with the class *non-melody*. Then the classifier would output the class of the dictionary track name closer to a given track’s name, using some string distance metric. Unfortunately, there is no standard naming procedure for MIDI tracks. Also, the language used by MIDI file authors for naming tracks is another issue. It is unlikely that the training corpus would contain examples from all possible languages. Therefore, this method would only perform reliably in a controlled environment. Another drawback of the text metadata approach for melody track discovery is that such a method would obviously tell us nothing about the content of melody tracks. Hence, the metadata approach was not considered here, being one of the goals in this research to provide some objective empirical insights on the nature of the *melody* concept. However, it was used to do some automatic pre-tagging in the first stages of building the corpora used for evaluation purposes. See section 3.1.6 for a discussion on some findings.

3.1.1 Melody characterization system overview

To overcome the aforementioned problems, machine learning and pattern recognition tools like decision trees, rule inference algorithms, and fuzzy logic are used in this research to obtain melody characterizations. Low-level statistics are used for voice/track content description. The models obtained by training such classifiers on the proposed track descriptions are used to assign a probability of being a melody to each candidate track ¹. Furthermore, the models are processed in order to obtain compact human-friendly melody characterizations.

In a first approach, the random forest classifier (RF, see section 2.2.5) –an ensemble of decision trees– was chosen to model melody parts, due to its good overall classification performance, a built-in feature selection method, and the fact that it produces a human-readable model. In order to obtain a compact characterization for melodies, the decision trees obtained in the training process were converted to rules. These rules were simplified and ranked in order to select a small number of compact rules much more manageable

¹A modified version of the WEKA (Hall et al., 2009) toolkit was used for the feature extraction and classification phases.

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

for human understanding. In section 3.1.6 classification results using the RF classifier are discussed. Section 3.2 presents a method for extracting compact melody characterization rule sets from RF models.

As an alternative approach, the RIPPER rule inference algorithm (section 2.2.5) is used as a tool for melody track classification and characterization. The training process focus on underrepresented classes to infer rules. It is well-suited to this problem, as melody tracks are far less abundant in multi-part music files than other track types. Melody representations produced by RIPPER are quite compact and can be compared to those obtained by the first approach. Once a rule set for characterization of melodies is obtained, they are converted to a natural language-like definition, namely a fuzzy rule system. This is achieved by a process of *fuzzification* of *crisp* rule systems. A *genetic fuzzy system* (Cordón and Herrera, 1995) is used to automate the conversion. The resulting fuzzy description of melody tracks is, to our knowledge, the first of its kind in the MIR field. Section 3.3 discuss melody track classification results using RIPPER, and presents a method for fuzzifying such rule sets.

Moreover, under the same approach, any other kind of track could be learnt. For example, we can infer a set of rules to identify the bass track, or piano tracks, etc. Work in this line is being undertaken within our research group. The characterization of bass tracks in MIDI files is being investigated using the same methodology described here for melody tracks.

The classification method proposed here can be used as a tool for extracting the melody track in conjunction with a system for music genre recognition presented in the next chapter. This system is a melody-based genre recognition system. This way, multi-part music files need not to be preprocessed by an expert in order to identify the melody track.

3.1.2 MIDI track description

The multi-part music file format chosen for evaluating the proposed method is the Standard MIDI file format. It is a widely available symbolic music format, compatible with most platforms and music software, making it easier to build MIDI file corpora from several sources (private or public collections, web sites, etc.) for evaluation purposes. However, with no loss of generality, the proposed method could be applied to any other symbolic music multi-part encoding format.

CHAPTER 3. MELODY PART SELECTION

Empty tracks and tracks playing on the percussion channel² are filtered out in this approach. A track is considered an *empty track* if it contains less than ϵ notes. This value has been set to $\epsilon = 3$, the minimum number of values necessary to produce non-trivial statistics. Also, it is unlikely that a melody is played on the MIDI percussion channel, as most modern MIDI equipment is configured by default to use this channel for playing so-called percussion or drum banks, mapping different pitch values to different instruments.

Each remaining track in a MIDI file is described by a vector of statistics that summarize track content information. A set of such descriptors has been defined based on several categories of features that assess melodic and rhythmic properties of a music sequence, as well as track related properties. This set is presented in Table 3.1. The first column indicates the category being analyzed, and the second one shows the kind of statistics describing properties from that category. The third column indicates the range of the descriptor. This kind of statistical content description is sometimes referred to as *shallow structural description* (Pickens, 2001; Ponce de León et al., 2004).

Some features were designed to describe the track as a whole while others describe particular aspects of its content. Let us denote T as the set of non-empty tracks t_i in a MIDI file. Four track information descriptors were proposed for track t_i , namely its normalized duration, D_i , in MIDI ticks, the number of notes in the track, N_i , its occupation rate, OR_i , i.e., the proportion of the track length occupied by notes, and its polyphony rate, PR_i , defined as the ratio between the number of ticks in the track where two or more notes are active simultaneously and the number of ticks occupied by notes:

$$D_i = \frac{|t_i|}{\max_j |t_j|} \quad (3.1)$$

$$OR_i = O_i / D_i \quad (3.2)$$

$$PR_i = P_i / O_i \quad (3.3)$$

where O_i is the amount of ticks in t_i where at least one note is active, and P_i is the amount of ticks in t_i where two or more notes are active simultaneously. $|t_i|$ is the length of the track in ticks. Figure 3.1 shows an example of occupation rate and polyphony rate computation for an 1-bar sample.

Track content-related descriptors, both normalized and non-normalized versions, were devised. Normalized descriptors are computed using the expression $(v - \min) / (\max - \min)$, where v is the descriptor to be normalized

²MIDI channel number 10 is the channel for percussion/drums instrument kits, as specified in the General MIDI standard.

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

Table 3.1: MIDI track descriptors

Category	Descriptors	Domain ³
Track info.	Normalized duration	$[0, 1]$
	Number of notes	$[0 .. +\infty[$
	Occupation rate	$[0, 1]$
	Polyphony rate	$[0, 1]$
Note pitch	Highest	$[0 .. 127]$
	Lowest	$[0 .. 127]$
	Mean	$[0, 127]$
	Standard deviation	$[0, +\infty[$
Pitch intervals	Number of distinct intv.	$[0 .. 127]$
	Largest	$[0 .. 127]$
	Smallest	$[0 .. 127]$
	Mean	$[0, 127]$
	Mode	$[0 .. 127]$
	Standard deviation	$[0, +\infty[$
Note durations	Longest	$[0, +\infty[$
	Shortest	$[0, +\infty[$
	Mean	$[0, +\infty[$
	Standard deviation	$[0, +\infty[$
Syncopation	No. of syncopated notes	$[0 .. +\infty[$
Class	IsMelody	$\{\text{true}, \text{false}\}$

corresponding to a particular track, and *min* and *max* are, respectively, the minimum and maximum values for this descriptor for all tracks in the target MIDI file. This allows to know these properties in relation to other tracks in the same file. This way, a total number of $4 + 15 \times 2 = 34$ descriptors are initially computed for each track. The class label *IsMelody* is a boolean tag where a true value indicates that the sample corresponds to a melody track. This label will obviously be missing for new input files. The labeling process of tracks is explained in section 3.1.6.

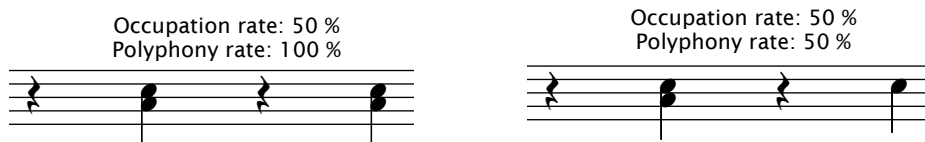


Figure 3.1: Occupation and polyphony rates examples.

CHAPTER 3. MELODY PART SELECTION

Note pitch is measured using MIDI pitch values. The maximum possible MIDI pitch is 127 (note G_9), and the minimum is 0 (note C_{-1}), being 60 the value of middle C (C_4). The interval descriptors summarize information about the difference in pitch between consecutive notes⁴. Absolute pitch interval values are computed. This means that regardless the interval is an ascending or descending one, its value is always positive. Note duration values are computed as *beats*, so they are independent from the MIDI file resolution. Finally, the number of *syncopations* found in a track is computed. A note which starts in a given range around the middle of a beat and continues sounding over the next is considered a syncopation. This range is set to a sixteenth note around the middle of the beat.

3.1.3 Feature selection

The descriptors listed above are an initial set of computed features, but any pattern recognition system needs to explore which are those features that are actually able to discriminate between target classes.

Some descriptors show evidence of statistically significant differences when comparing their distributions for melody and non-melody tracks, while other descriptors do not. This property is implicitly observed by the classification techniques utilized (see Section 2.2.5 and 2.2.5), that perform a selection of features in order to build decision trees or induce rules.

A look at the plots in Figure 3.2 provides some hints on how a melody track could look like. This way, melody tracks seem to have less notes than other non-melody tracks, an average mean pitch, they do not contain the longest notes in the song and tend to be near-monophonic. When this sort of hints are combined by the classifier, a decision about the track “melodicity” is taken.

Some attempts were made to use single descriptors to characterize melodies. However, results showed that no descriptor among the ones presented here was able by itself to characterize melody against other type of tracks. For example, using the polyphony rate descriptor with the Bayes rule yielded a classifier that always predicted the majority class (*IsMelody=false*). Similar results were obtained using a nearest-neighbour rule.

Other authors (Madsen and Widmer, 2007b) report more successful results on the same task when using entropy-based single descriptors on pop and rock hits. However, best results for whole tracks were still obtained using an interval and inter-onset-interval (IOI) joint entropy. This suggests

⁴*consecutive notes* are notes whose MIDI onset events are consecutive in the MIDI track content stream.

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

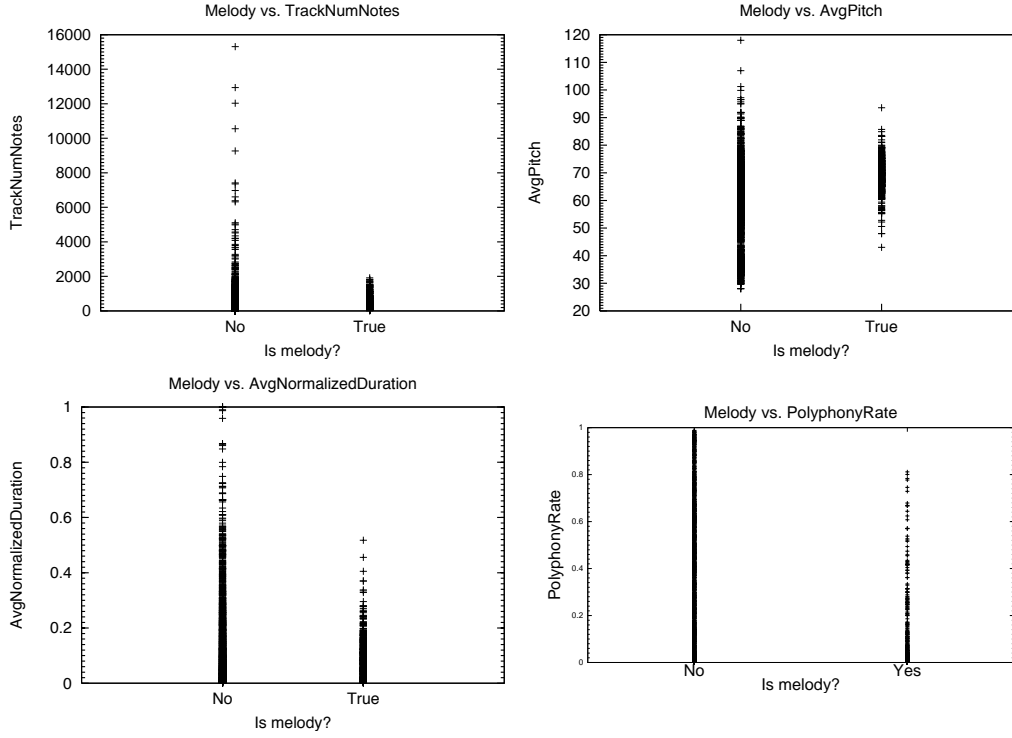


Figure 3.2: Distribution of values for some descriptors (left-to-right, top-to-bottom: number of notes, average pitch, mean normalized note duration, and track polyphony rate).

that the combination of information from different musical dimensions (pitch and duration, in this case) can improve on results obtained using single dimension descriptors. Interestingly, when classifying short track fragments using entropy measures (so called *entropy-based local prediction* by the authors), best results were obtained by single dimensional, IOI-based entropy descriptors.

3.1.4 Classification methods

One of the objectives of this research is to produce an human-friendly characterization of melody tracks. Decision trees and rule inference algorithms are two of the machine learning techniques that can build such models learning from data. As explained in section 3.1.1, the random forest classifier and the RIPPER rule inference algorithm were selected as classification methods for the experiments presented in this chapter.

In what follows, let's denote class $IsMelody=true$ as M , and class $IsMelody=false$ as \bar{M} . The membership probability for M is denoted as

$p(M|t_i)$, and interpreted as the probability that track t_i contains a melodic line. On the other hand, $p(\bar{M}|t_i)$ denotes the probability that track t_i is not a melody. Obviously, it holds that $p(M|t_i) + p(\bar{M}|t_i) = 1$.

3.1.5 Melody track categorization and melody part selection

Considering tracks separately, so each track is a input sample, *melody track categorization* is the problem of deciding whether a given track contains a melody or not. In experiments performed in this research, a track t_i is tagged as a melody if $p(M|t_i) > 0.5$.

On the other hand, a general problem that arise when dealing with digital scores, in particular MIDI files, is to decide which track sections play the melody in any given instant. This is often called *melody line extraction* from polyphonic data, and has been addressed by several authors (Ozcan et al., 2005; Shan and Kuo, 2003; Uitdenbogerd and Zobel, 1998). Given the ability of a system to categorize entire tracks as melodies, a simpler formulation of the problem is, given a MIDI file, to identify which of its tracks most probably contains the melody. Let us call this a *melody part selection* problem. The question arise whether the answer should be a single track or a list of tracks. From what can be learned by examining the corpora at hand, it has been found that most songs in MIDI format contain just one tagged melody track (see Table 3.3). Thus, the system presented here outputs only the most probable track per song.

The underlying model training phase uses the track as the sampling unit, like in the melody track categorization problem. As a file can have zero, one or more melody tracks, a model prediction is considered correct if:

1. At least one track in the file is tagged as *melody* and the predicted track is one of them.
2. There are no melody tracks and the classifier predicts no melody track.

Subsequently, models may produce three different classification error scenarios:

- **Type 1 error:** Wrong melody track. Actually, there is a melody track, but it is not the one predicted.
- **Type 2 error:** No melody expected, but a melody track was predicted.
- **Type 3 error:** A melody track exists but no one was predicted.

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

Note that, in the corpora used in the experiments, songs with zero or more than one melody track are much less frequent than single melody songs. Thus, type 1 errors are expected to occur more often.

A classification method like RF, that is able to assign a membership probability $p(M|t_i)$ to an input instance t_i , is well suited for this task. This property also decreases the possibility of ties that would arise from classifiers that issue a single class answer.

Given an input file with a set of tracks T , the predicted melody track index is

$$\hat{i} = \begin{cases} \arg \max_i p(M|t_i) & p(M|t_i) \geq p_\epsilon, i = 1..|T| \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

where p_ϵ is a probability threshold for a track to be considered as a melody track candidate. A prediction $\hat{i} = 0$ means that the system predicts no track in the given file containing a melody.

3.1.6 Experiments and results

Datasets and tools

Six corpora (see Table 3.2) were constructed for evaluating the melody identification approaches presented above. The files were downloaded from a number of freely accessible Internet sites. First, three corpora (named JZ200, CL200, and KR200) were created to set up the system and tune the parameter values. JZ200 contains jazz music files, CL200 has classical music pieces, and KR200 contains popular music songs with a part to be sung⁵. All of them are made up of 200 files. Three other corpora (named JAZ, CLA, and KAR) from the same music genres were compiled from a number of different sources to validate our method.

The main difficulty for building the data sets was to label the tracks in the MIDI files. Text tagging of MIDI tracks based on metadata such as the track name, is unreliable, due to reasons explained in section 3.1.

Thus, a manual labeling approach was carried out. Several musicians listened to each one of the MIDI files playing all tracks simultaneously. For each file, tracks containing the perceived melody were identified and tagged as *melody*. The rest of tracks in the same file were tagged as *non-melody*. In particular, introduction passages, second voices or instrumental solo parts were tagged as *non-melody*. During this process, the *melody* concept revealed itself as ambiguous, and definitively loosely related to the *track* concept.

⁵these are actually files in karaoke (.kar) format

CHAPTER 3. MELODY PART SELECTION

Corpus ID	Genre	Files	Tracks	Tracks per file	Melody tracks	Non-melody ratio
CL200	Classical	200	688	3.4	205	2.4
JZ200	Jazz	200	759	3.8	200	2.8
KR200	Popular	200	1657	8.3	204	7.1
CLA	Classical	511	2258	4.4	607	2.9
JAZ	Jazz	856	4370	5.1	877	4.0
KAR	Popular	1360	13119	9.6	1318	8.6

Table 3.2: Corpora used in the evaluation, with identifier, music genre, number of files, number of non-empty tracks, number of melody tracks, and ratio of non-melody tracks per melody track.

Roving and *compound* melodies were specially hard to identify and tag. So, the key question the tagging musicians posed themselves was

Do the track contain (possibly a significant part of) the melody?

This means any track containing significant portions of the perceived melody, possibly containing accompaniment parts, either in sequence or simultaneous to the melody, were tagged as melody tracks. So, the relation between the concepts *melody* and *track* is understood as an inclusion, rather than an equivalence relation. The extent to which a melody part in a track is considered significant in a given piece was left to the musician criterion. As a matter of fact, a track containing roughly at least 50% of the perceived melody was always considered a melody track. Table 3.3 shows a summary of tagged tracks.

Corpus ID	no melody	one melody	more than one
CL200	0	199	1
JZ200	0	200	0
KR200	9	182	9
CLA	70	353	88
JAZ	4	845	7
KAR	90	1234	36

Table 3.3: Summary of number of tagged melody tracks per song.

Some songs turned out to have no tracks tagged as melody because either it was absent, or the song contained some kind of melody-less accompaniment

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

with a *lyrics* track. These kind of tracks can be found in karaoke MIDI files. Most of them contain text meta events and no note events. However some of these tracks contain also melody note events, but with the velocity (volume) value set to a minimum (inaudible) level. In this latter case, *lyrics* tracks were tagged as melody.

Other songs contained more than one melody track (e.g. duplicates, often with a different timbre) and all those tracks were tagged as *melody* as well.

The WEKA package was used to carry out the experiments described here. It was extended to compute the proposed track descriptors directly from MIDI files.

Four experiments have been carried out, as listed below:

- Melody vs. non-melody classification (Melody track categorization)
- Melody track selection
- Genre specificity on melody part selection
- Training set specificity on melody part selection (generalization capability)

The first one tries to assess the ability of the classifier to classify melodic and non-melody tracks properly. In the second experiment, the aim is to evaluate how accurate the system is for identifying the melody track in a MIDI file. Finally, the specificity of the system with respect to both the music genre and the corpora utilized is tested.

Melody vs. non-melody (Melody track categorization)

Given a set of tracks, this experiment classifies them either as melody or non-melody. As a proof of concept, three independent sub-experiments were carried out, using the 200-file corpora (CL200, JZ200, and KR200), and the RF classifier with default Weka parameters. A 10-fold cross-validation scheme was used to estimate the accuracy of the method. The results are shown in Table 3.4 and Figure 3.3, along with the baseline ratio when considering a dumb classifier that always outputs the most frequent class (*non-melody* for all datasets). The remarkable success percentages show that track content statistics in combination with decision tree based learning can produce good results on the task at hand. Also, precision (P), recall (R) and F-measure (F) are shown for melody tracks.

CHAPTER 3. MELODY PART SELECTION

Corpus	Success	Std. dev.	Baseline	P	R	F
CL200	99.1%	0.7	70.2%	0.981	0.99	0.985
JZ200	98.3%	1.4	73.6%	0.984	0.95	0.967
KR200	95.9%	1.8	87.7%	0.926	0.728	0.815

Table 3.4: Melody versus non-melody classification results.

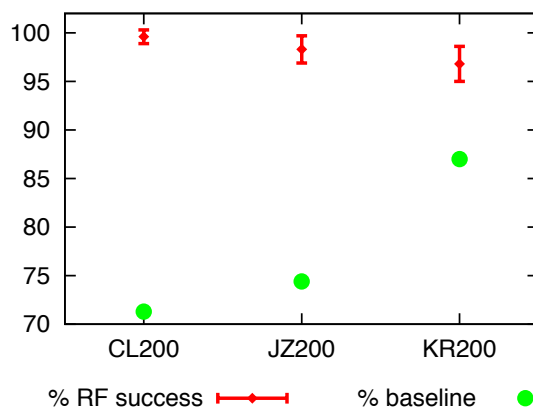


Figure 3.3: Melody vs. non-melody classification success and baseline

These results have been obtained using $K = 10$ trees and $F = 5$ randomly selected features for the RF classifier. The same parameters have been used in the rest of experiments using RF presented in the next sections. The results are very good, in general. A summary of classification errors is shown in table 3.5. Tables 3.6, 3.7, and 3.8 show examples of false positives and false negatives for each corpus. Almost in every error case there is evidence that could explain the misclassification (see the remarks column on the tables).

Corpus	size	FP	FN (due to draws)
CL200	688	4	4 (1)
JZ200	759	3	10 (4)
KR200	1657	12	56 (12)

Table 3.5: Melody track categorization error summary.

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES



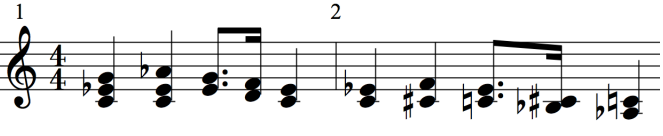



<i>Song</i> : Track	$p(M t_i)$	Remarks
False Negatives		
<i>Canon in D major</i> : Tenor/Violin II 	.2	Song: melody change track at 4-bar boundaries. Track: 32/114 bars of actual melody. Remaining bars are accompaniment.
<i>Prelude #1 in CM</i> (Bach) : Melody 	.5	Equal probability. Melody 16ths arpeggio like, steady rhythm.
<i>Prelude in C minor</i> (Chopin) : Melody 	.4	2-track song. piano right-hand-like track. Polyphonic rate higher than accompaniment track.
False Positives		
<i>Eine Kleine Nachtmusik</i> (Mozart) : Strings 	.9	String quartet like sequence. Track plays melody seldom. Low polyphonic rate. Cloned track (so two errors reported).
<i>Nabucco. Va, Pensiero</i> (Verdi) : Right Hand 	.6	Mimics melody for a significant part of the song, first in mono, then adding voicings, then switching to accompaniment.
<i>Prelude in C minor</i> (unknown) : Strings 	.7	String ensemble track with monophonic parts backing melody.

Table 3.6: CL200 classification error examples

CHAPTER 3. MELODY PART SELECTION


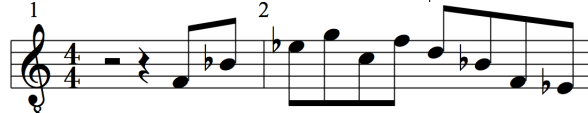

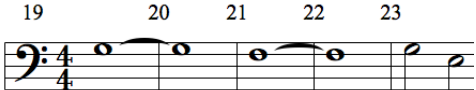
<i>Song</i> : Track	$p(M t_i)$	Remarks
False Negatives		
<i>Peace</i> : Melody 	.4	2-voice melody track.
<i>Freedom Jazz Dance</i> : Melody 	.2	Instrumental melody, two octave range.
False Positives		
<i>Limehouse Blues</i> : Soloist 	1.0	Monophonic instrument track.
<i>Ay, Arriba!</i> : Strings 	.6	Polyphonic String ensemble track with large monophonic parts.

Table 3.7: JZ200 classification error examples

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

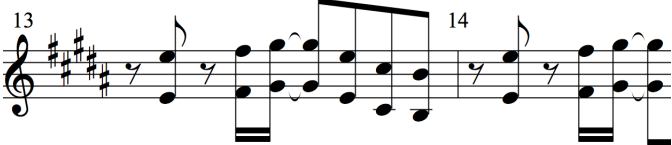
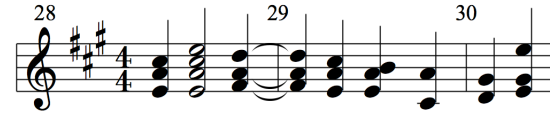


Song : Track	$p(M t_i)$	Remarks
False Negatives		
<i>This Kiss</i> : Melody 	0	Melody in octaves, scatter 4-voice polyphony.
<i>He is your brother</i> : Melody 	0	Up to 3-4 voice polyphony. Top voice is the melody. 45% polyphonic rate.
False Positives		
<i>Cenizas</i> : Flute 	1	Plays intro and melody response parts. Occupation rate about 50%
<i>Solamente una vez</i> : Trumpet 	.7	Plays melody counterparts throughout the song.

Table 3.8: KR200 classification error examples

CHAPTER 3. MELODY PART SELECTION

The experiment on the CL200 corpus reported 8 misclassified tracks from 6 different songs. Take the *Prelude #1 in C Major* melody, from Bach, for example. This *arpeggiato*-like melody has served as accompaniment in another context, the *Ave Maria* song by Charles Gounod⁶. Also, as in the two first false positives examples in Table 3.6, it is not uncommon to find accompaniment parts in classical music where melody parts are mimicked in certain sections. It seems difficult, for models built on global statistics like the ones in this work, to adequately capture such track traits.

The KR200 dataset proved to be the more challenging one for the RF classifier. In particular, recall on melody tracks is low; 56 out of 206 melody tracks were not correctly classified. From these, 12 errors were due to draws, where RF assigned equal probabilities to each class, choosing the most frequent one (*IsMelody* = false) as the default option.

An informal musical analysis of the traits of misclassified tracks leads to the following general findings:

False negatives Most tracks contain some form of polyphony; either melody in octaves, or two-voice melodies or even coral-like parts. Also, some tracks were found where accompaniment-like sections are played.

False positives The most common finding were tracks mimicking the actual melody, or tracks having significant monophonic melody-like sections, like melody response, backing voices or melody counterparts.

Melody part selection experiment

In this second experiment, the goal was to test whether the method selected the proper melody track from a MIDI file. The system was trained the same way as in the previous one, but now a test sample is not a single track but a MIDI file. Due to the limited number of songs available (200 per corpus), this experiment was performed using a leave-one-song-out scheme at the MIDI file level to estimate the classification accuracy. The RF classifier has been chosen for this task. p_ϵ was set empirically to 0.25 (see next section for a discussion on the p_ϵ parameter). Results are shown in Table 3.9.

Note the high quality of the results for CL200 and JZ200. However, a lower success rate has been obtained for the karaoke files. This is partially

⁶[http://en.wikipedia.org/wiki/Ave_Maria_\(Gounod\)](http://en.wikipedia.org/wiki/Ave_Maria_(Gounod))

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

Corpus	Success	Type 1	Type 2	Type 3
CL200	99.0%	2	0	0
JZ200	99.0%	1	0	1
KR200	84.5%	14	4	13

Table 3.9: Melody track selection success and errors (leave-one-song-out, $p_\epsilon = 0.25$).

due to the higher non-melody track ratio in these files (see Table 3.2). Let us briefly analyze errors in KR200. Table 3.10 shows an example from KR200 for each kind of error. Interestingly, in all songs reporting a type 1 error, the actual melody tracks got $p(M|t_i) > 0$, although not always above p_ϵ . The model produced type 2 errors in 4 out of 9 songs without melody. In all cases, $p(M|t_i) < 0.5$. Type 3 errors (all tracks in a song with melody got $p(M|t_i) < p_\epsilon$) are somewhat too many, especially compared to the other corpora. In order to investigate which value is appropriate for p_ϵ , results using different values for this threshold are presented in the next section. However, it will remain an open question in this work whether p_ϵ should be adapted in a per corpus basis.

Genre specificity

It is interesting to know how specific the classifier’s inferred model are with respect to the music genre of files considered for training. For it, two melody part selection sub-experiments were performed: in the first one, the classifier was trained with a 200-file corpus of a given music genre, and tested against all validation corpora (see Table 3.11). This experiment was performed thrice, with different values for the p_ϵ threshold. For the second sub-experiment, the classifier was trained using data from two genres and then tested with files from the third genre validation dataset. (see Tables 3.13 and 3.14).

Let’s first discuss the p_ϵ parameter. As shown in Table 3.11, a near-zero value (0.01) yielded the best results in general. Table 3.12 breaks down the error patterns in each dataset for each threshold value. As expected, setting a high value for p_ϵ increases the number of *Type 3* errors. It also lowers the amount of *Type 1* errors, but not enough to compensate for the, sometimes huge, *Type 3* error rise. It is worth to note that *Type 2* errors made by the CL200 model are almost independent of the threshold. This means that this model assigns $p(M|t_i) \geq 0.5$ to at least one track per file more frequently than the other trained models, for melody-less files particularly!

CHAPTER 3. MELODY PART SELECTION





<i>Song</i> : Track	$p(M t_i)$	Remarks
Type 1 error		
<i>Solamente una vez</i> : Trumpet 10 	.7	Trumpet: monophonic track. Plays melody counterparts throughout the song.
<i>Solamente una vez</i> : Melody 9 	.3	
Type 2 error		
<i>Les murs de poussiere</i> : (lyrics track) 20  Une vil le de filles et de jeux	.3	Track contains lyrics metaevents and their corresponding melody notes with a fixed pitch.
Type 3 error		
<i>Venus</i> : Melody 14  /Burn ing like a sil ver flame	.2	Track contains melody plus lyrics. $p(M t_i)$ is just below p_ϵ (0.25)

Table 3.10: KR200 melody part selection error examples (leave-one-song-out estimation).

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

Train.	p_ϵ	CLA	JAZ	KAR	Avg. diff. genres
CL200	0.01	64.9%	71.8 %	42.7 %	57.3%
JZ200		61.8%	97.2%	51.8%	56.8%
KR200		64.7%	87.1%	88.2%	75.9%
CL200	0.25	57.4%	71.1%	42.3%	56.7%
JZ200		62.2%	96.5%	51.8%	57.0%
KR200		57.2%	70.1%	85.4%	63.6%
CL200	0.5	50.4%	66.7 %	41.2 %	54.0%
JZ200		56.4%	94.8%	51.0%	53.7%
KR200		44.2%	50.0%	76.8%	47.1%

Table 3.11: Genre specific melody part selection accuracy results. Last column shows the average success for genres different from the trained model’s genre.

Recall that a *Type 2* error is produced when a melody track is predicted, but no melody is expected. This means that, in the process of tagging such files, the listener didn’t consider any track to contain a significant amount of melodic material. It could be that in some cases, however, some tracks would actually fit as melody, as predicted by the model. The inherent subjectivity in the ground-truth building process, though, prevents us from drawing some clear conclusions on this issue.

Focusing on results for $p_\epsilon = 0.01$ from herein, Table 3.11 shows that the performance of the system degrades when more complex files are tested. The 200-file corpora are datasets that include MIDI files that were selected among many others for having an ‘easily’ (for a human) identifiable melody track. This holds also for the JAZ corpus, as most jazz music MIDI files have a lead voice (or instrument) track plus some accompaniment tracks like piano, bass, and drums. In fact, each trained model performed at its best on this corpus, regardless the genre it was trained on.

This does not hold in general for the other two corpora. Classical music MIDI files (CLA corpus) come in very different structural layouts, due to both the way that the original score is organized and the idiosyncrasy of the MIDI file authors. This makes this corpus the most difficult dataset for models to tag correctly. Even the classical model was able to perform better on the jazz corpus than on the classical one. This trend is also evidenced for the KAR corpus. Moreover, karaoke files tend to make intensive use of duplicate voices and dense pop arrangements with lots of tracks containing many ornamentation motifs. In addition, we have verified the presence of

CHAPTER 3. MELODY PART SELECTION

Test	Model	p_ϵ	Type 1	Type 2	Type 3
CLA	CL200	0.01	103	70	2
	JZ200		116	68	6
	KR200		88	61	27
	CL200	0.25	80	68	64
	JZ200		112	55	21
	KR200		64	26	123
	CL200	0.5	57	68	122
	JZ200		99	33	85
	KR200		43	16	219
JAZ	CL200	0.01	239	2	0
	JZ200		19	2	3
	KR200		42	2	66
	CL200	0.25	233	2	12
	JZ200		17	2	11
	KR200		9	1	245
	CL200	0.5	206	2	76
	JZ200		13	1	30
	KR200		5	1	421
KAR	CL200	0.01	703	75	0
	JZ200		580	75	0
	KR200		80	71	9
	CL200	0.25	698	75	11
	JZ200		579	74	1
	KR200		52	49	97
	CL200	0.5	680	70	48
	JZ200		570	70	25
	KR200		33	35	247

Table 3.12: Number of classification errors for melody part selection experiments.

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

very short sequences for the CLA corpus, causing less quality in the statistics that also degrades the classification results.

Better results were expected when training on the same genre as the validation dataset. However, the CLA and KAR corpora are definitively harder to deal with, as it became clear in the second experiment presented next in this section. So, it can be said that the difficulty of the task resides more on the particular internal organization of tracks in the MIDI files than on the file music genre, despite that the results in Table 3.11 seem to point out that genre makes a difference.

The KR200 model evidences better generalization capabilities than the other two, as indicated by its average success in other genres (last column of Table 3.11).

Train.	Test	Success
KAR200+JAZ200	CLA	66.2%
CLA200+KAR200	JAZ	89.2%
CLA200+JAZ200	KAR	51.9%

Table 3.13: Melody track selection across genres ($p_\epsilon = 0.01$).

Model	Test	Type 1	Type 2	Type 3
JAZ200+KAR200	CLA	89	65	14
CLA200+KAR200	JAZ	78	2	12
CLA200+JAZ200	KAR	578	75	0

Table 3.14: Number of classification errors for melody part selection across genres ($p_\epsilon = 0.01$).

Results from the second sub-experiment, melody part selection across genres, shown in Tables 3.13 and 3.14, indicate that performance is poorer (with respect to the first experiment) when no data from the test genre were used for training. This does not happen in classical music, probably due to effects related to the problems expressed above.

Training set specificity

To see how conditioned these results are by the particular training sets utilized, a generalization study was carried out building a new training set merging the three 200-files corpora (named *ALL200*), and then using the

CHAPTER 3. MELODY PART SELECTION

other corpora for test. The problem to solve is again the one discussed in section 3.1.6: selecting the proper melody track from a MIDI file. The results are detailed in Tables 3.15 and 3.16.

They show that, when using a multi-genre dataset, the performance of the system is improved, specially for the KAR dataset. Now the training set contains samples from the same genre as the test dataset. Note that *Type 3* errors have dropped in general, while the number of *Type 1* errors has decreased for the JAZ and KAR datasets. Also, the presence or absence of the test genre in the training dataset has no influence on *Type 2* errors.

Training	Test	Success
ALL200	CLA	64.5%
ALL200	JAZ	97.8%
ALL200	KAR	83.8%

Table 3.15: Melody track selection by genres when training with data from all genres ($p_\epsilon = 0.01$).

Model	Test	Type 1	Type 2	Type 3
ALL200	CLA	102	69	6
ALL200	JAZ	12	2	5
ALL200	KAR	144	75	1

Table 3.16: Number of classification errors for melody part selection training with all genres ($p_\epsilon = 0.01$).

When combining all results from Table 3.15, taking into account the different cardinalities of the test sets, the average successful melody track selection percentage is 84.6%.

Conclusions on melody part selection experiments

Figure 3.4 shows a summary of the results from all melody part selection experiments. The obvious conclusion is that the classification method performs better for a given genre when samples of this genre are used for training the underlying model, except for the CLA dataset. It reveals itself as the hardest dataset to deal with, no matter which training data are used. Comparing results shown in Tables 3.9 and 3.12 for the CL200 model, it can be concluded that it does not generalize well. It seems that this

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

approach based on global statistical descriptors does not work well enough for MIDI-encoded classical music. The task is probably better undertaken applying some kind of melody extraction technique (Isikhan and Ozcan, 2008; Uitdenbogerd and Zobel, 1998).

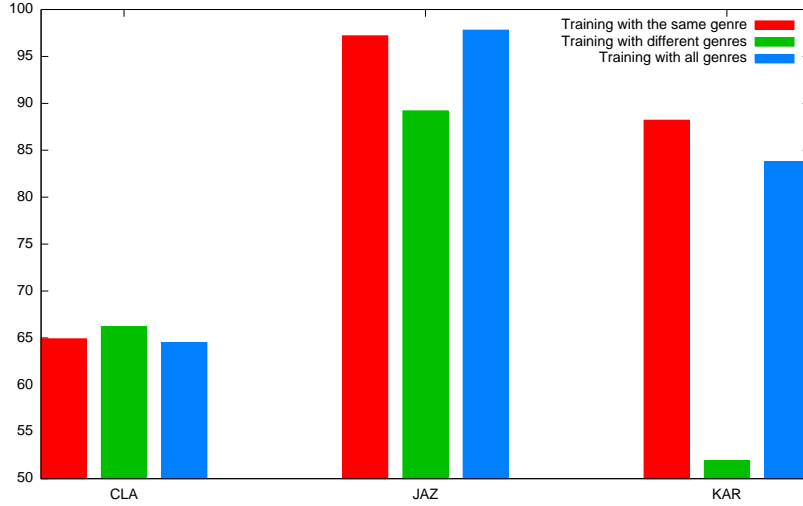


Figure 3.4: Melody track selection results. Accuracy comparison according to genre inclusion in training set ($p_\epsilon = 0.01$).

The JAZ dataset was definitively the easiest to process. The presence of jazz samples in the training data has proven to improve the results, up to a 97.8% success ratio. However, even without those training samples, the system was able to achieve reasonably good performance.

Selecting melody tracks from files in the KAR corpus is particularly sensitive to the presence of similar samples in the training set. A dramatic improvement (up to 36%) was seen when such samples are used in the training phase, compared to using samples from *a priori* different genres.

In summary, in the experiments presented here, using a model built exclusively on the genre of test data has proven to be a good and simple approach for melody part selection. In the case of classical music, however, comparable performance is achieved using genres different from classical but, as discussed above, a melody extraction approach could reveal to be better suited for this music genre.

3.1.7 A graphical interface for melody part selection

The RF melody track classifier has been used as the base model for a graphical interface for melody part selection. This application can be used to train and test the classifier. In ‘test’ mode, a list of MIDI files are opened, statistical features describing the content of their tracks are extracted and used to feed the classifier. Results from the classifier are gathered and melody part selection results are displayed in a graphical way. A snapshot of the interface is shown in Figure [3.5](#). A more detailed description of such application can be found in appendix [A.2](#).

3.1. MELODY CHARACTERIZATION IN MULTI-PART FILES

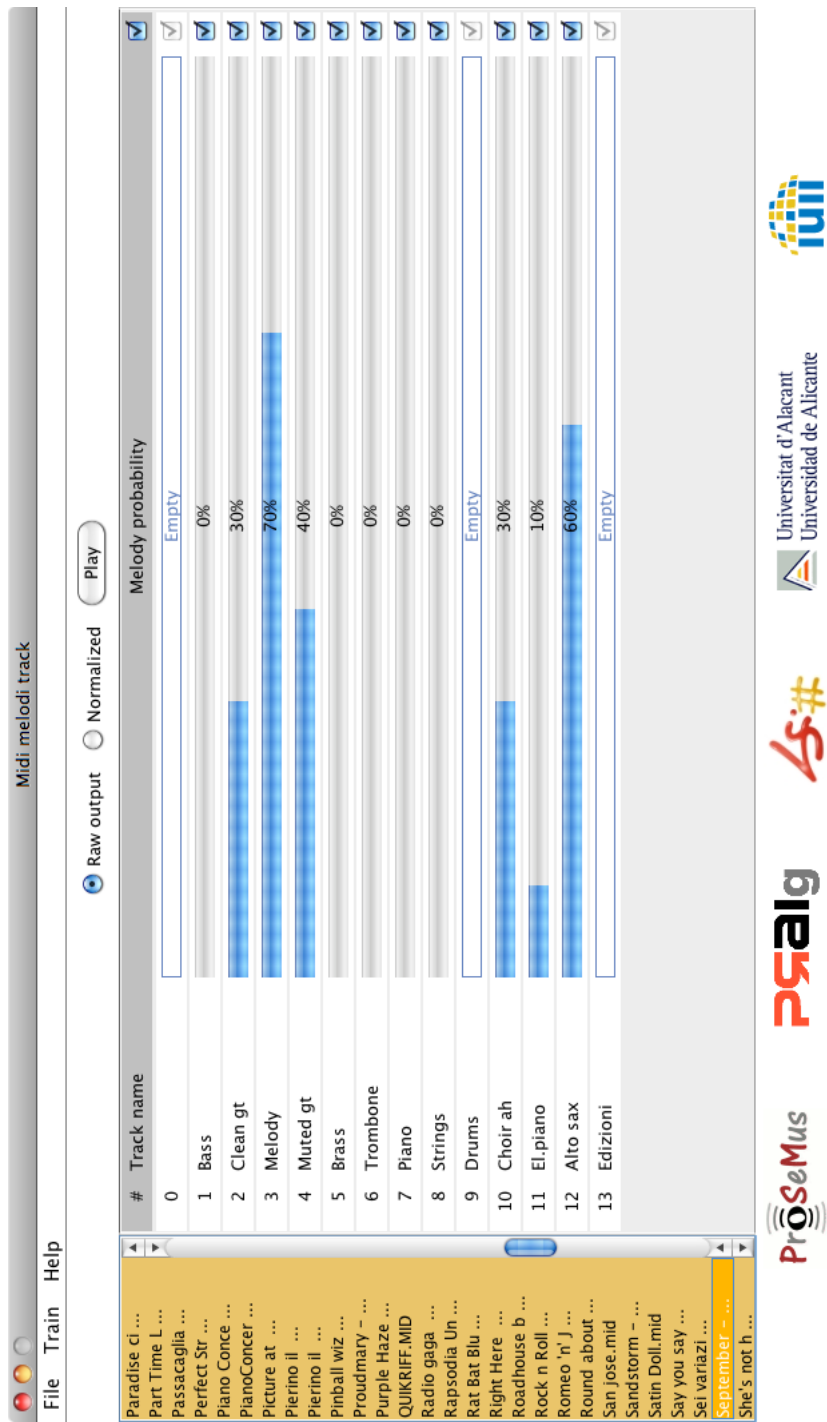


Figure 3.5: A snapshot of the melody part detection graphical interface.

3.2 Towards a human-friendly model for melody part selection

Random forest trees built for melody track characterization have hundreds of leaves each. For this reason, the melody model they provide, although explicit, is complex and nearly incomprehensible, and so it makes difficult to draw conclusions about their performance. In practice, they are working like a black box. Therefore, it would be interesting to extract a simpler model from the RF trees without significant loss of performance. This more compact model will be more suitable for further analysis and comparison with other rule-based models. It will be also more useful for the potential user of a melody characterization system, since a compact representation of the characterization can be presented to her in order to understand the decision taken by the system.

The answer is given as sets of rules that are both automatically learnt from score corpora and human-readable. This would allow us to end up with an objective description of what a melody should be. This could be of interest for musicologists or helpful in applications such as melody matching, motif extraction, melody extraction (ringtones), etc.

The rest of the section presents the methodology applied to extract a compact rule system representation from RF decision trees. Then, the data used and the experiments designed to test the method are discussed.

3.2.1 Methodology

The steps performed to obtain sets of rules from decision trees that characterize melody tracks can be outlined in these main steps:

Rule extraction Extract rules from decision tree branches.

Rule simplification Prune rule antecedents.

Rule selection Select best rules by ranking.

The rule extraction step is performed on the decision trees built by the Random Forest algorithm. The rules are simplified by pruning non useful antecedents and then, for each rule set, rules are ranked according to a scoring function. The objective here is to obtain a compact rule system with performance comparable to that of the original decision trees they are derived from.

Expressing a concept by rules has the advantage of being a human-readable description of the characterization process. The simplification and

3.2. HUMAN-FRIENDLY MELODY PART SELECTION

ranking steps focus on obtaining a small, manageable rule system. A side goal has been to test whether a small number of selected rules can perform comparably to an ensemble of decision trees, typically containing hundreds of nodes per tree.

Rule extraction from decision trees

In section 3.1, Random Forest classifiers were used to learn an ensemble of K decision trees capable of discriminating melody tracks in a MIDI file. The leaves of the decision trees are tagged with a boolean *IsMelody* value, indicating whether the leaf characterizes a melody track or a non-melody one. So there are *positive* leaves, the ones with a *true* tag, and *negative* leaves. For each tree, a rule set is extracted following positive branches from the root (the ones leading to a positive leaf). Negative branches are ignored. From each positive branch a rule is obtained of the form:

$$(X_1^i(\mathbf{s}) \wedge X_2^i(\mathbf{s}) \wedge \dots \wedge X_{n_i}^i(\mathbf{s})) \implies IsMelody = \mathbf{true}$$

where $X_j^i(\mathbf{s})$ are the tests found in each tree node traversed following a positive branch. Such tests, applied on a sample $\mathbf{s} = (s_1, s_2, \dots, s_d, \dots, s_n) \in \mathbb{R}^n$, have the general form $(s_d \gtrless v)$ that represents an inequality test involving only one descriptor s_d and one value v . As all rules have the same consequent, we will drop it from herein, thus leaving us with the following definition of $r^i(\mathbf{s})$:

$$r^i(\mathbf{s}) = \bigwedge_{j=1}^{n_i} X_j^i(\mathbf{s})$$

which is not properly a rule, but a boolean function definition. However, I will use the term *rule* throughout the text when referring to r^i .

A rule set R_k is defined as a logical disjunction of all rules r_k^i extracted from tree T_k :

$$R_k(\mathbf{s}) = \bigvee_{i=1}^{|R_k|} r_k^i(\mathbf{s})$$

Track characterization by rule set ensemble When such a rule set is applied to a sample, firing at least one rule suffices for that sample to be tagged as a melody track. A rule set R is thus a function $R : S \longrightarrow \{0, 1\}$ applied on the domain S of MIDI track samples. A model built as an ensemble of such rule sets is a conditional probability function defined as

CHAPTER 3. MELODY PART SELECTION

$$p(M|\mathbf{s}) = \frac{\sum_{j=1}^K R_j(\mathbf{s})}{K}$$

As in section 3.1.4, $p(M|\mathbf{s})$ represents the probability for a MIDI track, represented here by \mathbf{s} , to be a melody track.

Rule simplification by antecedent pruning

The number of conditions in the antecedent part of a rule can be too large to be easily understood. Moreover, complex rules are often very specific, overfitting the training set. A rule can be generalized by dropping some of the conditions from its antecedent. This technique is known as *antecedent pruning*.

The method for pruning rule antecedents performs a test for consequent's independence from each condition X_i for each rule (Quinlan, 1999). In particular, a χ^2 test with a 95% confidence interval is performed here, considering condition relevance as independent from other conditions in the same rule. This makes the test to be very conservative, dropping only conditions that do not satisfy the hypothesis for all validation samples. A validation dataset different from the initial training set is used to test rule conditions.

For each condition X_i in a rule, a contingency table is built, like in Table 3.17.

	Melody	Not Melody
$X_i = \text{true}$	o_{11}^i	o_{12}^i
$X_i = \text{false}$	o_{21}^i	o_{22}^i

Table 3.17: Contingency table for a condition X_i . Rows indicate whether the condition is true or not. Columns indicate whether the samples are melodies or not. o_{jk}^i are the number of samples meeting the criteria.

Once the o_{jk}^i values from the table are known for a given condition X_i , proceed as follows:

1. Compute expected frequencies e_{jk}^i for each contingency table cell:

$$e_{jk}^i = (RT_j^i \times CT_k^i) / T^i \quad (3.5)$$

where

$$RT_j^i = \sum_{k=1,2} o_{jk}^i, \quad CT_k^i = \sum_{j=1,2} o_{jk}^i, \quad T^i = \sum_{j,k} o_{jk}^i$$

3.2. HUMAN-FRIENDLY MELODY PART SELECTION

2. Perform a χ^2 test to determine if the conclusions are independent from the antecedent at the 95% level of significance ($\chi^2_{\alpha=0.05} = 3.841$)⁷

$$\chi^2 = \sum_j \sum_k (o_{jk}^i - e_{jk}^i)^2 / e_{jk}^i \quad (3.6)$$

3. If $\chi^2 > \chi^2_{\alpha=0.05}$ then keep antecedent else prune antecedent

Pruning rule antecedents considering condition relevance dependent from other ones in the same rule has not been considered, due to performance reasons. The order in which conditions are tested is important, as the relevance of a condition can be determined by another one which is in turn not relevant. With the lack of *a priori* information about which antecedents are more likely to be relevant, a straight forward approach would be to test antecedents in all possible orderings, keeping the solution with less relevant conditions. This has time complexity $O(n!)$ on the number of antecedents of a rule. Also, a criteria would be needed to resolve ties between equally sized solutions. Another approach would be to use an instance of the OPUS algorithm for unordered search (Webb, 1995), that often produce searches with polynomial time complexity in the average case.

Rule selection

Decision trees learnt by the RF classifier from large training sets are usually big, leading also to huge rule sets⁸. In order to get a compact model, it is desirable to remove rules that do not characterize a lot of samples. Furthermore, some rules could be redundant, as they fire together with other rules for most input samples, thus making their presence in the set unnecessary. The method used here to reduce the size of a rule set consists of ranking rules according to a measure based on how many samples from a validation dataset fire them. The more samples fire a rule, the better the rule. After the ranking is made, the best rules are selected from each rule set in order to classify new samples, discarding the rest.

The procedure used here to rank the rules in a rule set R using a validation dataset D is as follows:

1. Sort rules in R decreasingly, according to the number of samples in the training set firing each rule.

⁷for one degree of freedom

⁸In our experiments, trees with more than five hundred leaves (and thousands of nodes) have been obtained.

CHAPTER 3. MELODY PART SELECTION

2. For each rule r^i in R :⁹

$$(a) \ r_s^i = s(r^i, D)$$

$$(b) \ D = D - r_{\oplus}^i - r_{\ominus}^i$$

3. Sort R according to r_s , in decreasing order.

4. Select the first N rules of R and discard the rest.

Sorting R in the first step of the procedure results in *a priori* ‘best’ rules to be evaluated first. The number of samples in the training set that fire a rule is provided by the RF model representation, so there is no need to compute this values again. This is a greedy approach we have found to produce similar results than finding the best rule with respect to the validation set at each iteration, but at a less computational cost. The greedy approach has $O(|R|(\log|R| + |D|))$ time complexity, and the second one has $O(|R|^2|D|)$ complexity.

In this work, two scoring functions have been tested:

- $s_1(r^i, D) = r_{\oplus}^i$
- $s_2(r^i, D) = r_{\oplus}^i / (r_{\oplus}^i + r_{\ominus}^i)$

Additionally, a variant of the ranking procedure that does not remove samples from D (step 2b) has been tested along with the s_1 function. We denote this variant as s_0 .

The same validation dataset is used for all rule sets in the rule system derived from the decision trees.

3.2.2 Experiments and results

For the experiments, the rule systems have been derived from an ensemble of decision trees built using a *RF* classifier with $F = 5$ features and $K = 10$ trees. Therefore, rule systems consist of ten rule sets. There is one rule system per training corpus.

⁹ r_{\oplus}^i is the number of positive samples (melody tracks) in D that fire rule r^i and r_{\ominus}^i is the number of negative samples (non-melody tracks) in D that fire rule r^i .

3.2. HUMAN-FRIENDLY MELODY PART SELECTION

Datasets

Five corpora have been used: *ALL200*, *LARGE*, *RWC-G*, *RWC-P* and *AJP*. The corpus *ALL200* is described in section 3.1.6. The *LARGE* corpus is the union of the *CLA*, *JAZ* and *KAR* corpora described in section 3.1.6. These two corpora have been used to train and validate the system. The *RWC* corpora, used to test the system, are described in section 2.8.4.

The *AJP* corpora is a two level multi-genre corpus. At the first, coarse level, there are three genres: classical, jazz and popular music. At the second level there are eight sub-genres: baroque, classicism, romanticism, pre-bop, bop, bossa-nova, country and pop-rock. Tracks in all corpora (see detail in table 3.18) have been manually tagged as melody or non-melody tracks, following the guidelines discussed in section 3.1.6.

	Corpus	Files	Tracks	Tracks per file	Melodies	Non-melody ratio	Validation set
Train	ALL200	600	2775	4.6	554	5.2	LARGE
	LARGE	2513	15168	4.9	2337	5.7	ALL200
Test	RWC-G	48	311	7.5	44	6.1	—
	RWC-P	75	801	11	74	10	—
	AJP	762	3732	5.9	760	5.2	—

Table 3.18: Corpora for rule-based melody track selection.

Antecedent pruning results

Once rules are extracted from positive branches of the RF decision trees, non-significant antecedents are pruned using the method discussed in section 3.2.1. The rule systems obtained from the decision trees are named after the dataset used for training the corresponding RF classifier. The validation sets used for antecedent pruning are the ones specified in Table 3.18. The results of the antecedent pruning process are summarized in Table 3.19.

CHAPTER 3. MELODY PART SELECTION

Rule system	ALL200	LARGE
Unique conditions	779 (816)	1878 (2297)
% pruned unique conds.	4.5%	18.2%
Conditions	4370 (4692)	22481 (25581)
Avg. condition/rule	8.3 (9)	10.8 (12.3)
% pruned conditions	6.9%	12.1%

Table 3.19: Rule antecedent pruning results. Numbers in parentheses indicate number of conditions in the rule antecedents before pruning.

As the figures in that table are not that impressive, recall from section 3.2.1 that the testing procedure is very conservative, removing only conditions that are not relevant for the whole validation dataset.

For the experiments that follow, the resulting pruned rule systems are used.

Rule ranking results

Table 3.20 summarizes rule set coverage of the validation dataset when considering all rules in a rule set. Rule coverage percentages are a rough estimation of a rule system accuracy. In this case, both rule systems perform comparably on their respective validation datasets, with more than 85% of the melody tracks firing at least one rule in a rule set. However, the coverage of non-melody tracks is somewhat high, especially for the *LARGE* rule system.

Rule system	ALL200	LARGE
melody tracks	86% (3)	87% (4)
non-melody tracks	17% (7)	32% (21)
zero scoring	23%	67%

Table 3.20: Rule coverage summary. Figures in the second and third rows are average coverage of the validation dataset by a rule set from the corresponding rule system. Numbers in parenthesis are standard deviations.

The last row shows the percentage of rules in a rule set, in average, that scored zero, i.e. rules not fired by any melody track in the validation set. These rules are dropped from the rule sets, thus reducing their size proportionally. This is a great reduction for the *LARGE* rule system, as it

3.2. HUMAN-FRIENDLY MELODY PART SELECTION

shrinks to one third of its original size. This means keeping about 70 rules out of more than 200 for each rule set, in average.

Figure 3.6 shows the average rule set coverage percentages when selecting the N best rules from each rule set in a rule system. Note that when N is small, the s_1 function gives a better coverage ratio for melodies. On the other hand, the s_2 function gives better coverage ratios for large N . In particular, the *LARGE* rule system achieves more than 75% melody track coverage in average while maintaining the non-melody track coverage very low (about 3%).

A hint on what combination of scoring function and N value is the best can be seen on Figure 3.7. This is much like a ROC curve, with false positive rate on x -axis and true positives on the y -axis, so the perfect model is the one at point $(0, 1)$. The closer to that point, the better the rule system. The arc of circumference with origin at $(0, 1)$ intersects the curve for rule systems derived from the *ALL200* dataset at $(0.09, 0.84)$ for $N = 32$ with scoring function s_2 . Recall that the plot presents average coverage from each rule set in that rule system. The standard deviation for both TP and FP rates is 0.03. However, as the goal is to obtain a compact model for melodies, this N value is still too large. This ‘best’ model is used for comparison purposes on the classification experiments presented in the next section.

Track selection experiments

Two experiments are presented in this section. The first one is a melody track categorization experiment, while the other one is a melody part selection experiment (see section 3.1.6 for recalling experiment definitions). For each experiment, results from several combinations of scoring function and number of selected rules are discussed. These rule system results are compared to those obtained by the *RF* model from which the rules are derived.

Melody track categorization results The *RWC-G*, *RWC-P* and *AJP* datasets have been used as test sets. The rule systems have three free parameters: the number of best rules N to be selected from each rule set, the minimum number θ of rule sets that must agree to tag a track as a melody, and the scoring function s used to rank the rules. Each combination (N, θ, s) results in a distinct rule system classifier. Recall that there are 10 rule sets in a rule system. The range of values used in these experiments is $N \in [1, 20]$, $\theta \in [1, 10]$, and $s \in \{s_0, s_1, s_2\}$. A summary of the best results obtained applying these classifiers to test datasets is shown in tables 3.21, 3.22, and 3.23. The rule system success (%OK), precision, recall, and F -measure are shown. The F -measure is used as the performance value to

CHAPTER 3. MELODY PART SELECTION

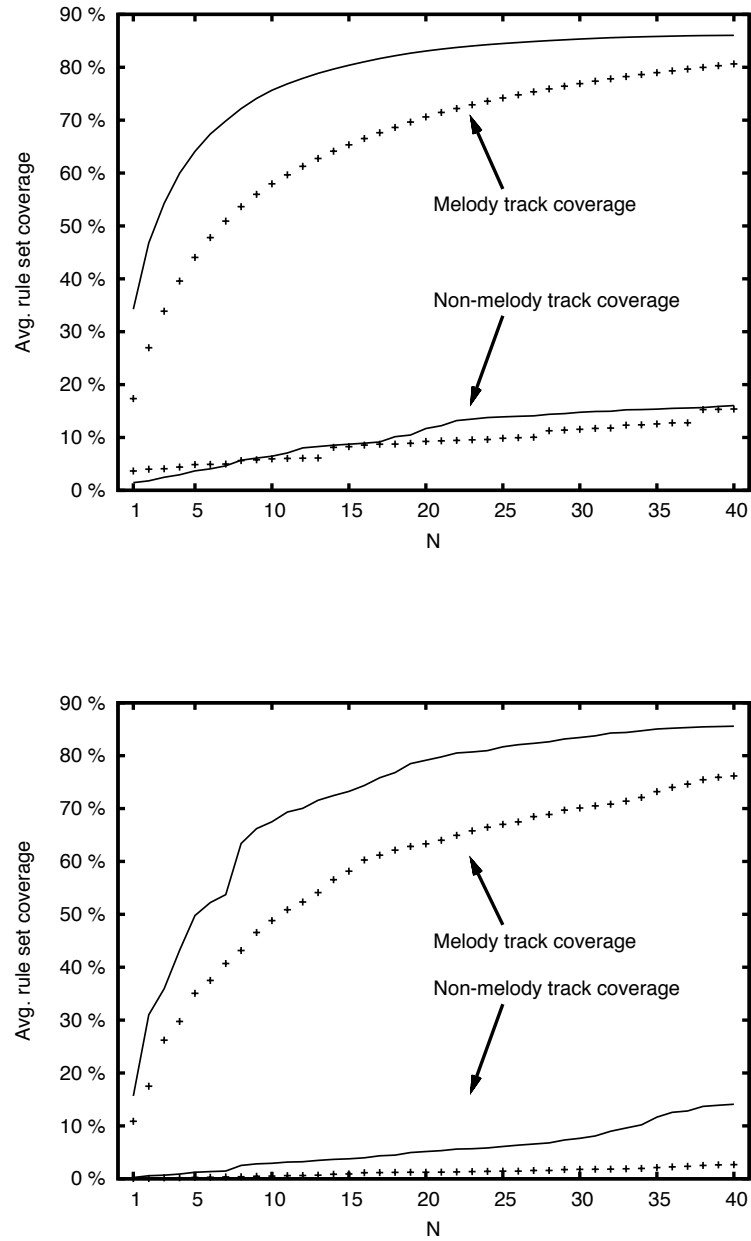


Figure 3.6: Average rule set coverage when the number of rules selected (N) varies from 1 to 40, for both rule systems and score functions s_1 (top) and s_2 (bottom). Solid lines indicate coverage for *ALL200* and (+) indicate coverage for *LARGE*. Plots over 20% are for melody tracks. Plots under 20% are for non-melody tracks.

3.2. HUMAN-FRIENDLY MELODY PART SELECTION

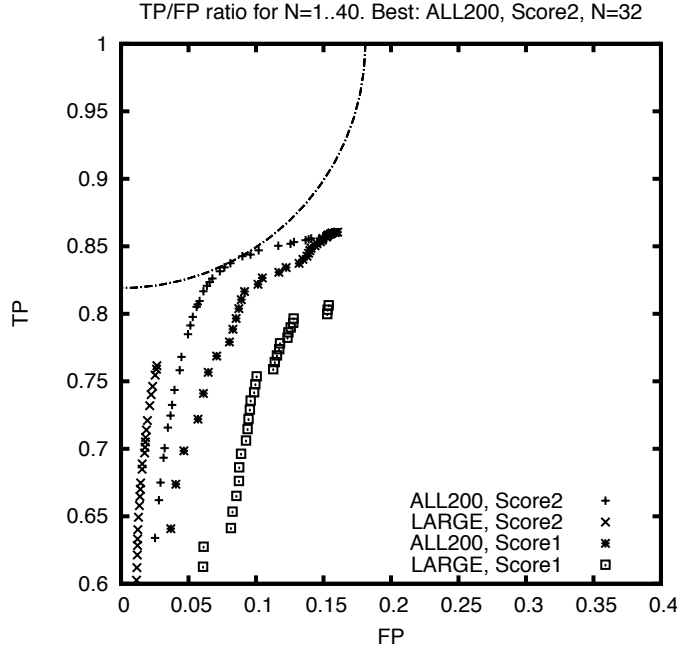


Figure 3.7: TP/FP ratio for average rule coverage.

select the best combinations. The table shows the results for parameter combination with best trade-off between low N values and F -measure for each scoring function. Note that $\theta = 6$ for the RF results, indicating that a sample is assigned the class $IsMelody=true$ if the RF assigns a probability greater than 0.5 to this class. The last row in every table shows results obtained when using the best rule system according to Figure 3.7.

Some interesting results on melody track categorization are shown in Table 3.24. This table shows the best results for all test datasets when $N = 1$. As shown, the *ALL200* model with score s_0 performs best when $N = 1$. Table 3.26 shows the first rules in rule sets from rule system *ALL200*, according to score s_0 . When it comes for the user to better understand what kind of tracks do the rules in that table characterize, it is not obvious what the actual numerical values mean. An attempt at making such rule systems more human-readable is described in section 3.3.

Melody part selection results Now, the goal is to know how many times the method selects the correct melody track among those in a file. See section 3.2.2 for details on this problem definition.

Rule systems *ALL200* and *LARGE* were used. The task is approached by fixing the θ parameter to 1. The track in a file that fires most rule sets is

CHAPTER 3. MELODY PART SELECTION

Model	s	N	θ	%OK	Prec	Rec	F
ALL200 (rules)	0	2	2	91.3	0.67	0.75	0.71
	1	2	2	90.4	0.64	0.73	0.68
	2	10	4	89.4	0.60	0.80	0.68
ALL200(RF)	–	–	6	90.4	0.65	0.68	0.67
LARGE (rules)	0	17	4	89.1	0.60	0.66	0.63
	1	11	4	90.7	0.67	0.65	0.67
	2	8	3	92.3	0.76	0.66	0.71
LARGE(RF)	–	–	6	92.0	0.71	0.72	0.72
ALL200 (Best)	2	32	5	86.2	0.51	0.84	0.63

Table 3.21: Melody track categorization. Best results for the *RWC-G* dataset. (RF = Random Forest)

	s	N	θ	%OK	Prec	Rec	F
ALL200 (rules)	0	1	3	98.1	0.89	0.91	0.90
	1	5	6	98.1	0.90	0.89	0.90
	2	2	2	98.4	0.88	0.96	0.91
ALL200(RF)	–	–	6	97.8	0.84	0.93	0.89
LARGE (rules)	0	3	3	98.6	0.91	0.95	0.93
	1	9	4	98.8	0.90	0.97	0.94
	2	5	2	98.8	0.90	0.97	0.94
LARGE(RF)	–	–	6	98.6	0.90	0.97	0.94
ALL200 (Best)	2	32	7	97.8	0.84	0.93	0.89

Table 3.22: Melody track categorization. Best results for the *RWC-P* dataset.

selected as the melody. Note that this is equivalent to applying Eq. 3.4 with $p_\epsilon = 0.1$, as $p(M|t_i)$ is computed here as the ratio of fired rule sets in the rule system. In case of draws, the track that activates the larger total number of rules wins the tie. If a draw still exists after comparing the number of rules fired by each track, one of them is chosen randomly as the melody track.

Table 3.25 shows the best rule system results and those obtained using *RF* classifiers. The results for the *RWC-P* dataset show a hint that rule systems with a few tens of rules can perform comparably to an ensemble of decision trees with hundreds of leaves.

3.2. HUMAN-FRIENDLY MELODY PART SELECTION

	s	N	θ	%OK	Prec	Rec	F
ALL200 (rules)	0	17	3	95.3	0.86	0.92	0.89
	1	15	4	95.6	0.88	0.91	0.89
	2	12	2	94.6	0.86	0.88	0.87
ALL200(RF)	–	–	6	95.9	0.90	0.89	0.90
LARGE (rules)	0	13	2	95.1	0.88	0.89	0.88
	1	17	3	95.6	0.88	0.90	0.89
	2	11	1	95.3	0.88	0.89	0.89
LARGE(RF)	–	–	6	94.0	0.92	0.77	0.84
ALL200 (Best)	2	32	5	95.6	0.88	0.91	0.89

Table 3.23: Melody track categorization. Best results for the *AJP* dataset.

Dataset	Rule sys.	s	θ	%OK	Prec	Rec	F
<i>RWC-G</i>	ALL200	0	2	91.0	0.69	0.66	0.67
<i>RWC-P</i>	ALL200	0	3	98.1	0.89	0.91	0.90
<i>AJP</i>	ALL200	0	1	93.2	0.81	0.87	0.84

Table 3.24: Melody track categorization. Best results with $N = 1$.

	<i>RWC-G</i>			<i>RWC-P</i>			<i>AJP</i>		
	s	N	%OK	s	N	%OK	s	N	%OK
ALL200	1	3	70.5	0	2	97.3	1	10	91.5
<i>ALL200 (RF)</i>	–	–	75.0	–	–	<i>94.7</i>	–	–	98.6
LARGE	1	5	68.2	0	6	97.3	0	7	86.3
<i>LARGE (RF)</i>	–	–	72.9	–	–	<i>96.0</i>	–	–	97.5

Table 3.25: Best melody part selection results for test datasets.

RS	First rule
1	$(TrackOccupationRate \geq 0.32)$ $\wedge (LowestPitch \geq 53.5) \wedge (AvgNormalizedPitch < 0.37)$ $\wedge (MostRepeatedInterval < 12.5) \wedge (DistinctIntervals \geq 21.5)$
2	$(TrackOccupationRate \geq 0.42) \wedge (TrackPolyphonyRate < 0.28) \wedge (TrackNumNotes < 569.5) \wedge (TrackNumNotes \geq 178.0)$ $\wedge (LowestNormalizedPitch \geq 0.14) \wedge (AvgPitch \geq 56.8)$ $\wedge (HighestAbsInterval < 18.0) \wedge (AvgAbsInterval < 3.11)$ $\wedge (AvgDuration < 1.58)$
3	$(TrackNormalizedDuration \geq 0.92)$ $\wedge (HighestPitch \geq 72.5) \wedge (AvgPitch \geq 65.06) \wedge (StdDeviationPitch \geq 3.37)$ $\wedge (LowestAbsInterval < 0.5) \wedge (AvgAbsInterval < 4.37)$ $\wedge (HighestNormalizedDuration \geq 0.06)$
4	$(TrackOccupationRate \geq 0.42) \wedge (TrackNormalizedSyncopation \geq 0.03) \wedge (TrackNormalizedDuration \geq 0.17) \wedge (TrackPolyphonyRate < 0.01)$ $\wedge (LowestPitch \geq 55.5)$ $\wedge (AvgNormalizedAbsInterval < 0.12)$ $\wedge (AvgNormalizedDuration < 0.21)$
5	$(TrackNormalizedSyncopation \geq 0.06) \wedge (TrackOccupationRate \geq 0.42) \wedge (TrackPolyphonyRate < 0.08)$ $\wedge (LowestPitch \geq 54.5) \wedge (NormalizedStdDeviationPitch < 0.66)$ $\wedge (NormalizedStdDeviationAbsInterval < 0.02)$
6	$(TrackNormalizedSyncopation \geq 0.08) \wedge (TrackNumNotes \geq 261.5) \wedge (TrackSyncopation \geq 29.0)$ $\wedge (LowestPitch \geq 47.5) \wedge (StdDeviationPitch < 6.99)$ $\wedge (StdDeviationAbsInterval < 2.86) \wedge (AvgAbsInterval < 3.52) \wedge (AvgAbsInterval \geq 0.62) \wedge (HighestAbsInterval \geq 6.5)$
7	$(TrackPolyphonyRate < 0.26) \wedge (TrackOccupationRate \geq 0.61)$ $\wedge (LowestPitch \geq 52.5) \wedge (NormalizedStdDeviationPitch < 0.32) \wedge (StdDeviationPitch < 5.27)$ $\wedge (AvgNormalizedAbsInterval < 0.24)$ $\wedge (AvgDuration < 1.13)$
8	$(TrackPolyphonyRate < 0.32) \wedge (TrackNormalizedSyncopation \geq 0.06) \wedge (TrackNumNotes \geq 67.0) \wedge (TrackOccupationRate \geq 0.67)$ $\wedge (AvgPitch \geq 60.59)$ $\wedge (AvgAbsInterval < 4.39)$ $\wedge (NormalizedStdDeviationDuration \geq 0.98)$
9	$(TrackOccupationRate \geq 0.42) \wedge (TrackPolyphonyRate < 0.19)$ $\wedge (AvgNormalizedPitch \geq 0.26) \wedge (LowestPitch \geq 50.5) \wedge (LowestPitch < 71.0)$ $\wedge (StdDeviationAbsInterval < 5.23) \wedge (AvgNormalizedAbsInterval < 0.12)$ $\wedge (TrackNumNotes \geq 257.0)$
10	$(TrackNormalizedSyncopation \geq 0.11) \wedge (TrackPolyphonyRate < 0.13) \wedge (TrackSyncopation \geq 29.0)$ $\wedge (HighestNormalizedPitch \geq 0.33) \wedge (AvgPitch \geq 60.88)$ $\wedge (AvgAbsInterval < 4.12) \wedge (HighestNormalizedAbsInterval < 0.76)$

Table 3.26: First rules from rule system ALL200, with scoring function s_0 (RS=Rule set).

3.3 A fuzzy approach

The rule systems built so far are more compact melody models and can perform comparably to RF on some datasets. However, as it can be seen in table 3.26, they are still hard to understand, as the meaning of numerical values is not always easily readable. For example, is a *LowestPitch* value of 53.5 high or low? Moreover, taking the first rule in the table to characterize a sample track, it would not fire if the track has a *LowestPitch* value equal to 53, no matter whether the track meets the rest of conditions in the rule antecedent. These two problems can be circumvented by the use of *fuzzy rules*. Preliminary work on automatic human-readable melody characterization by fuzzy rules is presented in this section. A method to convert a *crisp* classification rule system for melody track characterization into a fuzzy rule system is laid out. The method applies a genetic algorithm (GA) to generate fuzzy membership functions for rule attributes. These fuzzy models should achieve good performance in discriminating melody tracks when compared to the crisp models they are derived from.

The rest of the section is organized as follows: first, the methodology used is discussed. Second, the experimentation framework is outlined. Results on several datasets for both crisp and fuzzy rule systems are discussed and compared.

3.3.1 Methodology

The methodology applied to obtain such fuzzy models is outlined in two steps: first, a crisp rule system that characterize melody tracks is learned from tagged corpora, like the ones presented in section 3.1.6. This system is then converted to a fuzzy one applying a *fuzzification* process to the input domain. This is discussed in this section.

A rule system for melody characterization

A rule system obtained by the RIPPER algorithm (Sec. 2.2.5) is used as the basis to induce a fuzzy rule system. The ALL200 dataset (Sec. 3.1.6) is used for training, so it is called the RIPPER-ALL200 crisp rule system from herein. The parameters used for the algorithm setup are shown in table 3.27. Table 3.28 shows the RIPPER-ALL200 rule system. Note that only 13 out of 34 in the initial set of statistical descriptors have been selected by the algorithm to characterize melody tracks. Figures about this rule system performance will be presented in section 3.3.3.

CHAPTER 3. MELODY PART SELECTION

Number of folds for training/pruning	3
Minimum per-class rule antecedent coverage ¹⁰	2
Number of optimization runs	2

Table 3.27: RIPPER setup parameters

From crisp to fuzzy rule systems

Although informative, RIPPER-ALL200 is not easily understandable at a first sight, at least for musicians or musicologists. Also, being *melody* such a vague concept, a fuzzy description of melody would be more sensible in the imprecise domain of music characterization. In order to produce such a fuzzy description, a fuzzification process is applied to the crisp rule system. Such process is performed in two steps. First, the data representation must be fuzzified. That is, numerical input and output attributes must be converted to fuzzy variables. Second, the rules must be translated into fuzzy rules, substituting linguistic terms for numerical boundaries.

Attribute fuzzification

The procedure to fuzzify a crisp attribute is outlined as follows:

1. Define crisp attribute domain.
2. Define linguistic terms (such as *low*, *average*, or *high*) for every domain.
3. Select the shape of the fuzzy membership function (or fuzzy set, for short) associated with each linguistic term.
4. Set the value of each fuzzy set parameter within the attribute domain.

Crisp attribute domain definition As stated above, a MIDI track is described by a set of statistical descriptors (called attributes from herein). As seen in Table 3.1, most attributes have a finite domain. For practical application of the fuzzification method, infinite domains should be converted to finite domains, defining appropriate bounds for them. As all attribute ranges are in the positive domain, only the upper bound for infinite domains is defined as the value 999,999.0.

¹⁰In the Weka toolkit implementation, this is expressed in terms of instance weights.

3.3. A FUZZY APPROACH

Name	Rule
R1	$(TrackOccupationRate \geq 0.51) \wedge (TrackNumNotes \geq 253)$ $\wedge (AvgPitch \geq 65.0)$ $\wedge (AvgAbsInterval \leq 3.64)$ $\implies IsMelody = \mathbf{true}$
R2	$(TrackOccupationRate \geq 0.42) \wedge (TrackPolyphonyRate \leq 0.21)$ $\wedge (AvgPitch \geq 62.6)$ $\wedge (NormalizedDistinctIntervals \geq 1)$ $\implies IsMelody = \mathbf{true}$
R3	$(TrackNumNotes \geq 284)$ $\wedge (AvgPitch \geq 65.4)$ $\wedge (NormalizedDistinctIntervals \geq 1)$ $\wedge (ShortestNormalizedDuration \leq 0.001) \wedge (ShortestDuration \geq 0.02)$ $\implies IsMelody = \mathbf{true}$
R4	$(TrackOccupationRate \geq 0.42) \wedge (TrackSyncopation \geq 16)$ $\wedge (AvgPitch \geq 60.5) \wedge (StdDeviationPitch \leq 5.0)$ $\wedge (AvgAbsInterval \leq 2.72)$ $\implies IsMelody = \mathbf{true}$
R5	$(TrackNormalizedDuration \geq 0.95) \wedge (TrackSyncopation \geq 24)$ $\wedge (LowestNormalizedPitch \geq 0.14)$ $\wedge (DistinctIntervals \geq 25) \wedge (AvgAbsInterval \leq 3.87)$ $\implies IsMelody = \mathbf{true}$
R6	$(TrackOccupationRate \geq 0.31) \wedge (TrackPolyphonyRate \leq 0.001)$ $\wedge (TrackNumNotes \geq 130)$ $\wedge (AvgPitch \geq 55.2)$ $\wedge (AvgAbsInterval \leq 2.44)$ $\implies IsMelody = \mathbf{true}$

Table 3.28: RIPPER-ALL200 (crisp) rules.

CHAPTER 3. MELODY PART SELECTION

Linguistic term definitions In order to select the number of linguistic terms per attribute, an attribute frequency analysis has been performed on RIPPER-ALL200 and the RF-derived rule systems learned from the ALL200 dataset (Sec. 3.2.2). The presence of each attribute in those systems has been accounted for. Three RF-derived rule systems, using each of the scoring functions presented in section 3.2.1 have been learned. Only the first rule in each rule set (as the ones in Table 3.26) has been utilized, along with the RIPPER-ALL200 system. Altogether, there are 36 rules being analysed. Table 3.29 presents the number of times, in average, an attribute appears in a rule system. Five linguistic terms were assigned to the four most frequent attributes, and three to the rest of attributes present in RIPPER-ALL200, as shown in Table 3.30.

Fuzzy set shape selection Given an attribute x on a domain X , every linguistic term T^i defined on X has a fuzzy set or membership function μ^i associated to it:

$$\mu^i : X \longrightarrow [0, 1] \quad (3.7)$$

such that

$$\mu^i(x) = P(T^i|x), \forall x \in X \quad (3.8)$$

represents the probability for the attribute value x to be named T^i . Figure 3.8 shows a 3-term fuzzy set example.

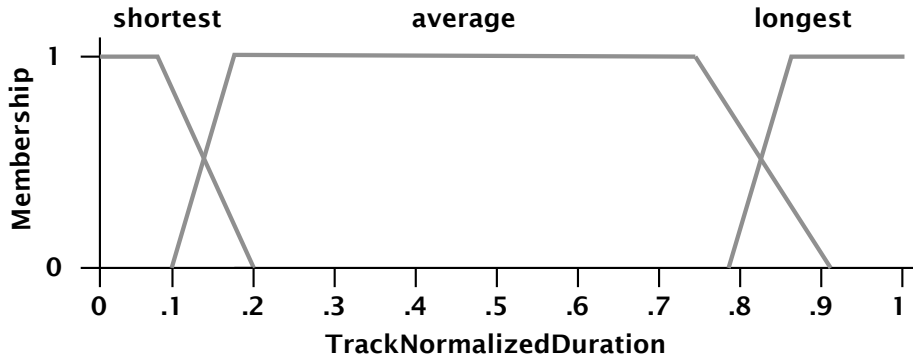


Figure 3.8: Fuzzy set example for attribute *TrackNormalizedDuration*

For efficiency reasons, the shape for a fuzzy set in this work is restricted to be either trapezoidal or triangular, being the latter a special case of the former. Figure 3.9 depicts such fuzzy set shapes.

3.3. A FUZZY APPROACH

Attribute	avg. frequency
TrackOccupationRate	8
LowestPitch	6.25
TrackPolyphonyRate	6
AvgAbsInterval	5.25
TrackNormalizedSyncopation	4
AvgPitch	3.75
TrackNumNotes	3.25
AvgNormalizedAbsInterval	2.5
AvgDuration	2.25
TrackSyncopation	2
TrackNormalizedDuration	2
StdDesviationPitch	2
StdDesviationAbsInterval	2
NormalizedStdDesviationPitch	2
NormalizedStdDesviationDuration	1.75
HighestNormalizedDuration	1.75
NormalizedStdDesviationAbsInterval	1.25
HighestNormalizedPitch	1.25
AvgNormalizedPitch	1.25
LowestNormalizedPitch	1
HighestPitch	1
HighestAbsInterval	1
DistinctIntervals	1
ShortestDuration	0.75
NormalizedDistinctIntervals	0.75
MostRepeatedInterval	0.75
HighestNormalizedAbsInterval	0.75
AvgNormalizedDuration	0.75
LowestNormalizedDuration	0.5
LowestAbsInterval	0.5
StdDesviationDuration	0.25
NormalizedMostRepeatedInterval	0.25
LowestNormalizedAbsInterval	0
LongestDuration	0

Table 3.29: Average attribute presence frequency in crisp rule systems.

CHAPTER 3. MELODY PART SELECTION

Attribute	Linguistic terms
TrackNormalizedDuration	<i>shortest, average, largest</i>
TrackNumNotes	<i>low, average, high</i>
TrackOccupationRate	<i>void, low, average, high, full</i>
TrackPolyphonyRate	<i>none, low, average, high, all</i>
LowestNormalizedPitch	<i>low, average, high</i>
AvgPitch	<i>veryLow, low, average, high, veryHigh</i>
StdDeviationPitch	<i>low, average, high</i>
DistinctIntervals	<i>few, average, alot</i>
NormalizedDistinctIntv.	<i>lowest, average, highest</i>
AvgAbsInterval	<i>veryShort, short, average, large, veryLarge</i>
ShortestDuration	<i>low, average, high</i>
ShortestNormalizedDur.	<i>shortest, average, longest</i>
TrackSyncopation	<i>few, average, alot</i>

Table 3.30: Fuzzy linguistic terms

Fuzzy set parameter setup Each fuzzy set is modeled by four parameters, corresponding to the extreme points of the *core* (or *prototype*) and *support* of a fuzzy set. The *support* is the range of the input domain where $\mu_i(x) > 0$. Given the following attribute domain definition,

$$X = [X_{min}, X_{max}] \quad (3.9)$$

the fuzzy set parameters $(x_L^i, x_{LC}^i, x_{LR}^i, x_R^i)$ (Fig. 3.9) are such that

$$X_{min} \leq x_L^i \leq x_{LC}^i \leq x_{LR}^i \leq x_R^i \leq X_{max} \quad , \quad (3.10)$$

and a trapezoidal fuzzy set is defined as

$$\mu^i(x) = P(T^i|x) = \begin{cases} 0 & \text{if } x < x_L^i \quad \vee \quad x > x_R^i \\ 1 & \text{if } x > x_{LC}^i \quad \wedge \quad x < x_{RC}^i \\ \frac{x - x_L^i}{x_{LC}^i - x_L^i} & \text{if } x \geq x_L^i \quad \wedge \quad x \leq x_{LC}^i \\ \frac{x - x_R^i}{x_{RC}^i - x_R^i} & \text{if } x \geq x_{RC}^i \quad \wedge \quad x \leq x_R^i \end{cases} \quad , \quad (3.11)$$

being a triangular fuzzy set a special case where $x_{LC}^i = x_{RC}^i$.

The definition of such fuzzy sets on numerical attributes usually involves the participation of a human expert who provides domain knowledge for

3.3. A FUZZY APPROACH

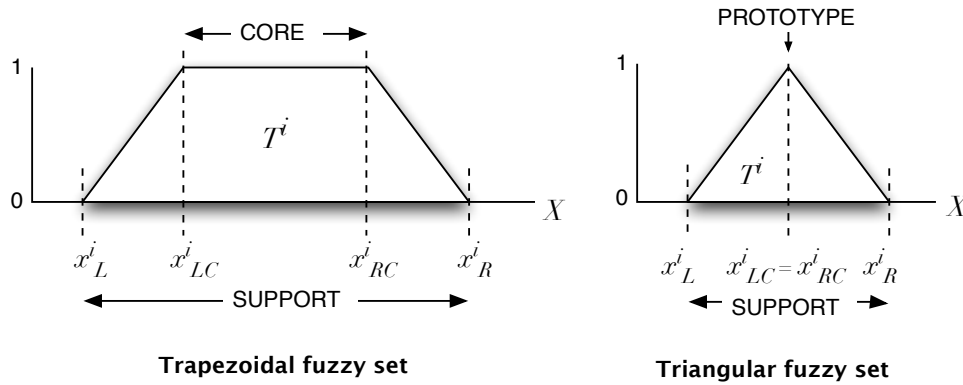


Figure 3.9: Fuzzy set shapes utilized and their parameters

every attribute. The expert usually takes into consideration the distribution of values for an attribute in a reference data collection, as well as any other information available.

Our approach is to replace the human expert by a genetic algorithm to setup fuzzy set parameters. The GA automatically learns the fuzzy set parameters, given the linguistic term definitions for each attribute. Such combination of a fuzzy system with a genetic algorithm is known as a genetic fuzzy system (GFS) (Cordón and Herrera, 1995), described in section 2.6.1.

The output fuzzy variable *IsMelody* has been modeled by *singleton* fuzzy sets. These are fuzzy sets whose support is a single point in X , with a membership function value of one, as represented in Figure 3.10.

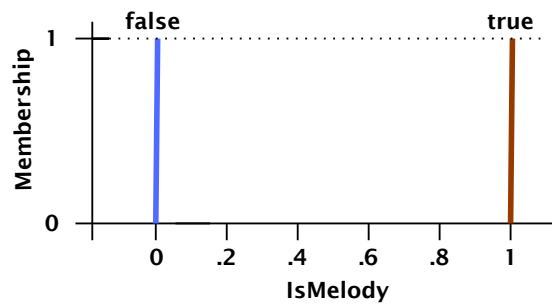


Figure 3.10: Fuzzy output attribute *IsMelody* modeled by singleton fuzzy sets.

3.3.2 Fuzzy set parameter discovery

The objective for the GFS is to optimize the fuzzy set parameters for every attribute in a fuzzy rule system. This optimization process is guided by a fitness function that, given a reference fuzzy rule system, tests potential solutions against a reference dataset. The combination of the reference rule system and the fuzzy set candidates is set to work in order to correctly identify melody and non-melody tracks in the reference dataset. The more tracks are correctly identified, the fitter the proposed solution is.

Fuzzy set representation scheme

An individual's chromosome in the GA encodes all attributes of the fuzzy rule system. The representation scheme used here is similar to the one in (Makrehchi et al., 2003). In order to have an uniform GA representation for every attribute, their domains are normalized to range $[0, 1]$, using the limits explained in section 3.3.1.

The support is considered the most important part of a fuzzy set, while its shape is considered a subjective and application-dependent issue (Lee, 1990). The core is defined here as a function of its support. So, the only parameters that need to be optimized are the support points of each fuzzy set for every attribute. Figure 3.11 shows how an attribute domain, X , is partitioned in overlapping *fuzzy partitions*, each corresponding to a fuzzy set. A fuzzy partition of X is defined as

$$X^i = [x_L^i, x_R^i], X^i \subset X, \quad 1 \leq i \leq M \quad (3.12)$$

where x_L^i and x_R^i are the *left* and *right support points* of the fuzzy set i , respectively; M is the number of fuzzy sets for the attribute. Partitions are defined so that $\forall i, 1 \leq i < M$:

$$x_L^i \leq x_L^{i+1} \wedge x_R^i \leq x_R^{i+1} \quad (3.13)$$

and

$$X = \bigcup_i X^i, \quad (3.14)$$

that is, every input value belong to at least one partition. It follows from (3.13) and (3.14) that $x_L^1 = 0$, so we can drop it from the representation. We also force adjacent partitions to overlap:

$$Z^{i,i+1} = X^i \cap X^{i+1} = [x_L^{i+1}, x_R^i] \neq \emptyset \quad (3.15)$$

3.3. A FUZZY APPROACH

Then the set of parameters to optimize for a given attribute is

$$\Theta = \{x_L^2, x_R^1, x_L^3, x_R^2, \dots, x_L^M, x_R^{M-1}, x_R^M\} \quad (3.16)$$

where they have been set in increasing value order. For the sake of simplicity, let express Θ as

$$\Theta = \{p_1, p_2, \dots, p_{2M-1}\} \quad (3.17)$$

In order to make Θ suitable for crossover and mutation operations, a relative parameter representation scheme is used in the GA. Such scheme is defined as follows

$$\theta = \{p_1, r_2, r_3, \dots, r_{2M-1}\} \quad (3.18)$$

where $r_i = p_i - p_{i-1}$. Clearly, from (3.15) it follows that $r_i > 0$. Figure 3.12 shows this representation scheme. Note that

$$|Z^{i,i+1}| = r_{2i}, \quad 1 \leq i < M \quad (3.19)$$

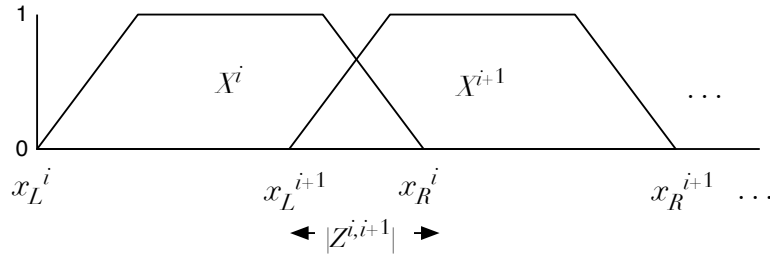


Figure 3.11: Overlapping fuzzy set partitions.

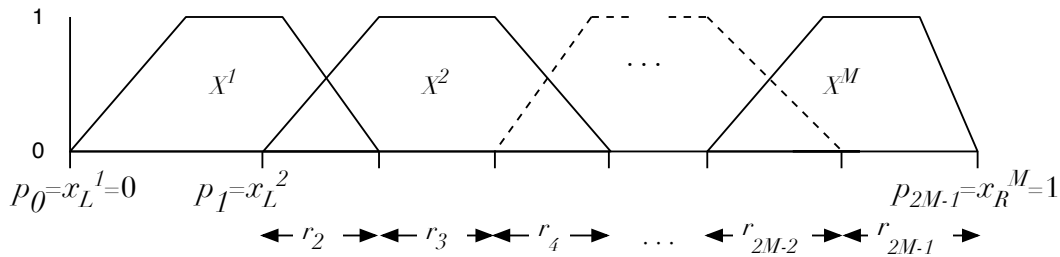


Figure 3.12: Representation scheme of fuzzy sets.

CHAPTER 3. MELODY PART SELECTION

A chromosome of a GA individual (its *genotype*) is made up of one representation scheme per fuzzy attribute,

$$\Omega = \{\theta_1, \theta_2, \dots, \theta_N\} \quad (3.20)$$

where N is the number of attributes in the system.

The chromosome Ω encodes only support points. Once they are known, core points are obtained by setting left and right *boundaries* (Fig. 3.13) for each fuzzy set. These boundaries are restricted to lie inside the overlapping section of their corresponding partition:

$$B_L^i \subset Z^{i-1,i}, \quad 1 < i \leq m \quad (3.21)$$

$$B_R^i \subset Z^{i,i+1}, \quad 1 \leq i < m \quad (3.22)$$

so they are randomly set to be

$$|B_L^i| \leq |Z^{i-1,i}| = r_{2i-2}, \quad 1 < i \leq m \quad (3.23)$$

$$|B_R^i| \leq |Z^{i,i+1}| = r_{2i}, \quad 1 \leq i < m \quad (3.24)$$

Two particular cases are the left boundary of X^1 and the right boundary of X^M .

$$|B_L^1| \leq p_1 = x_L^2, \quad \text{and} \quad (3.25)$$

$$|B_R^M| \leq r_{2M-1} \quad (3.26)$$

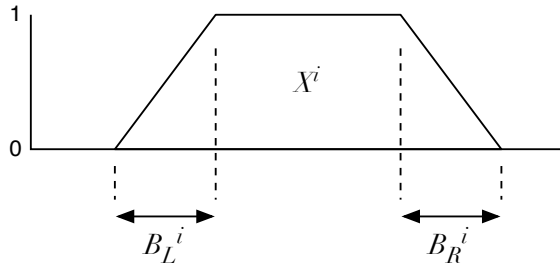


Figure 3.13: Boundaries of a fuzzy set.

These restrictions ensure that the core of a fuzzy set is always equal or greater than zero. The points characterizing all fuzzy sets for a given attribute are set up as

$$x_L^1 = 0 \quad (3.27)$$

$$x_L^i = p_{2i-3}, \quad 1 < i \leq M \quad (3.28)$$

$$x_R^i = p_{2i}, \quad 1 \leq i < M \quad (3.29)$$

$$x_R^M = 1 \quad (3.30)$$

$$x_{LC}^i = x_L^i + |B_L^i|, \quad 1 \leq i \leq M \quad (3.31)$$

$$x_{RC}^i = x_R^i - |B_R^i|, \quad 1 \leq i \leq M \quad (3.32)$$

Remember these points are in the range $[0, 1]$. Before the individual is tested by the fitness function, a de-normalization step is performed for each fuzzy set, using again the limits in Table 3.1.

Fitness function

The fitness function for the GA uses a Fuzzy Inference System (FIS) to test candidate solutions. A FIS, or fuzzy rule-based system, it is a method of inference that uses fuzzy rules, a database (or dictionary) of fuzzy sets, and a fuzzy reasoning mechanism to infer conclusions from input data. The fuzzy set definitions coded in an individual's chromosome are converted to fuzzy sets, which are used as the database of the FIS. The fuzzy rule system discussed below is used by the fitness function to test the FIS database on a reference dataset (see section 3.3.3). The better the performance, the better the individual's score. Several metrics can be used to measure the performance of the FIS. In this work, two different metrics have been tested: 1) number of hits (i.e., number of melody tracks identified by the FIS), and 2) F -measure (harmonic mean of precision and recall of class *IsMelody=true*).

Crisp rule system fuzzification

In order to fuzzify a crisp rule system, antecedents of the form $(x \oslash v)$ ¹¹ are translated into one or more antecedents of the form $(x \text{ IS } T)$, where T is a linguistic term defined for attribute x . The value v partitions the attribute domain in two subsets, and the direction of the inequality guides the selection of the fuzzy terms to be included in fuzzy antecedents.

For a quick depiction of the method see the example in figure 3.14. Suppose four linguistic terms A, B, C and D have been defined for attribute x , along with their respective fuzzy sets. Suppose also that these linguistic terms are ordered in the attribute domain. In this example, $A < B < C < D$.

¹¹ \oslash is any of $>$, \geq , $<$, \leq inequality operators

CHAPTER 3. MELODY PART SELECTION

Suppose the antecedent is $(x > v)$. The value v from the crisp antecedent split x 's domain in two partitions. We obtain v 's membership for each fuzzy set defined on x , then select the linguistic term T_i with higher membership value and convert $(x > v)$ in a disjunction of antecedents of the form $(x \text{ IS } T)$, where T is a linguistic variable equal or greater than T_i . In our example, the fuzzified antecedent equivalent to $(x > v)$ is $((x \text{ IS } C) \vee (x \text{ IS } D))$. It is trivial to define the method for antecedents with other inequality operators.

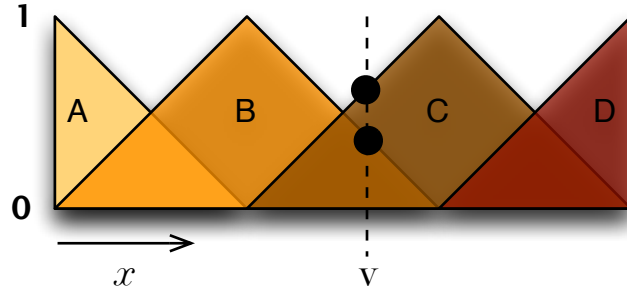


Figure 3.14: fuzzifying a crisp boundary.

The crisp RIPPER-ALL200 rule system (section 3.3.1) has been fuzzified in order to present a proof of concept of the applied methodology. A disjunctive fuzzy rule set is then obtained. Table 3.31 shows fuzzy rules corresponding to the crisp ones shown in Table 3.28. In this method, the fuzzy sets needed to fuzzify the rule system should be the ones provided by GA individuals. The crisp rule fuzzification algorithm should be executed each time an individual's fitness is to be computed. This is a very time consuming task. The time complexity for the algorithm is $O(RCL)$, where R is the number of rules in the crisp rule system, C is the number of conditions per rule, and L is the number of linguistic terms (fuzzy sets) defined for the attribute being used in the condition to fuzzify. This algorithm should be executed $G \times P$ times, being G the number of generations of the GA optimization process, and P the number of individuals in the GA population. With parameters from section 3.3.3, this is ten thousand to one million times per optimization experiment. Therefore, for efficiency reasons, the work presented here uses predefined fuzzy sets, defined by hand. This way, the crisp rule fuzzification algorithm is executed only once at the start of the optimization process. These fuzzy sets are shown in appendix B.

Name	Rule	Name	Rule
FR1	$((\text{AvgPitch IS high}) \vee (\text{AvgPitch IS veryHigh}))$ $\wedge (\text{TrackOccupationRate IS NOT void})$ $\wedge (\text{TrackOccupationRate IS NOT low})$ $\wedge (\text{AvgAbsInterval IS NOT fourth})$ $\wedge (\text{AvgAbsInterval IS NOT high})$ $\wedge (\text{TrackNumNotes IS high})$ $\rightarrow \text{IsMelody IS true}$	FR2	$((\text{AvgPitch IS high}) \vee (\text{AvgPitch IS veryHigh}))$ $\wedge (\text{TrackOccupationRate IS NOT void})$ $\wedge (\text{TrackOccupationRate IS NOT low})$ $\wedge (\text{TrackPolyphonyRate IS NOT average})$ $\wedge (\text{TrackPolyphonyRate IS NOT high})$ $\wedge (\text{TrackPolyphonyRate IS NOT all})$ $\wedge (\text{NormalizedDistinctIntervals IS highest})$ $\rightarrow \text{IsMelody IS true}$
FR3	$((\text{AvgPitch IS high}) \vee (\text{AvgPitch IS veryHigh}))$ $\wedge (\text{TrackNumNotes IS high})$ $\wedge (\text{LowestNormalizedDuration IS shortest})$ $\wedge (\text{ShortestDuration IS NOT low})$ $\wedge (\text{NormalizedDistinctIntervals IS highest})$ $\rightarrow \text{IsMelody IS true}$	FR4	$((\text{AvgPitch IS high}) \vee (\text{AvgPitch IS veryHigh}))$ $\wedge (\text{TrackOccupationRate IS NOT void})$ $\wedge (\text{TrackOccupationRate IS NOT low})$ $\wedge (\text{AvgAbsInterval IS NOT third})$ $\wedge (\text{AvgAbsInterval IS NOT fourth})$ $\wedge (\text{AvgAbsInterval IS NOT high})$ $\wedge (\text{TrackSyncopation IS NOT few})$ $\wedge (\text{StdDeviationPitch IS NOT high})$ $\rightarrow \text{IsMelody IS true}$
FR5	$(\text{AvgAbsInterval IS NOT fourth})$ $\wedge (\text{AvgAbsInterval IS NOT high})$ $\wedge (\text{TrackSyncopation IS alot})$ $\wedge (\text{LowestNormalizedPitch IS NOT low})$ $\wedge (\text{DistinctIntervals IS alot})$ $\wedge (\text{TrackNormalizedDuration IS largest})$ $\rightarrow \text{IsMelody IS true}$	FR6	$(\text{AvgPitch IS NOT veryLow})$ $\wedge (\text{AvgPitch IS NOT low})$ $\wedge (\text{TrackOccupationRate IS NOT void})$ $\wedge (\text{TrackOccupationRate IS NOT low})$ $\wedge (\text{AvgAbsInterval IS NOT third})$ $\wedge (\text{AvgAbsInterval IS NOT fourth})$ $\wedge (\text{AvgAbsInterval IS NOT high})$ $\wedge (\text{TrackPolyphonyRate IS none})$ $\wedge (\text{TrackNumNotes IS NOT low})$ $\rightarrow \text{IsMelody IS true}$

Table 3.31: Fuzzy rule system equivalent to the crisp RIPPER-ALL200 shown in Table 3.28

3.3.3 Experiments and results

Datasets

Datasets used to test the fuzzy rule systems obtained by the GA optimization process are discussed in section 3.2.2. The ALL200 reference dataset has been used to obtain the crisp rule system from which fuzzy rule systems are derived. It is also the dataset used in the GA fitness function to test the performance of potential solutions. The other datasets are used for testing the system. Recall these are multi-genre datasets.

FIS optimization experiment setup

The GFS has six free parameters that provide for different experiment setups. Table 3.32 shows these parameters and the values chosen to build a set of experiments. They have been restricted to at most three different values per parameter. The *selection strategy* parameter values mean to keep the best individual, 10% or 20% of population for next generation, respectively. The outcome of a rule must be greater than the *defuzzification threshold* to consider it was fired.

The restricted number of values per parameter allows the use of an orthogonal array (Sec. 2.7.1) to explore the free parameter space. An OA(18, 6, 3, 2) of strength 2 and 18 runs has been used to setup experiments. As each parameter is supposed to have exactly three values, the F-measure *fitness metric* value is used twice. The FIS optimization experiments resulting from applying the OA to the FIS parameters are summarized in table 3.33.

The FIS engine used in this work is the *jFuzzyLogic* framework (Cingolani, 2008). The GA optimization engine has been implemented using *JGAP* (Meffert et al., 2008). Please see appendix A for a brief introduction to these frameworks.

Experiment parameter	Values
GA population size	100, 500, 1000
GA no. of generations	100, 500, 1000
GA mutation ratio	none, 0.05, 0.1
GA selection strategy	Best one, Best 10%, Best 20%
GA fitness metric	Hit count, F-measure
Defuzzification threshold	0.5, 0.6, 0.7

Table 3.32: FIS optimization setup parameters

3.3. A FUZZY APPROACH

FISO	Population	Generations	Fitness	Mutation	Selection	Defuzz. thrs.
0	100	100	Hits	–	Best	0.5
1	500	500	F-measure	.1	Best 10%	0.6
2	1000	1000	F-measure	.05	Best 20%	0.7
3	100	100	F-measure	.05	Best 10%	0.7
4	500	500	F-measure	–	Best 20%	0.5
5	1000	1000	Hits	.1	Best	0.6
6	100	500	Hits	.05	Best 20%	0.6
7	500	1000	F-measure	–	Best	0.7
8	1000	100	F-measure	.1	Best 10%	0.5
9	100	1000	F-measure	–	Best 10%	0.6
10	500	100	Hits	.1	Best 20%	0.7
11	1000	500	F-measure	.05	Best	0.5
12	100	500	F-measure	.1	Best	0.7
13	500	1000	Hits	.05	Best 10%	0.5
14	1000	100	F-measure	–	Best 20%	0.6
15	100	1000	F-measure	.1	Best 20%	0.5
16	500	100	F-measure	.05	Best	0.6
17	1000	500	Hits	–	Best 10%	0.7

Table 3.33: FIS optimization (FISO) experiments

FIS optimization results

Table 3.34 shows the performance on melody track categorization on the ALL200 dataset. Both evolved FIS and RIPPER-ALL200 crisp rule system performances are shown. Average results from the eighteen optimization experiments are presented. Precision, recall and F-measure are computed for the class 'IsMelody'. Also, the best performance of evolved FIS is presented. This has been obtained using the parameter combination number 1 in table 3.33. Note that the best evolved FIS performance is very close to that from the crisp rule system. The definition of fuzzy sets for the best evolved FIS is presented in Figure 3.15. Note that overlapped fuzzy set cores (e.g. fuzzy sets *low* and *average* for attribute *TrackNumNotes*) are valid. It means that, for any input value x in the overlapping range, both linguistic terms would have $\mu^i(x) = 1$.

Figure 3.16 presents the fitness evolution for each experiment. Beyond three hundred generations there is little improvement in fitness with both measures. In average, FIS evolved with a F -measure based fitness function performed better than using hit-based fitness. There are some experiments that reached the 0.82 best F -measure mark on the ALL200 dataset. Notably, experiment 3 needed only 100 generations to reach this score (against 500 generations needed by experiment 1, chosen as the best in Table 3.34).

CHAPTER 3. MELODY PART SELECTION

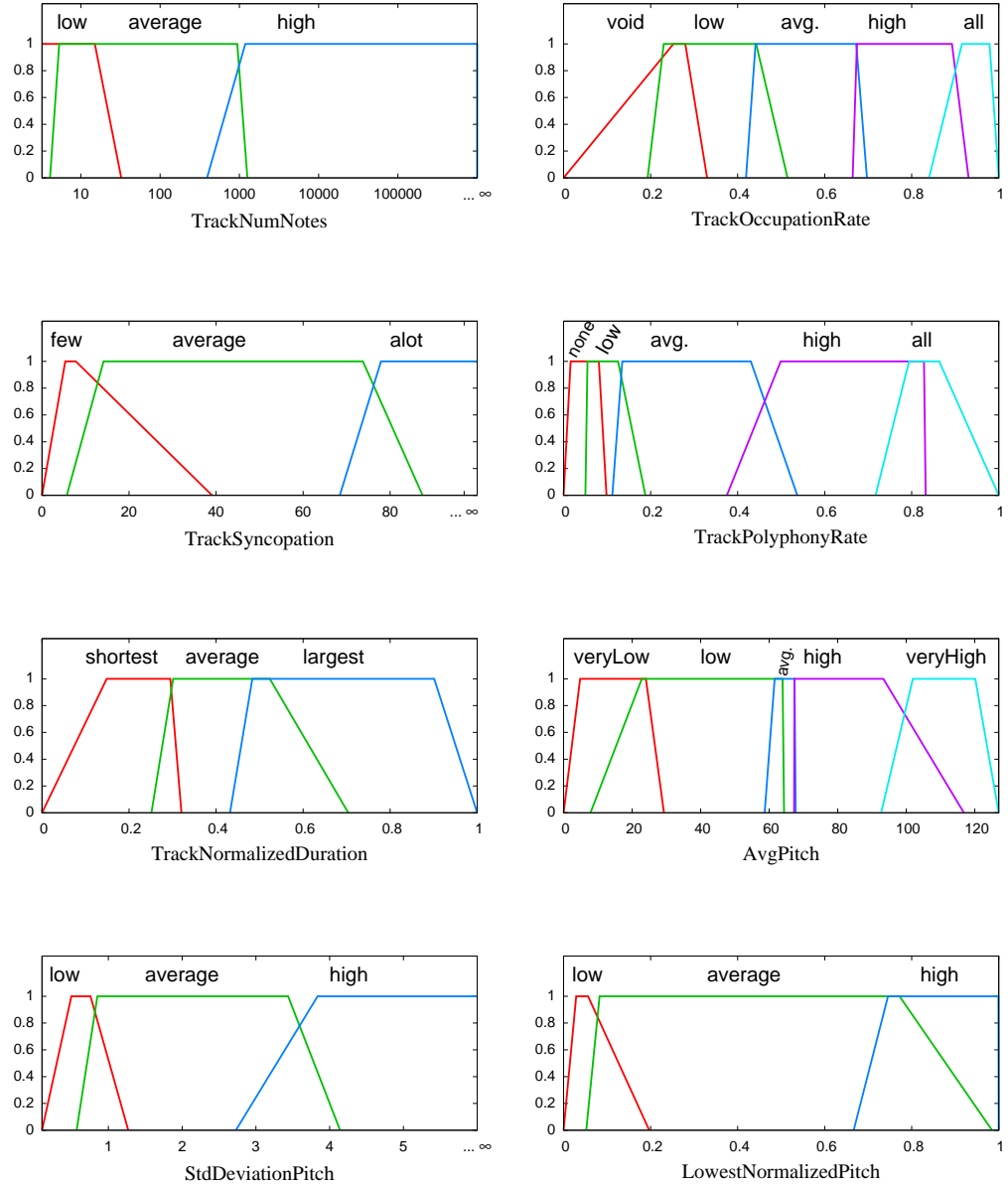


Figure 3.15: Fuzzy set definitions from the best evolved FIS (FISO 1).

3.3. A FUZZY APPROACH

Rule sys.	Precision	Recall	F	Error rate
<i>crisp</i>	0.89	0.87	0.88	0.05
Best FIS	0.81	0.83	0.82	0.06
Avg. FIS	0.80 (.03)	0.77 (.09)	0.78 (.05)	0.08 (.01)

Table 3.34: Best and average performance of evolved FIS vs. crisp RIPPER-ALL200 rule system on the ALL200 dataset. Figures in parenthesis are standard deviations.

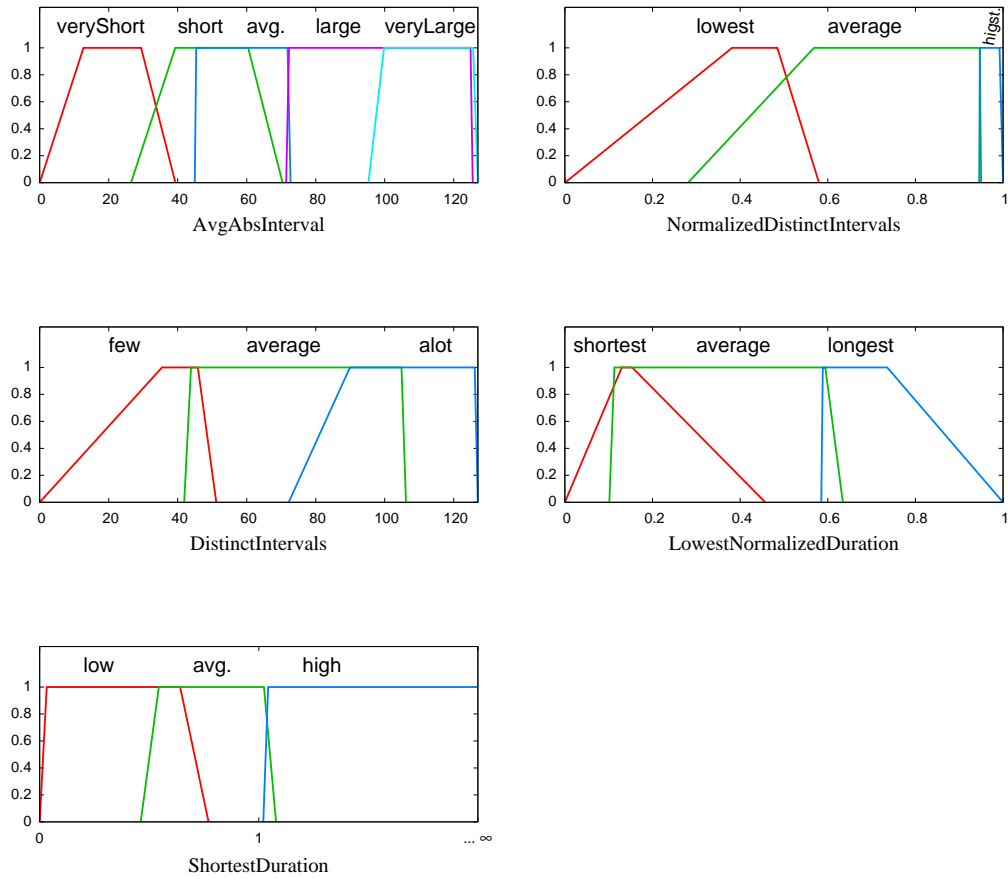


Figure 3.15: Fuzzy set definitions from the best evolved FIS (cont.).

CHAPTER 3. MELODY PART SELECTION

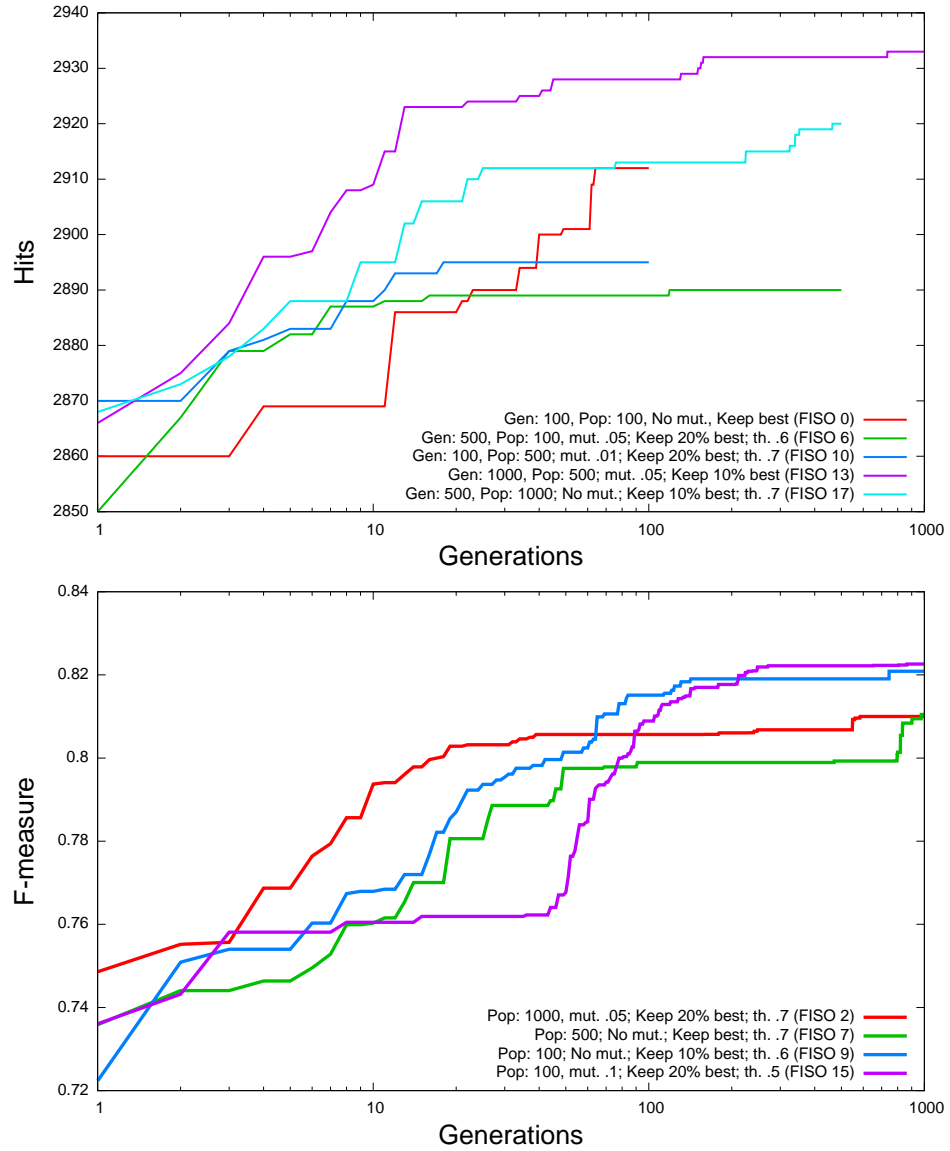


Figure 3.16: (top) GA fitness by hits. (bottom) GA fitness by F -measure (1000 generations).

3.3. A FUZZY APPROACH

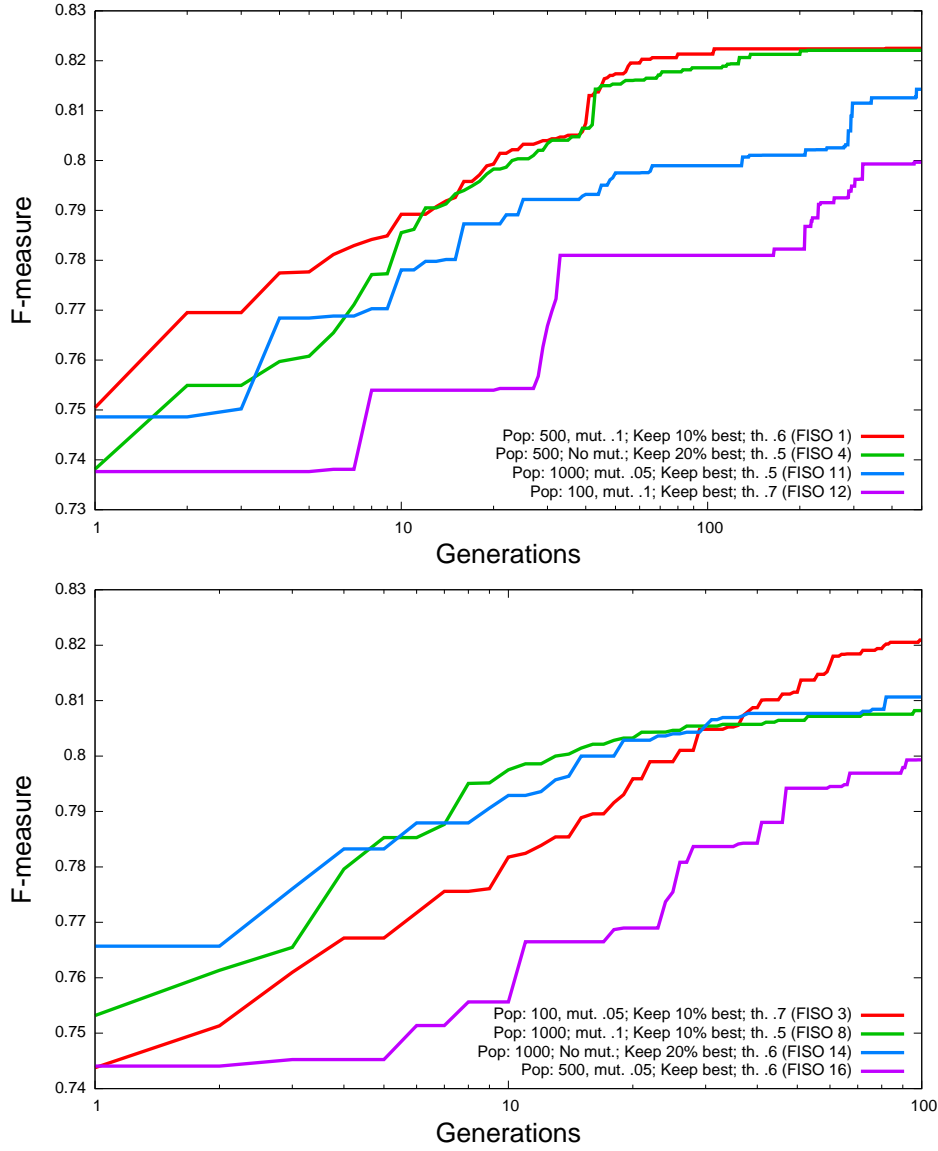


Figure 3.16: (cont.) GA fitness by F -measure. (top) 500 generations and (bottom) 100 generations experiments.

CHAPTER 3. MELODY PART SELECTION

Results on test datasets.

Table 3.35 presents a summary of results for the best solution from each FIS optimization experiments on test datasets. Relying on the F -measure as the measure of performance, rather than error rate, no single combination of parameters performed better than others in more than one dataset. However, results are very close between several parameter combinations on the same test dataset. Analysing optimization process parameters one at a time, it seems that setting the population to a value larger than 500 does not make any significant improvement. Neither does selecting 20% of the population for survival to the next generation. This is in accordance with conclusions drawn from inspecting fitness curves in figure 3.16.

For comparison purposes, table 3.36 presents results from applying both the crisp rule system and the best evolved FIS to test datasets. For the FIS system, a track is classified as a melody track if it fires at least one rule with probability greater than 0.5. Otherwise, the track is classified as non-melody. As the results show, the fuzzified rule system performances are lower than the original crisp rule system performance, except maybe for the AJP dataset. This is, by the way, the most realistic scenario among the considered databases. The biggest differences are observed in the smaller data sets, i.e. RWC-P and RWC-G, with a limited set of examples (e.g. RWC-G contains only 44 melody examples). Analyzing the errors made by the fuzzy rule systems, it follows that most errors are false positives, that is, some non-melody tracks are classified as melody tracks. It is important to remind here that the goal of the fuzzification process is not to improve classification accuracy, but to obtain a human-readable comprehensible characterization of melodies within MIDI tracks. However, a comparable performance would be desirable, which was not the case, in general. One of the causes for this could be the use of predefined fuzzy sets (section B) for crisp rule fuzzification, leading to a predefined, fixed fuzzy rule system in the fitness function.

Some attempts were made trying to improve fuzzy rule systems. The output fuzzy attribute *IsMelody* membership functions were set to non-singleton shapes, like the example in Figure 3.17. This way the output of the system is either `true` or `false` with a probability between 0.5 and 1. With this setup, where the crossover points of both fuzzy sets is 0.5, results were identical to those obtained using the singleton-based setup. Changing the crossover point produced only worse results. In particular, results on the recall metric got dramatically worse when this value was set slightly above 0.5.

Also, the `AvgAbsInterval` firing in rules has been investigated. Only the term *veryShort* gets membership probability greater than zero for every

3.3. A FUZZY APPROACH

	<i>ALL200</i>		<i>LARGE</i>		<i>RWC-G</i>		<i>RWC-P</i>		<i>AJP</i>	
	<i>F</i>	Err.	<i>F</i>	Err.	<i>F</i>	Err.	<i>F</i>	Err.	<i>F</i>	Err.
0	0.75	0.08	0.69	0.09	0.41	0.15	0.58	0.08	0.83	0.06
1	0.82	0.06	0.72	0.09	0.43	0.16	0.57	0.09	0.85	0.06
2	0.78	0.08	0.69	0.10	0.44	0.15	0.59	0.09	0.85	0.06
3	0.78	0.07	0.69	0.09	0.43	0.15	0.59	0.08	0.84	0.06
4	0.82	0.07	0.72	0.09	0.44	0.16	0.56	0.09	0.86	0.06
5	0.79	0.07	0.71	0.09	0.40	0.15	0.56	0.08	0.84	0.06
6	0.75	0.08	0.71	0.08	0.37	0.15	0.48	0.08	0.84	0.06
7	0.81	0.07	0.72	0.09	0.45	0.15	0.60	0.09	0.86	0.06
8	0.77	0.09	0.68	0.11	0.40	0.18	0.55	0.10	0.80	0.08
9	0.81	0.07	0.72	0.09	0.45	0.16	0.56	0.09	0.85	0.06
10	0.69	0.09	0.71	0.08	0.28	0.15	0.51	0.07	0.82	0.07
11	0.81	0.07	0.71	0.09	0.45	0.15	0.58	0.08	0.85	0.06
12	0.80	0.07	0.72	0.09	0.53	0.14	0.59	0.09	0.86	0.06
13	0.80	0.07	0.72	0.08	0.37	0.16	0.50	0.09	0.84	0.06
14	0.79	0.07	0.70	0.09	0.41	0.15	0.50	0.10	0.86	0.06
15	0.81	0.07	0.71	0.10	0.44	0.16	0.55	0.10	0.84	0.07
16	0.80	0.07	0.71	0.10	0.49	0.16	0.59	0.09	0.86	0.05
17	0.63	0.11	0.51	0.12	0.26	0.15	0.52	0.08	0.67	0.11

Table 3.35: Summary of evolved FIS results on test datasets.

Dataset	Precision	Recall	F	Error rate
LARGE (<i>crisp</i>)	0.79	0.80	0.80	0.06
LARGE (<i>fuzzy</i>)	0.75	0.68	0.72	0.08
RWC-G (<i>crisp</i>)	0.54	0.77	0.64	0.13
RWC-G (<i>fuzzy</i>)	0.49	0.57	0.53	0.14
RWC-P (<i>crisp</i>)	0.95	0.80	0.87	0.02
RWC-P (<i>fuzzy</i>)	0.52	0.72	0.60	0.09
AJP (<i>crisp</i>)	0.88	0.89	0.88	0.05
AJP (<i>fuzzy</i>)	0.90	0.83	0.86	0.05

Table 3.36: Melody track categorization results.

CHAPTER 3. MELODY PART SELECTION

test sample. For this attribute, values in the ALL200 dataset are in the range $[0, 19]$, so this was somewhat expected. None of the FISO experiments achieved to capture the distribution of fitness test data on this attribute (see, for example, Fig. 3.15). As the attribute domain can not be restricted to the values found in a particular dataset, a straight forward approach is to set some attributes to have predefined fuzzy sets, so they parameters are not part of the FIS chromosome setup. An experiment was performed using the *FISO 1* evolved fuzzy sets, except for the attribute *AvgAbsInterval* whose fuzzy sets were fixed *ad hoc* to those shown in Figure 3.18. As shown in Table 3.37, results obtained this way are better on one dataset (RWC-P), while worse or comparable on the others. Anyway, this suggests that incorporating expert knowledge on some particularly ‘hard to evolve’ attributes makes a difference. Further investigation would be needed in order to know under what circumstances predefined fuzzy sets give better results than evolved ones.

An alternative method of fuzzy set fitness evaluation would be desirable to improve results in such cases where the domain is oversized. One approach would be to measure how well the fuzzy sets adapt to the distribution of values in a dataset. This can be achieved using information theory metrics, as in (Makrehchi et al., 2003). However, it would be desirable to measure this fitness on several data distributions, to prevent overfitting a particular dataset.

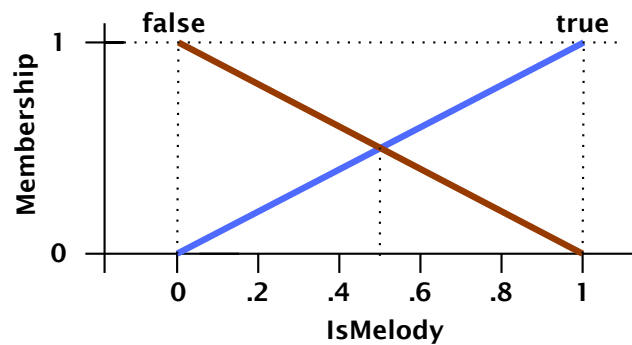


Figure 3.17: Alternative setup for the fuzzy output variable *IsMelody*.

3.3. A FUZZY APPROACH

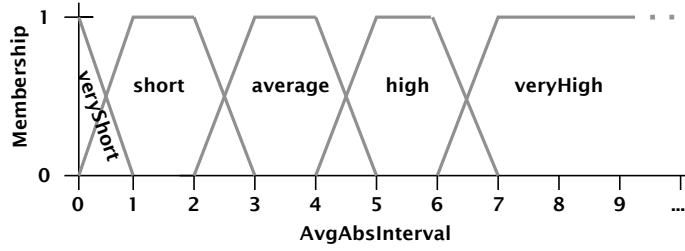


Figure 3.18: Predefined fuzzy sets for attribute *AvgAbsInterval*.

Dataset	Precision	Recall	F	Error rate
LARGE (<i>evolved</i>)	0.75	0.68	0.72	0.08
LARGE (<i>fixed</i>)	0.82	0.65	0.72	0.08
RWC-G (<i>evolved</i>)	0.49	0.57	0.53	0.14
RWC-G (<i>fixed</i>)	0.44	0.40	0.42	0.16
RWC-P (<i>evolved</i>)	0.52	0.72	0.60	0.09
RWC-P (<i>fixed</i>)	0.78	0.63	0.70	0.05
AJP (<i>evolved</i>)	0.90	0.83	0.86	0.05
AJP (<i>fixed</i>)	0.91	0.66	0.77	0.08

Table 3.37: Melody track categorization. Evolved vs. fixed *AvgAbsInterval* attribute fuzzy sets.

3.3.4 Comparison of crisp and fuzzy systems on some examples

This section discuss several melody characterization examples. The example excerpts are shown in Tables 3.38 and 3.39. The words 'Crisp' and 'Fuzzy' under the music systems indicate which rules from the crisp and fuzzy systems were fired, respectively. The fuzzy rule system used for comparison was the best evolved FIS on the ALL200 dataset (Fig. 3.15), using the rules in Table 3.31. For the crisp rules, please see Table 3.28.

The first two tracks are melody tracks that were correctly identified by the fuzzy rule system. The fuzzy rules fired by these examples are shown in Table 3.40 for convenience. Note how similar these two rules are. In terms of average pitch, track occupation rate, and average pitch interval they are almost identical. Both of them get activated by tracks with rather high average pitch, small average pitch interval values, and with a moderate to large portion of the track occupied by notes. In addition, rule FR4 expects tracks with some amount of syncopation, and moderate pitch standard

CHAPTER 3. MELODY PART SELECTION

deviation values. Finally, FR6 would see its activation level increased by monophonic tracks with a significant amount of notes in it.

Crisp rules failed at characterizing the first example. This first track almost fulfills rule *R2*, except that it has not the largest pitch interval variety (its *NormalizedDistinctIntervals* value is 0.85), as the last condition of the rule imposes. The next two tracks in Table 3.38 are non-melody tracks correctly categorized by both rule systems (no track fired any rule). The last two examples (Table 3.39) are tracks where both rule systems disagree. The melody track from *Satin Doll* is unusual in the sense that it is supposed to be played by a vibraphone (a polyphonic instrument), has one chorus of improvisation, and the melody reprise (which is the part shown in the example) is played in a polyphonic *closed chord* style. The last example is a piano accompaniment part, played in *arpeggiato* style, that the fuzzy rules incorrectly identified as a melody track. This track almost fired crisp rule *R6*, except for the second condition of the rule, because its *TrackPolyphonyRate* = 0.097. This is a clear example of why a fuzzy version of a crisp rule fires while the crisp rule doesn't. The value is accepted by the fuzzy rule as the linguistic term *none* for the *TrackPolyphonyRate* attribute. This is because it lies into the support of the fuzzy set corresponding to that term.

3.4 Conclusions on melody part selection

The method proposed here identifies the voice containing the melody in a multitrack digital score. It has been applied to standard MIDI files in which music is stored in several tracks, so the system determines whether each track is a melodic line or not. The track with the highest probability among the melodic tracks is eventually labeled as the one containing the melody of that song.

The decisions are taken by a pattern recognition algorithm based on statistical descriptors (pitches, intervals, durations, and lengths), extracted from each track of the target file. The classifier used for the experiments was a decision tree ensemble classifier named random forest. It was trained using MIDI tracks with the melody track previously labeled by a human expert.

The experiments yielded promising results using databases from different music genres, like jazz, classical, and popular music. Unfortunately, the results could not be compared to other systems because of the lack of similar works and benchmark datasets.

The results show that enough training data of each genre are needed in order to successfully characterize the melody track, due to the specificities of melody and accompaniment in each genre. Classical music is particularly

3.4. CONCLUSIONS ON MELODY PART SELECTION



True positive examples
<i>Air In F, Watermusic, Handel (Baroque)</i> Melody 
Crisp: – Fuzzy: FR6
<i>There Is No Greater Love, I. Jones (pre-Bop Jazz)</i> Melody 
Crisp: R2, R5 Fuzzy: FR4, FR6
True negative examples
<i>Air In F, Watermusic, Handel (Baroque)</i> Bass 
Crisp: – Fuzzy: –
<i>There Is No Greater Love, I. Jones (pre-Bop Jazz)</i> Piano (accompaniment) 
Crisp: – Fuzzy: –

Table 3.38: Track samples correctly classified by the fuzzy system.

CHAPTER 3. MELODY PART SELECTION

False negative example
<i>Satin Doll</i> , D. Ellington (<i>pre-Bop Jazz</i>)
Melody

Crisp: R2
Fuzzy: –
False positive example
<i>Sonata no. 3 K545, 2nd Mov.</i> , W.A. Mozart (<i>Classicism</i>)
Piano (accompaniment)

Crisp: –
Fuzzy: FR6

Table 3.39: Track samples incorrectly classified by the fuzzy system.

FR4	FR6
((AvgPitch IS <i>high</i>)	(AvgPitch IS NOT <i>veryLow</i>)
∨ (AvgPitch IS <i>veryHigh</i>))	∧ (AvgPitch IS NOT <i>low</i>)
∧ (TrackOccupationRate IS NOT <i>void</i>)	∧ (TrackOccupationRate IS NOT <i>void</i>)
∧ (TrackOccupationRate IS NOT <i>low</i>)	∧ (TrackOccupationRate IS NOT <i>low</i>)
∧ (AvgAbsInterval IS NOT <i>third</i>)	∧ (AvgAbsInterval IS NOT <i>third</i>)
∧ (AvgAbsInterval IS NOT <i>fourth</i>)	∧ (AvgAbsInterval IS NOT <i>fourth</i>)
∧ (AvgAbsInterval IS NOT <i>high</i>)	∧ (AvgAbsInterval IS NOT <i>high</i>)
∧ (TrackSyncopation IS NOT <i>few</i>)	∧ (TrackPolyphonyRate IS <i>none</i>)
∧ (StdDeviationPitch IS NOT <i>high</i>)	∧ (TrackNumNotes IS NOT <i>low</i>)
→ IsMelody IS true	→ IsMelody IS true

Table 3.40: Fuzzy rules fired by examples in Table 3.38 and 3.39.

3.5. CONTRIBUTIONS IN THIS CHAPTER

hard for this task, because the melody is not confined to a single track in some files. There, melody moves from one track to another, as different instruments take the lead role. To overcome this problem, more sophisticated schemes oriented to melodic segmentation are needed.

Being able to automatically obtain melody characterization rules that are easily understandable by humans could be of interest for musicologists and would help building better tools for searching and indexing symbolically encoded music. The extraction of human-readable rules from the trees in the random forest that help characterize melody tracks has been another topic of research. Several rule systems, including fuzzy rule systems, have been obtained (Ponce de León et al., 2007; Ponce de León et al., 2008). A method to obtain reduced rule systems from previously learnt random forests that characterize melody tracks has been exposed. Such rule systems perform comparably to the original decision tree ensembles. The study of these rules can lead also to the description of other track categories, such as *solo* or *chorus* tracks.

In order to obtain a better description (in term of readability) of the rule models obtained, they are automatically transformed into fuzzy rule systems by applying a genetic algorithm to generate the membership functions for the rule attributes. The classification accuracy of the resulting fuzzy rule system is lower than the original crisp rule system, but comprehensibility of the rules is improved. Most errors detected were false positives. This is probably due to the fact that all rules are 'melody rules', i.e., there are not rules for characterizing non-melody tracks, so the fuzzy inference system answers a track is a melody if at least one rule activates with probability 0.5 or greater. Adding rules to characterize non-melody tracks could help identify false positives correctly.

3.5 Contributions in this chapter

As far as the author knows, the first work to address the problem of melody part selection in MIDI files is (Tang et al., 2000). The authors apply heuristics on some statistical properties of MIDI tracks to rank or classify them as melody tracks. The work presented in this chapter is the first in the field, to the best of our knowledge, to address the problem of melody part selection from a machine learning point of view. Some of the reviewers of the first papers (Rizo et al., 2006c,d) published on the subject also agreed on that.

CHAPTER 3. MELODY PART SELECTION

The following are the main contributions of this research:

- A supervised learning system for discriminating between melody and non-melody MIDI tracks. The system is based on global statistical descriptions of track content, including descriptors that relate such content to what is found in other tracks.
- A collection of MIDI file corpora, differentiated by genre (classical music, jazz, and karaoke-like music), and devised for training and testing melody part selection methods. Melody tracks are tagged by adding a special tag to the track name inside the original MIDI file.
- A study on genre specificity for melody part selection problems, which shows that using a model built exclusively on the genre of the music piece under test is a good and simple approach, at least with genres used in this work.
- Human-readable (fuzzy) rule systems for melody track categorization. As far as the author knows, these rule systems are the first descriptions of the melody concept automatically extracted from symbolic music examples, at least inside the MIR community. Moreover, such description is human-friendly, at least in its fuzzy form, so it can serve as a basis for further musicological analysis of the melody concept.
- A greedy crisp rule ranking method for two-class problems.
- A genetic fuzzy system for learning membership functions from melody-tagged data.

As helper applications for this research, a graphical user interface for melody part selection in MIDI files, and a MIDI feature extraction plugin for the WEKA toolkit were developed. The GUI application is trainable, and can play whole songs or any combination of tracks to verify the solution proposed by the underlying model. The WEKA plugin allows to select MIDI file folders as input data, without the need to first generate ARFF files.

4

Music genre recognition

*“A name doesn’t make the music.
It’s just called that to differentiate it
from other types of music.”*
Art Blakey

The music genre recognition task has been approached by applying the general feature extraction methodology presented in sec. 1.3, which has been used thoroughly in this research about music genre recognition. The framework devised to apply such methodology comprises two main techniques:

The sliding window The extraction of useful information from a music fragment content will go thru the use of a sliding window. This window moves over the music sequence and takes snapshots of its content of a given length and at given sampling intervals, with both parameters usually expressed in terms of beats or bars. This is pretty much the same concept as the *sampling unit* concept used in statistics, which refers to the set of elements considered for selection in some stage of sampling, except for the fact that two windows can overlap, i. e., the end beats of a window can be the first ones of the next, given that the sampling interval is smaller than the window length.

Global statistical information A set of low-level or mid-level statistical descriptors are computed from the content of a given window. This way, an entire music sequence is described by one or more samples as vectors of numeric data. If the music sequence has been assigned any tag to describe its genre, all samples derived from this particular sequence will be assigned this same tag. This is called *global statistics information* to differentiate it from other approaches that use *local statistics information* based, for example, on *n*-grams (Doraisamy, 2004; Pérez-Sancho, 2009a). In these works statistical information is extracted from short sequences of musical events (the *n*-grams),

CHAPTER 4. MUSIC GENRE RECOGNITION

that usually elapses a few beats, as opposed to the present approach, where statistical information summarizes the content of several bars long music segments, typically made up of larger and more complex sequences of musical events. This kind of statistical description of musical content is sometimes referred to as *shallow structure description* (Pickens, 2001). It is similar to histogram-based descriptions, like the one found in (Toiviainen and Eerola, 2001), that tries to model the distribution of musical events in a musical fragment. Computing the minimum, maximum, mean and standard deviation from the distribution of musical features like pitches, durations, intervals, and non-diatonic notes we reduce the number of features needed (each histogram may be made up of tens of features), while assuming the distributions for the mentioned features to be normal within a melodic fragment. Other authors have also used some of the descriptors presented here to classify music (Blackburn, 2000; McKay, 2010).

Therefore the focus in these research will be put on

- assessing that the feature extraction methodology proposed above is valid for music genre classification purposes, that is, the hypothesis that melodies from the same genre are closer to each other, in the description space, than melodies from different genres, hold.
- build a set of music content statistical descriptors based on several properties of a music stream.

4.1 Feature space exploration by self-organising maps

Self-organising maps, or SOM, for short (c.f. 2.1.2), are an interesting tool for data exploration and visualization purposes. Other authors (Dopler et al., 2008; Lidy and Rauber, 2008; Pampalk, 2006), have used SOMs successfully for classifying, visualizing, and browsing music genres. They have been used here to explore and visualize the JvC1 dataset, using as data representation for melody tracks some of the statistical features described in section 4.3.1. A premise for the work discussed here is that monophonic melody tracks in JvC1 MIDI files are properly identified *a priori*.

Once a SOM is tagged, it can be used to classify new samples, and are particularly helpful to visualize the result of this classification process, specially when the samples are consecutive fragments of a sequence, as is the

4.1. FEATURE SPACE EXPLORATION BY SOM

case with melody track samples in this research. I take advantage of this capabilities to reinforce some results. As a side effect, several experimental tests are contributed as a proof of concept for the idea that a statistical description of a monophonic melody fragment contains enough information to recognize, in most cases, its musical genre in absence of timbral information.

4.1.1 Methodology

Some previous surveys revealed that melodies from a same musical genre share some common features that make it possible for a experienced listener to assign a musical genre to them (c.f. 4.2). In that study, melody fragments from 3 to 16 bars-long were proposed to the listeners. In order to have more restricted data, fragments of 8 bars length (32 beats) are used here (enough to get a good sense of the melodic phrase in the context of a 4/4 signature). For this, each melody sequence has been cut into fragments of such duration. The description model proposed in section 4.3.1 is used to represent these melody segments. SOMs are first unsupervisedly trained with this data representation, and then calibrated (tagged) with labeled training samples. From that point, they can be used to visualize the structure of the data, or to classify new samples. Fig. 4.1 presents an illustration of the methodology. The melody is a fragment of the jazz piece "Dexterity".

The SOM provides several visualization capabilities (c.f. 2.1.2). The main method for visualizing the structure of a SOM is the U-map representation, as in Fig 4.2-a. The map in this example has been calibrated (i.e., tagged) after training, with symbols to represent different classes. So, the left area of the map clearly corresponds to samples of class 'O', while the right area corresponds to class 'X'. Some descriptor values for the weight vectors correlate better than others with the label distribution in the map. It is reasonable to consider that these descriptors contribute more to achieve a good separation between classes. The SOMPAK toolkit used in this work allows to visualize this weight vector 'planes', as shown in Fig. 4.2-b and 4.2-c.

4.1.2 Visualization of random vs. real melodies

The first exploration experiment conceived has been to train a SOM with random (i.e., fake) and real melodies. 400 random, 8-bar-long monophonic melodies were generated using different proportions of notes and silences. 430 real melody samples were chosen among jazz songs in the JvC1 corpus. Two different bidimensional SOM geometries have been tested: 16×8 and 30×12 maps. An hexagonal geometry for unit connections and a

CHAPTER 4. MUSIC GENRE RECOGNITION

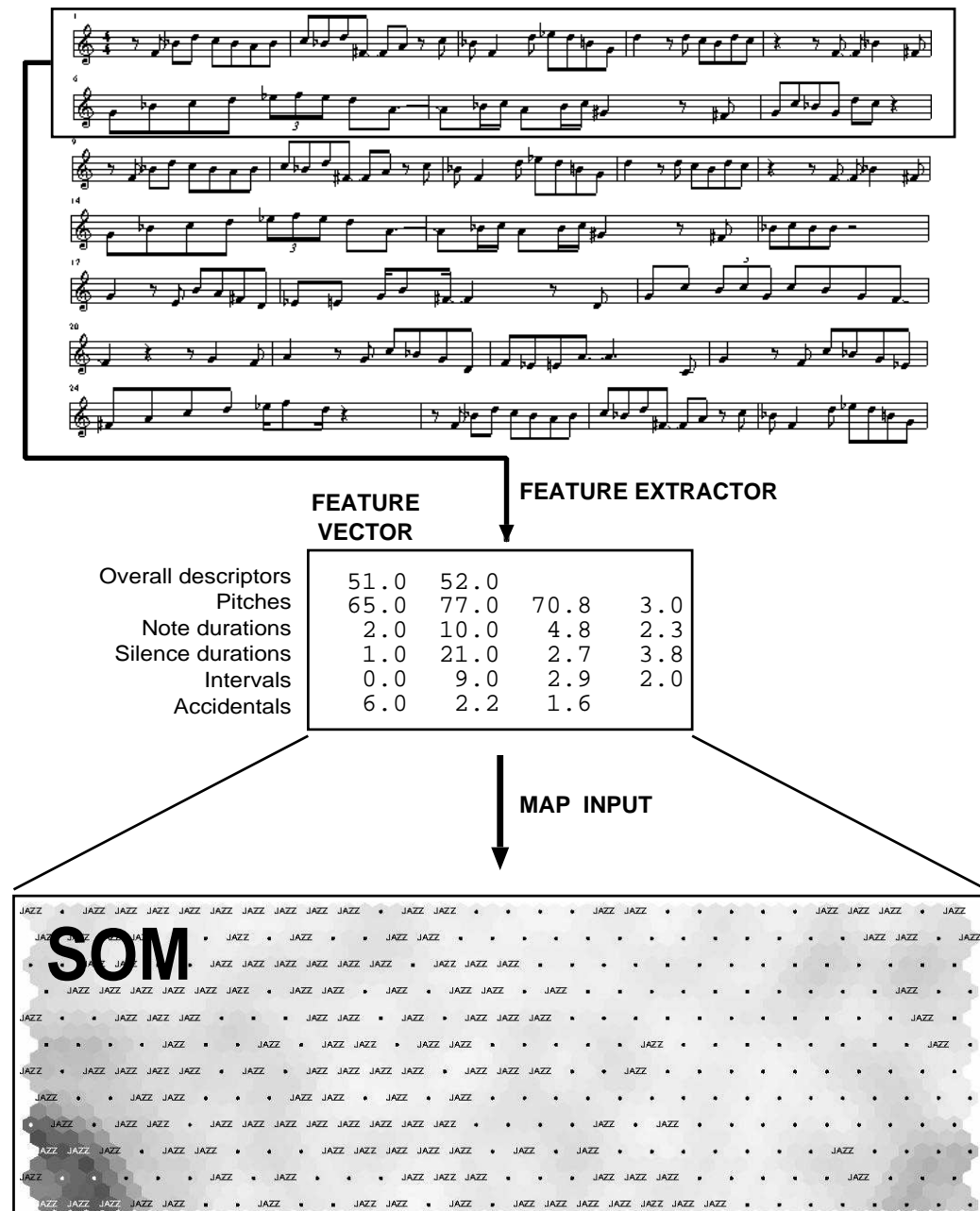


Figure 4.1: Depiction of the exploration methodology: musical descriptors are computed from a segment 8-bar wide and provided to the SOM.

4.1. FEATURE SPACE EXPLORATION BY SOM

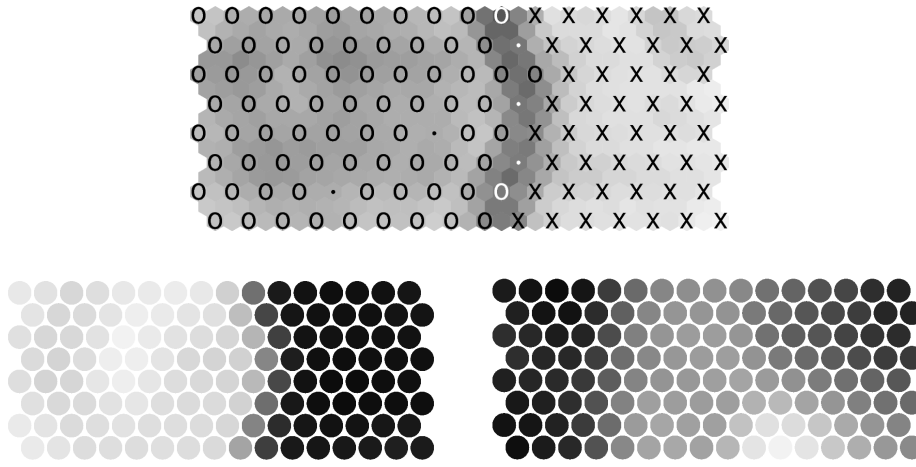


Figure 4.2: Contribution to classification: (top) calibrated map ('X' and 'O' are labels for two classes); (bottom:left) weight space plane for a feature that correlates with the classes (avg. interval); (bottom:right) plane for a feature that does not correlate (num. of notes).

bubble neighbourhood for training have been selected. The value for this neighbourhood parameter is constant for all the units in it and decreases as a function of time.

The training has been done in two phases, a coarse one of 1000 iterations with wide neighbourhoods (12 units) and a high learning rate (0.1) and then a fine one of 10,000 iterations with smaller neighbourhood ratio (4 units) and learning rate (0.05). These training parameters have been applied to the rest of experiments with little variations. Table 4.1 shows these parameters. The metrics used to compute distance between samples is the Euclidean distance. The 'Q error' gives us an idea of how well the SOM has adjusted its weight vectors to training data. Several training runs are performed, with random initial weight vector values, and the map with lower Q error is selected.

Figure 4.3 shows the Sammon projection of the trained SOM. Nodes correspond to map units. Note that there exists a clear gap between two zones in the projection. The small cluster on the right in the Sammon projection corresponds to the real melodies and that of the left to the random melodies. This map has been obtained with random melodies with no pitch range restriction, i.e., valid pitches are in the range $[0, 127]$. However, as stated before in section 4.1.1, real melodies use to have a much smaller pitch range. So, in this attempt, the pitch range (described by the minimum and maximum pitch descriptors) is responsible for the clear separation of

CHAPTER 4. MUSIC GENRE RECOGNITION

Geometry	16x8, hexa., bubble	
Trials	20	
	Phase I	Phase II
Iterations	1 000	10 000
$\alpha(0)$	0.1	0.05
$r(0)$	12	4
Q. Error	10.5	

Table 4.1: Training parameters for Random vs. Jazz experiment.

both type of melodies. Also, the figure shows evidence that real melodies are somewhat confined to a much smaller region in the feature space than random melodies.¹

In figures 4.3 and 4.5 the Sammon projection and the SOM map are displayed after training the SOM with restricted pitch range random melodies and real melodies. Nodes labeled with numbers correspond to different distributions of notes in random melodies. There still exists a clear gap between two zones in the map. The small cluster on the upper part of the Sammon projection corresponds to real melodies, while the bottom part corresponds to random melodies. On the map in Figure 4.5 the same gap can be observed: random samples lie on the right and real samples on the left of the map. The dark strip represents the separation between both zones. The SOM has been labelled using the training samples themselves. The “REAL” cluster has less extension than that of random samples, because the latter have more variability. There was an almost total lack of overlapping (units labelled with both genres) between the zones. Figure 4.6 shows the map labeled with random melodies only. Only two units from the right part of the map, corresponding to the real melodies’ cluster, are activated by random melodies. Still these two units were activated mostly by real melodies when the map was calibrated. Figure 4.7 shows how real melodies are constrained to the right side of the map, with no units in the left side being activated by them.

The descriptors that contribute more to that separation are those having a higher correlation among the samples in each of the zones. The planes in the weight space (see Fig. 4.8) corresponding to each descriptor provide information about this. An analysis of the most contributive features is an indication about how the discrimination has been carried out. For example:

¹This author firmly believes that only the hazard has made this Sammon figure to take the form of a brass instrument device called ‘sordina’ (mute).

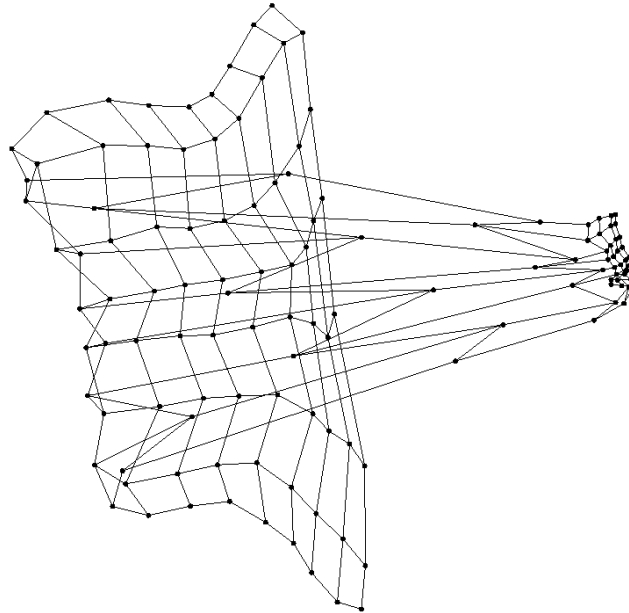


Figure 4.3: Sammon projection of a 16×8 SOM map trained with unrestricted pitch range random melodies vs. jazz melodies.

- *Pitch range.* In the random melodies, extreme note pitches appear more often than in real melodies.
- *Standard deviation of the pitches.* Is clearly larger for random melodies, due to the lack of a 'tonal centre' that usually acts like an attractor for the melodic line.
- *Average interval.* Much higher in random samples. Intervals higher than an octave are seldom present in real melodies. The average interval is usually between 2 and 3 for real melodies and higher than 10 for random ones.

The planes in weight space of the SOM for lesser correlated descriptors are shown in figure [4.9](#).

Jazz versus classical music

In another visualization experiment, melodies from classical samples in JvC1 have been substituted for the random samples. 522 classical music melody

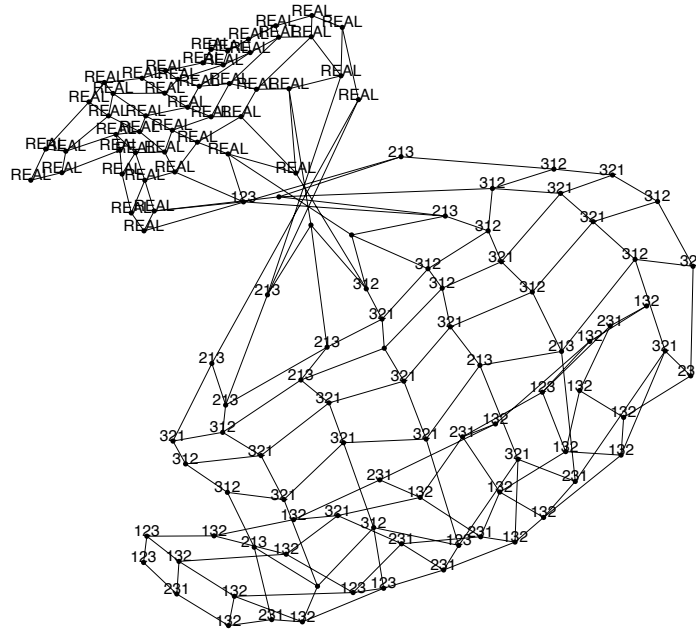


Figure 4.4: Sammon projection of the 16×8 map of figure 4.5: restricted pitch range random melodies versus real melodies.

fragments eight bars long were extracted from MIDI files for training, along with the previous 430 jazz samples. SOM training parameters are shown in table 4.2. The SOM obtained is shown in Figure 4.10.

When analysing the map it is observed that the left side is more “jazzy” and the right one is more classical, but there are a lot of mixed labels, making it harder to determine zones that could be assigned clearly to each genre. However, the overlapping in the map, that is, the ratio of units activated by samples from both genres, is relatively low (7.22%). This facts are fairly evident in figure 4.11. The overlapping probably evidences that there are some jazz melodies quite similar to classical music and vice versa, at least in terms of the description model used.

4.1. FEATURE SPACE EXPLORATION BY SOM

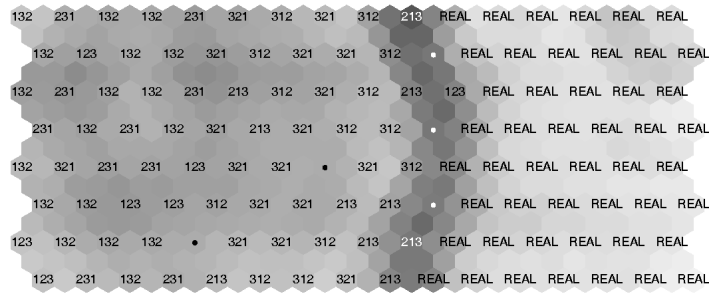


Figure 4.5: SOM map for the same weights as in figure 4.4.

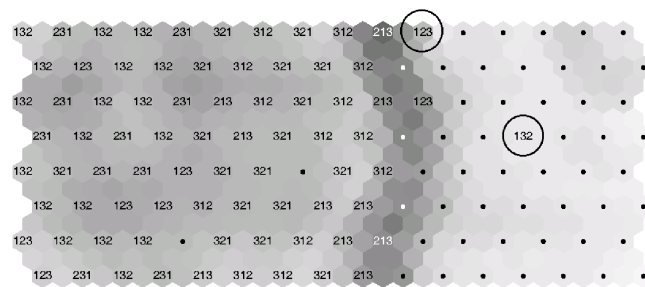


Figure 4.6: 16x8 SOM calibrated with random melodies only.

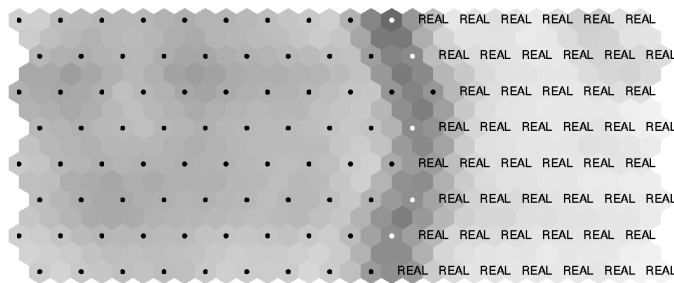


Figure 4.7: 16x8 SOM calibrated with real melodies only.

Geometry	30x12, hexa., bubble	
Trials	20	
	Phase I	Phase II
Iterations	10,000	100,000
$\alpha(0)$	0.1	0.05
$r(0)$	20	6
Q. Error	—	

Table 4.2: SOM parameters for Jazz vs. classical music training.

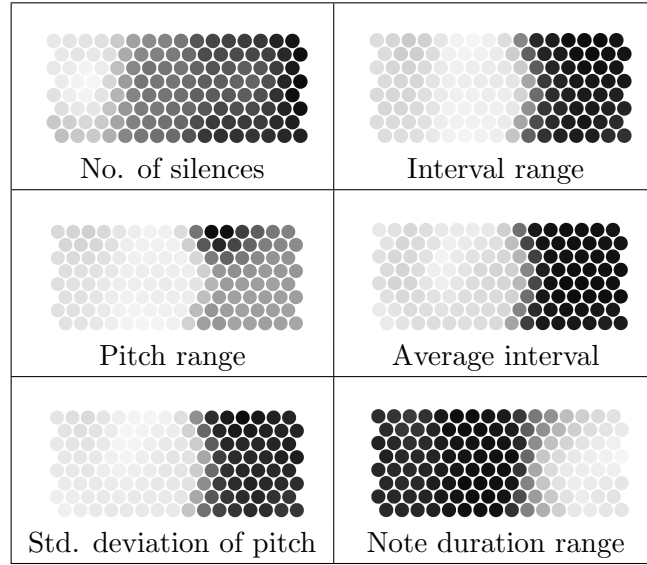


Figure 4.8: Weight planes of most contributive descriptors for the map in Fig. 4.5.

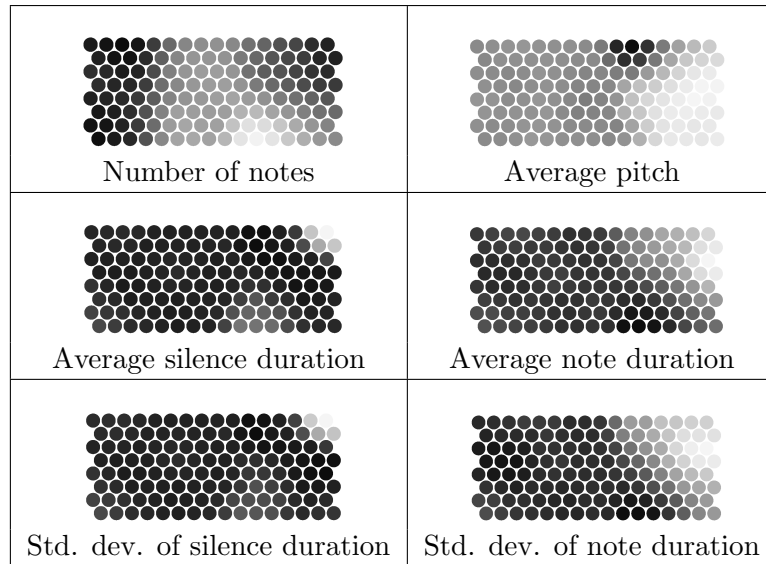


Figure 4.9: Descriptors with lesser influence on the configuration of the map in Fig. 4.5.

4.1. FEATURE SPACE EXPLORATION BY SOM



Figure 4.10: SOM after training using two music genres: (top) units activated by jazz samples and (bottom) units activated by classical music.

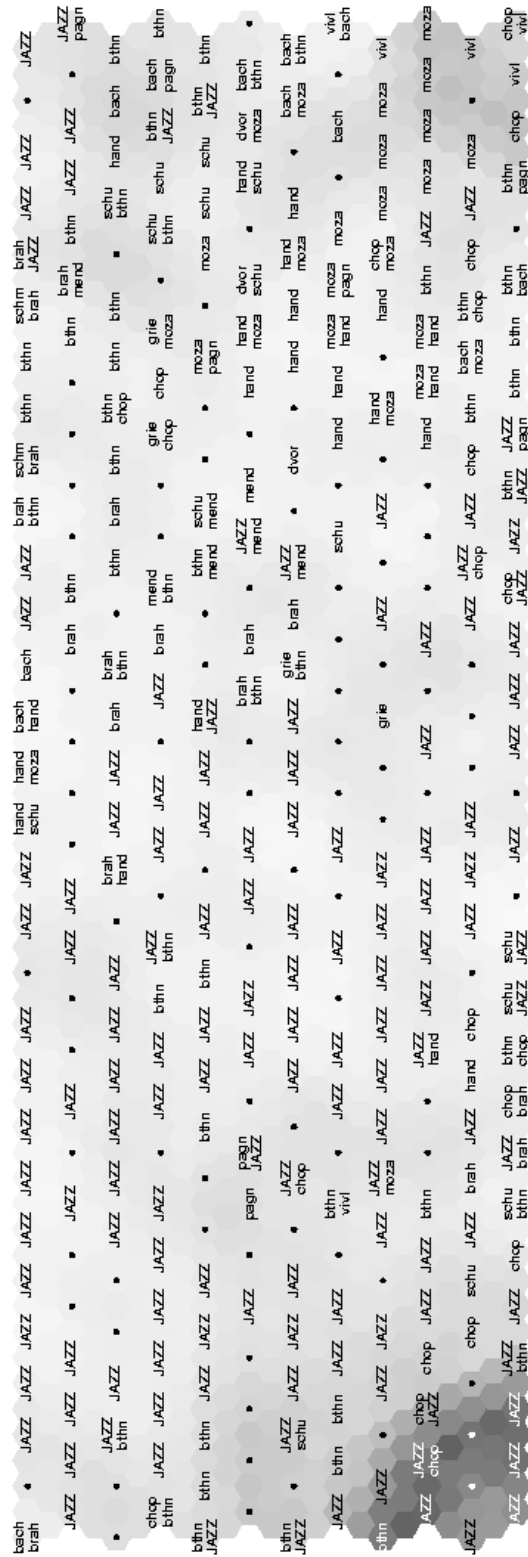


Figure 4.11: SOM trained with two genres (mixed labels). Units activated by classical samples from pieces of a number of composers are labelled with abbreviations of their names.

4.1. FEATURE SPACE EXPLORATION BY SOM

In the Sammon projection of figure 4.14 a knot separates two zones in the map. The zone at the left of the knot has a majority presence of units labelled with the jazz label and the zone at the right is mainly classical.

4.1.3 Visualizing temporal structure

While SOMs can certainly be used for classification, and in this case to classify music, as other authors successfully did, this section places emphasis on their visualization capabilities.

As an example, Figures 4.12 and 4.13 show how an entire melody is located in a SOM map. One melody of each genre is shown. The first figure displays three choruses of the standard jazz tune *Yesterdays*, and the second is a section of the Allegro movement from the *Divertimento in D* by Mozart. The grey area in each map corresponds to the genre the melody belongs to. The first map picture for each melody shows the map unit activated by the first fragment of the melody. The next pictures show the map unit activated by the next fragment of that melody. Consecutively activated units are linked by a straight line, displaying the path followed by the melody in the map.

This kind of visualization has great potential for being integrated in graphical applications that deal with music genre, like the one described in appendix A.3. The images can be presented in sequence to the user, creating a pseudo-animated picture, while she is listening to the music or browsing its different parts.

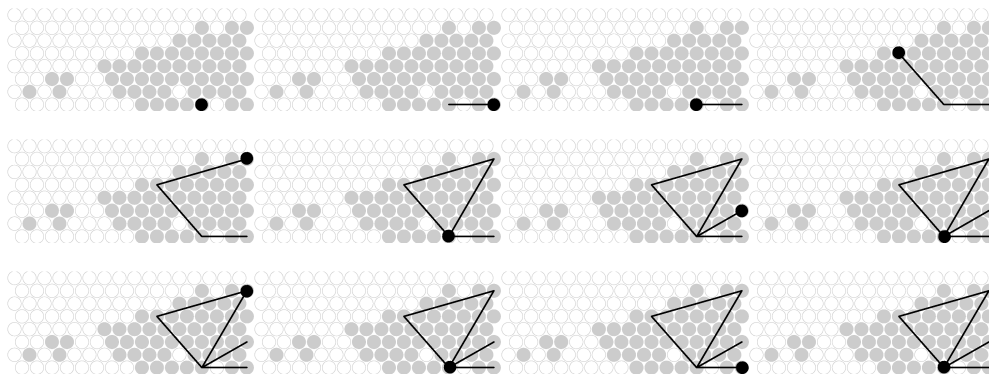


Figure 4.12: Trajectory of the winning units for the jazz standard *Yesterdays*.

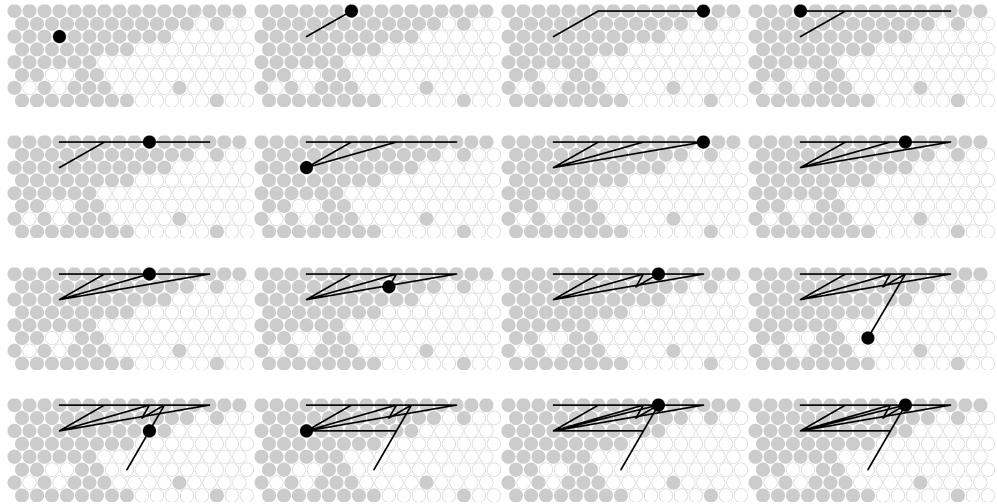


Figure 4.13: Trajectory of the winning units for the *Divertimento in D* (Mozart).

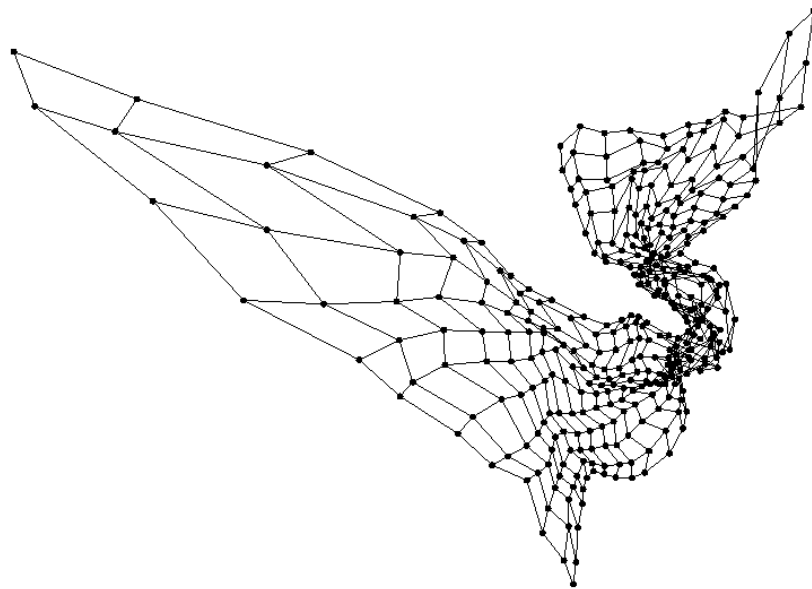


Figure 4.14: Sammon projection of the SOM map in figure 4.11.

4.2 Human music genre recognition

A number of computer systems have been published that are able in some degree to categorize music data both from audio (Soltau et al., 1998; Tzanetakis and Cook, 2002b; Zhu et al., 2004) or scores (Cruz-Alcázar et al., 2003; McKay and Fujinaga, 2004; Pérez-Sancho et al., 2005) in digital formats. Recently, papers have appeared trying to combine the best of both worlds (Cataltepe et al., 2007; Lidy et al., 2010a; Pérez-Sancho et al., 2010). Some authors claim for the need of having ground truth studies on music genre recognition, in order to compare results with them and lead to sound conclusions when analyzing software performances (Craft et al., 2007; Lippens et al., 2004).

When dealing with digital scores in any format (MIDI, MusicXML,...), timbral information is not always available or trustworthy because it depends on good sequencing practices. Furthermore, this thesis tries to study the extent to which genre information is conveyed by symbolically encoded melodies. So, in this context, the use of information about timbre has been avoided in computer models discussed in this manuscript, focusing only on the information coded by the notes in the melody. Under these conditions, some important questions arise: is a particular success rate in automatic genre classification good or bad? what is the human ability for recognizing the music genre of a melody just from the notes in the score? what remains of genre when no timbral information is provided?

Genre classification is of a hierarchical nature, so experiments should be placed in a given level of the hierarchy. It is non sense to classify between the whole classical music domain and a particular sub-genre like, for example, hip-hop. On the other hand, genre labels are inherently subjective and influenced by a number of cultural, art, and market trends, therefore perfect results can not be expected (Lippens et al., 2004; McKay and Fujinaga, 2008).

In (Lippens et al., 2004), the authors design a set of experiments for compare the results obtained by automatic computer models and by humans. For that, they utilized fragments of 30 seconds of 160 commercial recordings from classical, dance, pop, rap, rock, and ‘other’ (none of the previous labels). Those fragments were classified by a number of pattern recognition algorithms using different features extracted from audio. They were also presented to a set of 27 human listeners that were asked to choose a musical genre out of the 6 possibilities given above. In summary, the results reported a 65% of correct classification by the computer against a 88% for the human listeners. These results show that there is still a gap with human abilities when dealing with the audio data, where all the musical information (melody,

CHAPTER 4. MUSIC GENRE RECOGNITION

harmony, rhythm, timbre, etc.) is present. This is no surprise, since the data were presented in the way humans use to enjoy music, so our abilities to perform this task (in spite of subjectivity and other considerations) have been trained for years and we have a huge background knowledge compared to the training set used by the machine. Thus we are in a clear dominant position when competing against those artificial intelligence models.

The main goal of the study presented here is to compete with a machine model in equal conditions. For this, a survey has been carried out, presenting humans the same information available for the computer counterpart: fragments of melodies without accompaniment and timbral information. This way, a ground-truth reference on the human ability for recognizing music genres in absence of timbre is obtained, that can be used to assess comparatively the performance of computer models for this task. The survey has been designed around two well-established genres in the music literature, like classical and jazz music.

4.2.1 Method

Subjects

The melodic fragments were presented to 149 subjects (109 male and 40 female) classified into 3 groups: A) professional musicians (performers and teachers), B) amateurs (both musicians and music lovers), and C) a control group composed of people with no particular relation to music practice. Table 4.3 shows the statistics on the subjects to whom the test was applied.

Group	Number	Male	Female	Age
Profess.	29	18	11	28.3 ± 8.0
Amateurs	57	46	11	27.2 ± 6.3
Control	63	42	21	29.3 ± 6.2

Table 4.3: Statistical profile of the people subjected to the test.

Sex Despite the uneven distribution of people by sex (106 male, 43 female), no bias was detected in the answers according to this variable.

Age The minimum age was 9 years old and the maximum was 60. The overall average was of 28.1 years with a standard deviation of ± 9 .

4.2. HUMAN MUSIC GENRE RECOGNITION

Level of studies Two criteria were adopted: classification under their general studies and their specific music studies. Different categories were established:

General studies (number of subjects):

- 1: Elementary education (6)
- 2: Secondary education (75)
- 3: Graduate studies (12)
- 4: Master (43)
- 5: Doctorate (13)

Music studies (number of subjects):

- 0: No studies (43)
- 1: Self-trained (48)
- 2: Not-finished conservatory (12)
- 3: Conservatory elementary degree (9)
- 4: Conservatory intermediate degree (19)
- 5: Conservatory high degree (8)
- 6: Musicology (10)

Melodies

Concerning to the music data, a set of 40 melody fragments (20 of classical music and 20 jazz pieces), were synthesized using sinusoidal waves (just a fundamental frequency without timbral relation among spectral components). They were cut from the respective MIDI sequences by an expert. The durations were in average 19.4 ± 4.2 seconds in a range [12,30] (33 ± 32 [12,62] beats, 8.4 ± 8.0 [3,16] bars). In terms of number of notes, the range was [17,171] averaging 46.

The classical fragments covered a wide range of periods from Baroque (Haendel, Bach, Vivaldi,...) to Classical (Mozart, Paganini, Beethoven,...) and Romantic (Schumann, Schubert, Mendelssohn, Brahms,...). Jazz fragments were standards from a variety of genres like Pre-Bop, Bop, Bossanova, or fusion (Charlie Parker, Thelonious Monk, Antonio Carlos Jobim, Wayne Shorter,...).

All the fragments were pre-classified by an expert according to their *a priori* difficulty for being classified. For that, melodic, harmonic, and rhythmic aspects of the melodies were taken into account. Also their general public popularity was considered for assigning a difficulty degree for each

CHAPTER 4. MUSIC GENRE RECOGNITION

fragment. For jazz, 5 were considered ‘easy’, 8 ‘intermediate’, and 7 ‘difficult’, and for classical, it was 11, 6, and 3 respectively.

When presented to the subject (just once) he or she must identify whether the melody belongs to a classical or jazz piece. The fragments were randomly ordered for presentation, using always the same ordering.

4.2.2 Results

A number of analyses in terms of age, group, education, and music studies have been performed. Also, the difficulty level of the fragments, according to the *a priori* classification explained above, have been taken into account. The results (see Table 4.4) show that, on average, the error rate was 16.2%, although it ranged from 5.9% for the professionals to 19.2% for the control group. Note that there were no significant differences between amateurs and control. Only professional musicians performed much better than the other groups, showing much higher classification skills.

	control	amateurs	professionals
Error %	19.2	18.0	5.9

Table 4.4: Error percentages in terms of group of people.

The *a priori* difficulty of the fragments was clearly reflected in the ability for recognizing the genre (see Table 4.5) increasing from a 3.5% average error rate for the easy ones to 23.2% for those considered difficult. Note that the error rate for difficult fragments is more than twice that for the intermediate ones (10.8%).

This fact can also be seen in the distribution of errors for fragments (see Figure 4.15). All subjects gave the correct answer (zero errors) for Haendel’s “Fireworks - La Rejouissance” (an ‘easy’ fragment). For Jazz, just one error was committed for Telonious Monk’s “Well You Needn’t” (an ‘easy’ fragment too). In contrast, the maximum number of errors (102, a 68.5% of the total number of tests) were made for “Young and Foolish”, a jazz tune by Horwitt and Hague, while in classical music Schubert’s Symphony no. 4 in C minor “Tragic” received 61 misclassifications (40.9% of the tests).

The number of answers that classified the fragments as classical was 56.4% (43.6% for jazz). This bias is due to the fact that, in general, people is more familiar with classical tunes and tend to think that an unknown fragment is classical, in the actual case of absence of timbre, only because they are usually more exposed to this genre.

4.2. HUMAN MUSIC GENRE RECOGNITION

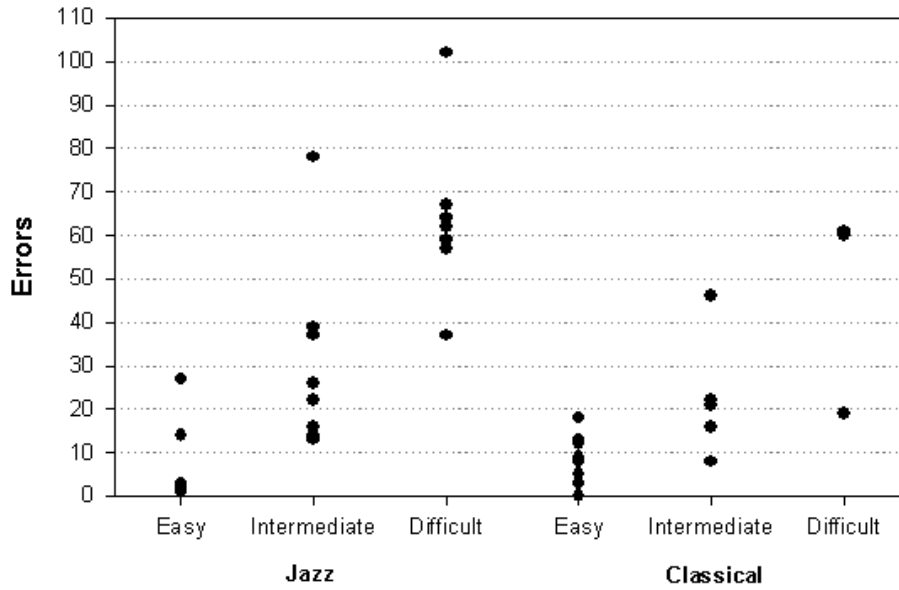


Figure 4.15: Number of errors as a function of the difficulty of the fragments.

(%)	easy	interm.	difficult
Jazz	3.5	14.3	27.8
Classical	3.5	9.3	21.0
Average	3.5	10.8	23.2

Table 4.5: Error percentages in terms of the difficulty levels assigned.

A negative correlation with age and general studies ($r = -0.28$) was observed (see figure 4.16). This suggests that people's experience is important in this ability. This is not surprising because through their lives people hear music and, even if they are not experts, they accumulate arguments in order to decide which kind of music they are hearing.

The evolution of the error as a function of study levels is also an important issue (see figure 4.17). Note that the error percentages are lower for higher levels of general studies (dark columns in the graph). More interesting and significant is to see what happens for different music studies (light columns). The higher the music studies the lower the error rate, and this tendency is neat in the graph. But there is the remarkable exception of musicologists, that performed clearly poorer than the average, showing a professional bias. We can consider (speaking in terms of a classification system) that they are 'overtrained'. Their high level of musical knowledge leads them to match fragments with some theme in their knowledge.

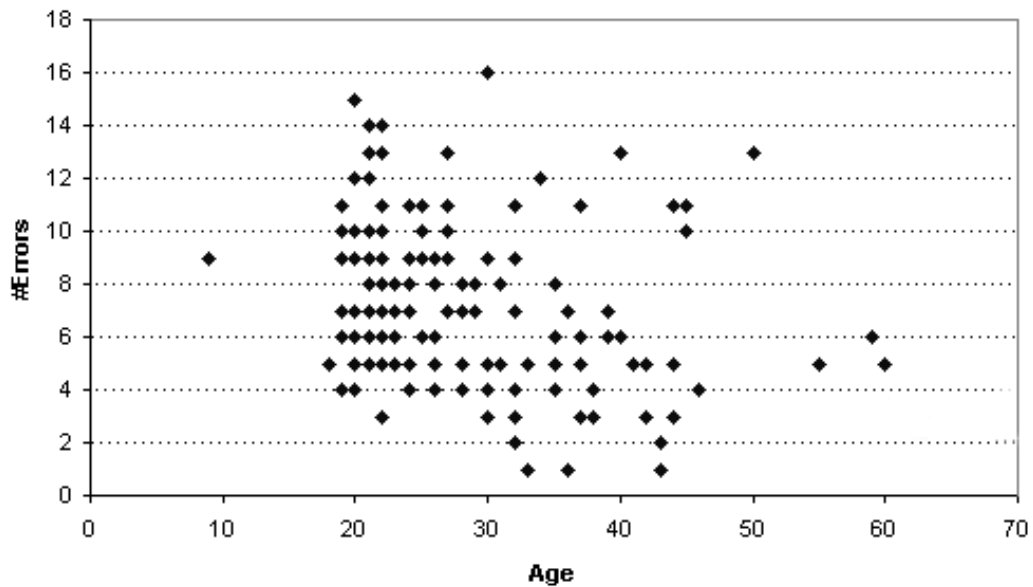


Figure 4.16: Number of errors as a function of the age.

Most of the professionals were involved in works related to classical music, so they bias their decisions in this direction. Control subjects answered classical in 55% of the queries, 56% for amateurs, while professionals did it 58% of times.

4.2.3 Conclusions

The results show that people are able to distinguish quite well between classical and jazz melodies when a timbre-less fragment is presented (roughly 4 out of 5 fragments were correctly classified). In other, less structured, experiments no one have had difficulties when these same fragments were presented with the timbral information (utilizing a synthesizer and the whole MIDI sequences). This suggests that something about music genre remains in just the melody notes without timbre, at least between well established genres like classical and jazz music. This is more doubtful if we need to distinguish between closer genres where timbre is a key feature, like for example pop and rock.

For use in computer music information retrieval experiments as a base line, a 16% of error shows the performance level to be improved if authors claim to report good results for this task without using timbral information.

The next section discusses research on automatic music genre recognition by machine learning methods, in the absence of timbre. Due to the

4.2. HUMAN MUSIC GENRE RECOGNITION

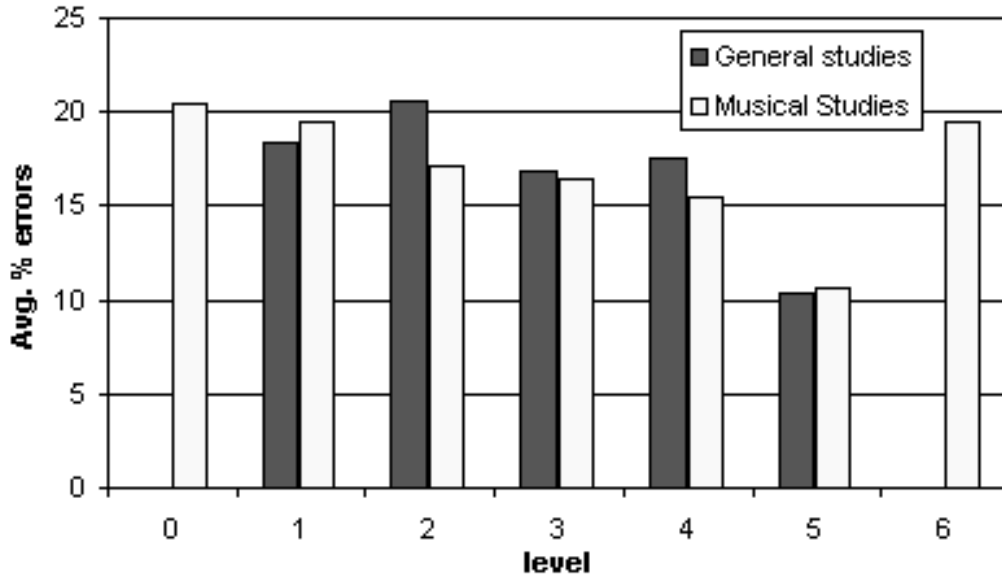


Figure 4.17: Number of errors as a function of both the general and musical level of studies.

methodology used in that research, based on fixed sliding window lengths, it is not possible to compare results with those obtained here directly, as the melodic fragments used in this survey are of variable length. However, given that these fragments range from 3 to 16 bars long, averaged results over this range can be informally compared to the 16% error (84% success) base line suggested above. The automatic genre recognition experiments were carried out with a variety of machine learning methods and feature sets. The classifiers used were Quadratic Bayes, SOM, 1-NN, Multi-Layer Perceptron (MLP), and Support Vector Machine (SVM). Four different statistical feature sets, of size 6, 10, 12, and 28, were utilized. Table 4.6 presents averaged error rates over sliding window lengths from 3 to 16, for every classifier / feature set combination. Both MLP and SVM achieve error rate below the human judgment base line, with almost every feature set, for the range of bars considered. 1-NN error rates are a few points above the base line. The other classification methods give not so good results at the fragment lengths considered, but they do much better when trained (and tested) on larger fragment sizes (see section 4.3.3). While these comparative results must be taken with caution, they suggest that MLP and SVM classifiers are better suited than the other classifiers to deal with melodic fragments in the size range considered.

CHAPTER 4. MUSIC GENRE RECOGNITION

Classifier	Feat. set size			
	28	12	10	6
Q. Bayes	49.8	14.5	17.9	21.0
SOM	23.9	21.1	20.8	17.0
1-NN	18.9	18.7	18.3	19.0
MLP	11.8	12.9	16.2	14.9
SVM	12.4	11.4	12.5	13.2

Table 4.6: Averaged error rates (%) over sliding window lengths from 3 to 16, for several classifier / feature set combination.

4.3 Supervised music genre recognition

One of the problems to solve in MIR is the modeling of music genre from tagged corpora. The computer could be trained to recognise the main features that characterise music genres in order to look for that kind of music over large musical databases. The same scheme is suitable to learn stylistic features of composers or even model a musical taste for users. Other application of such a system can be its use in cooperation with automatic composition algorithms to guide this process according to a given stylistic profile.

A number of papers explore the capabilities of machine learning methods to recognise music genre. (Pampalk, 2006) use self-organising maps (SOM) to pose the problem of organising music digital libraries according to sound features of musical themes, in such a way that similar themes are clustered, performing a content-based classification of the sounds. Whitman et al. (Whitman et al., 2001) present a system based on neural networks and support vector machines able to classify an audio fragment into a given list of sources or artists. Also in (Soltan et al., 1998) a neural system to recognise music types from sound inputs is described. An *emergent* approach to genre classification is used in (Pachet et al., 2001), where a classification emerges from the data without any *a priori* given set of genres. The authors use co-occurrence techniques to automatically extract musical similarity between titles or artists. The sources used for classification are radio programs and databases of compilation CDs. In (Lidy et al., 2010b) the authors analyze the performance of a range of automatic audio description algorithms on three music databases with distinct characteristics, specifically a Western music collection used previously in research benchmarks, a collection of Latin American music with roots in Latin American culture, but following Western

4.3. SUPERVISED MUSIC GENRE RECOGNITION

tonality principles, as well as a collection of field recordings of ethnic African music.

Other works use music data in symbolic form (most MIDI data) to perform genre recognition. One of the seminal works in this domain is (Dannenberg et al., 1997), where the authors use a naive Bayes classifier, a linear classifier and neural networks to recognize up to eight moods (genres) of music, such as lyrical, frantic, etc. Thirteen statistical features derived from MIDI data are used for this genre discrimination. In (Tzanetakis et al., 2003), pitch features are extracted both from MIDI data and audio data and used separately to classify music from five genres. Pitch histograms regarding to the tonal pitch are used in (Thom, 2000) to describe blues fragments of the saxophonist Charlie Parker. Also pitch histograms and SOM are used in (Toivainen and Eerola, 2001) for musicological analysis of folk songs. Other researchers use sequence processing techniques like Hidden Markov Models (Chai and Vercoe, 2001) and universal compression algorithms (Dubnov and Assayag, 2002) to classify musical sequences.

Stamatatos and Widmer (Stamatatos and Widmer, 2002) use stylistic performance features and the discriminant analysis technique to obtain an ensemble of simple classifiers that work together to recognize the most likely music performer of a piece given a set of skilled candidate pianists. The input data are obtained from a computer-monitored piano, capable of measuring every key and pedal movement with high precision.

Compositions from five well known eighteenth-century composers are classified in (van Kranenburg and Backer, 2004) using several supervised learning methods and twenty genre features, most of them being counterpoint characteristics. This work offers some conclusions about the differences between composers discovered by the different learning methods.

In other work (Cruz-Alcázar et al., 2003), the authors show the ability of grammatical inference methods for modeling musical genre. A stochastic grammar for each musical genre is inferred from examples, and those grammars are used to parse and classify new melodies. The authors also discuss about the encoding schemes that can be used to achieve the best recognition result. Other approaches like multi-layer feed-forward neural networks (Buzzanca, 2002) have been used to classify musical genre from symbolic sources.

(Perez-Sancho et al., 2009; Pérez-Sancho, 2009b) approach the symbolic music genre recognition through the use of stochastic language models. Genres are modeled as languages, using techniques successfully applied to language modeling in text information retrieval problems. n -grams and n -words extracted from note or chord sequences are used as positives examples

CHAPTER 4. MUSIC GENRE RECOGNITION

of their respective genres, and subsequently used to build language models capable of predicting the genre of new music instances.

McKay and Fujinaga ([McKay, 2010](#); [McKay and Fujinaga, 2004, 2007a](#)) use low and mid-level statistics of MIDI file content to perform music genre recognition by means of genetic algorithms and pattern recognition techniques. They have developed several tools for feature extraction from music symbolic sources (particularly MIDI files) or web sites (([McKay and Fujinaga, 2006a, 2007b](#))). In ([McKay and Fujinaga, 2006b](#)), the authors provide some insight on why is it worth continuing research in automatic music genre recognition, despite the fact that the ground-truth information available for research is often not too reliable, subject to market forces, subjective tagging, or being culture-dependent. Most of the classification problems detected seem to be related to the lack of reliable ground-truth, from the definition of realistic and diverse genre labels, to the need of combining features of different nature, like cultural, high- and low-level features. They also identify, in particular, the need for being able to label different sections of a music piece with different tags.

The system presented in this section share some characteristics with the one developed by McKay, as the use of low level statistics and pattern recognition techniques but, while McKay extract features from the MIDI file as a whole, our system focus on melody tracks, using a *sliding window* technique to obtain melody segments that become instances to feed the pattern recognition tools. This allows to obtain partial decisions for a melody track that can offer the users sensible information for different parts of a music work. Also, this decisions can be combined to output a classification decision for a music piece. Another difference to point out is the fact that no timbre related features are used in this thesis, which were identified as an important source of information related to genre in ([McKay and Fujinaga, 2005](#)).

4.3.1 Supervised music genre classification of melodic fragments

In this section a framework for experimenting on automatic music genre recognition from symbolic representation of melodies (digital scores) is presented. It is based on shallow structural features of melodic content, like melodic, harmonic, and rhythmic statistical descriptors. This framework involves all the usual stages in a pattern recognition system, like feature extraction, feature selection, and classification stages, in such a way that new features and corpora from different musical genres can be easily incorporated and tested.

4.3. SUPERVISED MUSIC GENRE RECOGNITION

The working hypothesis is that melodies from a same musical genre may share some common low-level features, permitting a suitable pattern recognition system, based on statistical descriptors, to assign the proper musical genre to them.

Two well-defined music genres, like jazz and classical, have been initially chosen as a workbench for this research. The first results were encouraging (Ponce de León and Iñesta, 2003) but the method performance for different classification algorithms, descriptor models, and parameter values needed to be thoroughly tested. This way, a framework for musical genre recognition needed to be set up, where new features and new musical genres could be easily incorporated and tested.

This section presents the proposed methodology, describing the musical data, the descriptors, and the classifiers used. The initial set of descriptors will be analyzed to test their contribution to the musical genre separability. These procedures will permit us to build reduced models, discarding not useful descriptors. Then, the classification results obtained with each classifier and an analysis of them with respect to the different description parameters will be presented.

Musical data

The JvC1 corpus (sec. 2.8.4), has been used for this research. Its length is around 10,000 bars (more than 6 hours of music). Table 4.7 summarizes the distribution of bars per song from each genre. This corpus has been manually checked for the presence and correctness of key, tempo, and meter meta-events, as well as the presence of a monophonic melody track. The original conditions under which the MIDI files were created are uncontrolled; They may be human performed tracks or sequenced tracks (i.e. generated from scores) or even something of both worlds. Nevertheless, most of the MIDI files seem to fit a rather common scheme: a human-performed melody track with several sequenced accompaniment tracks.

	Min.	Max.	Avg.	Total
JAZZ	16	203	73	4734
CLAS	44	297	116	5227

Table 4.7: Distribution of melody length in bars

The monophonic melodies consist of a sequence of musical events that can be either notes or silences. The pitch of each note can take a value from

CHAPTER 4. MUSIC GENRE RECOGNITION

0 to 127, encoded together with the MIDI note onset event. Each of these events at time t has a corresponding note off event at time $t + d$, being d the note duration measured in ticks². Time gaps between a note off event and the next note onset event are silences.

Description scheme

Each sample is a vector of statistical descriptors computed from each melody segment available (See section 4.3.1 for a discussion about how these segments are obtained). Each vector is labeled with the genre of the melody which the segment belongs to. A set of descriptors has been defined, based on a number of feature categories that assess the melodic, harmonic and rhythmic properties of a musical segment, much as has been done in chapter 3 for melody part selection.

This description model is made up of 28 descriptors summarized in table 4.8. Regarding silence descriptors, the adjective *significant* stands for silences explicitly written in the underlying score of the melody. In MIDI files, short gaps between consecutive notes may appear due to interpretation nuances like *stacatto*. These gaps (interpretation silences) are not considered significant silences since they should not appear in the score. To make a distinction between types of silence is not possible from the MIDI file, since silences are not explicitly encoded. They are distinguished based on the definition of a silence duration threshold. This value has been empirically set to a duration of a sixteenth note. All silences with longer or equal duration than this threshold are considered significant.

Note duration, silence duration and IOI³ descriptors are measured in ticks and computed using a time resolution of $Q = 48$ ticks per bar⁴. Interval descriptors are computed as the difference in absolute value between the pitches of two consecutive notes. Descriptors designed to identify harmonic traits in genre are the following:

- *Number of non diatonic notes*. An indication of frequent excursions outside the song key (a metaevent that must be present in the MIDI file⁵) or modulations.

² A *tick* is the basic unit of time in a MIDI file and is defined by the resolution of the file, measured in ticks per beat.

³Inter Onset Interval. It is the distance, in ticks, between the onsets of two consecutive notes. Two notes are considered consecutive even in the presence of a silence between them.

⁴ This is called *quantisation*. $Q = 48$ means that when a bar is composed of 4 beats, each beat can be divided, at most, into 12 ticks.

⁵ If no key metaevent is present, a tonality guessing method like (Rizo et al., 2006b).

4.3. SUPERVISED MUSIC GENRE RECOGNITION

- *Average degree of non diatonic notes.* Describes the kind of excursions. This degree is a number between 0 and 4 that indexes the non diatonic notes of the diatonic scale of the tune key, that can be major or minor key⁶
- *Standard deviation of degrees of non diatonic notes.* Indicates variety in the non diatonic notes.

Normality descriptors are computed using the D’Agostino statistic for assessing the distribution normality of the n values v_i in the segment for pitches, durations, intervals, etc. The test is performed using this equation:

$$D = \frac{\sum_i (i - \frac{n+1}{2}) v_i}{\sqrt{n^3 (\sum_i v_i^2 - \frac{1}{n} (\sum_i v_i)^2)}} \quad (4.1)$$

For pitch and interval properties, the range descriptors are computed as maximum minus minimum values, and the average descriptors are computed as the average value minus the minimum value (only considering the notes in the segment). This make them invariant to transposition. For durations (note duration, silence duration, and IOI descriptors) the range descriptors are computed as the ratio between the maximum and minimum values, and the average-relative descriptors are computed as the ratio between the average value and the minimum value.

This descriptive statistics is similar to histogram-based descriptions used by other authors (Thom, 2000; Toivainen and Eerola, 2001) that also try to model the distribution of musical events in a music fragment. Assuming normality in the distribution of musical properties, and computing the range, mean, and standard deviation from it, the number of features needed are reduced (each histogram may be made up of tens of features).

Free parameter space

Given a melody track, the statistical descriptors presented above are computed from equal length segments extracted from the track, by defining a window of size ω measures. Once the descriptors of a segment have been extracted, the window is shifted δ measures forward to obtain the next segment to be described. Given a melody with $m > 0$ measures, the number of segments s of size $\omega > 0$ obtained from that melody is

⁶ Non diatonic degrees are: 0: \flat II, 1: \flat III (\sharp III for minor key), 2: \flat V, 3: \flat VI, 4: \flat VII. The key is encoded at the beginning of the melody track. It has been manually checked for correctness in our data.

CHAPTER 4. MUSIC GENRE RECOGNITION

Category	Descriptors
Overall	Number of notes Number of significant silences Number of non-significant silences
Pitch	Pitch range Average pitch Dev. pitch
Note duration	Note duration range Avg. note duration Dev. note duration
Silence duration	Silence duration range Avg. silence duration Dev. silence duration
Inter Onset Interval	IOI range Avg. IOI Dev. IOI
Pitch interval	Interval range Avg. interval Dev. interval
Non-diatonic notes	Num. non-diatonic notes Avg. non-diatonic degrees Dev. non-diatonic degrees
Syncopation	Number of syncopes
Normality	Pitch distrib. normality Note duration distrib. normality Silence duration distrib. normality IOI distrib. normality Interval distrib. normality Non-diatonic degree distrib. normality

Table 4.8: Statistical musical descriptors considered.

$$s = \begin{cases} 1 & \text{if } \omega \geq m \\ 1 + \left\lceil \frac{m-\omega}{\delta} \right\rceil & \text{otherwise} \end{cases} \quad (4.2)$$

showing that at least one segment is extracted in any case (ω and s are positive integers; m and δ are positive fractional numbers).

Taking ω and δ as free parameters in our methodology, different datasets of segments have been derived from a number of values for those parameters. The goal is to investigate how the combination of these parameters influences the segment classification results. The exploration space for this parameters will be referred to as $\omega\delta$ -space. A point in this space is denoted as $\langle \omega, \delta \rangle$.

ω is the most important parameter in this framework, as it determines the amount of information available for the descriptor computations. Small values for ω would produce windows containing few notes, providing less

4.3. SUPERVISED MUSIC GENRE RECOGNITION

reliable statistical descriptors. Large values for ω would lead to merge –probably different– parts of a melody into a single window and they also produce datasets with fewer samples for training the classifiers (see Eq. 4.2). The value of δ would affect mainly to the number of samples in a dataset. A small δ value combined with quite large values for ω may produce datasets with a large number of samples (see also Eq. 4.2). The details about the values used for these parameters can be found in section 4.3.2.

Feature analysis and selection

The features described above have been designed according to those used in musicological studies, but there is no theoretical support for their genre classification capability. It should be of interest to check how each feature does perform on its own on the problem at hand. An analysis of their distribution and independent discrimination potential on the music genre recognition task has been performed. This analysis results are presented in section 4.3.2.

Following the previous analysis, the selection procedure described in sec 2.5.1 has been applied in order to keep those descriptors that better contribute to the classification. When this test is performed on a number of different $\omega\delta$ -point datasets, a threshold on the number of passed tests can be set as a criterion to select descriptors. This threshold is expressed as a minimum percentage of tests passed. Once the descriptors are selected, a second criterion for grouping them permits to build several descriptor models incrementally. First, selected descriptors are ranked according to their z value averaged over all tests. Second, descriptors with similar z values in the ranking are grouped together. This way, several descriptor groups are formed, and new descriptor models can be formed by incrementally combining these groups. See the section 4.3.2 for the models that have been obtained.

Classifier implementation and tuning

Two different supervised classification methods are used here for automatic genre identification: The bayesian classifier and the k NN classifier, described in section 2.2. For the Bayesian classifier, it must be assumed that individual descriptor probability distributions for each genre are normal, with means and variances estimated from the training data. This classifier computes the squared Mahalanobis distance from test samples to the mean vector of each genre in order to obtain a classification criterion. The k NN classifier uses an euclidean metrics to compute distance between samples. A number of odd values for k , ranging from 1 to 25, have been tested.

4.3.2 Experiments and results on feature selection

Analysis of independent feature performance

A series of classification experiments using only one feature at a time have been performed. The *1R* classifier has been used for this purpose. The experiments were performed for $\delta = 1$ and $\omega = 1..100$, using ten-fold crossvalidation at the MIDI file level (this partitioning scheme is discussed in section 4.3.3). Average success results are displayed in figure 4.18. Features are grouped by categories in the graphs, in order to make them visually comparable within each category.

Some interesting remarks are worth to be discussed. First, some of the descriptors seem to perform fairly well. Except for the *note duration*, and non-diatonic notes categories, the rest have at least one descriptor with an average success beyond 80% for some ω . Table 4.9 presents a summary of the best descriptor performances.

Last, there are some descriptors that present a more regular behaviour in terms of average success than others, as the *pitch range* or *pitch interval range*. However, they also suffer from the lack of robustness, as evidenced by high accuracy deviation values (Figure 4.19). The accuracy standard deviation tend to increase with ω , but this is due to the fewer number of samples available for training and testing as ω increases. This trend is inherent to the *sliding window* method, and is present in all the classification experiments performed in this work.

Find below some comments on several descriptors.

Descriptor	ω	Accuracy	Std. dev.
<i>Number of notes</i>	100	86%	± 11
<i>Std. deviation of interval</i>	98	86%	± 13
<i>Interval range</i>	83	84%	± 13
<i>Avg. relative pitch</i>	52	83%	± 9
<i>Std. dev. of silence duration</i>	100	82%	± 15
<i>Std. dev. of IOI</i>	80	82%	± 9
<i>Syncopations</i>	23	81%	± 12

Table 4.9: Best descriptor performances with *1R* classifier.

4.3. SUPERVISED MUSIC GENRE RECOGNITION

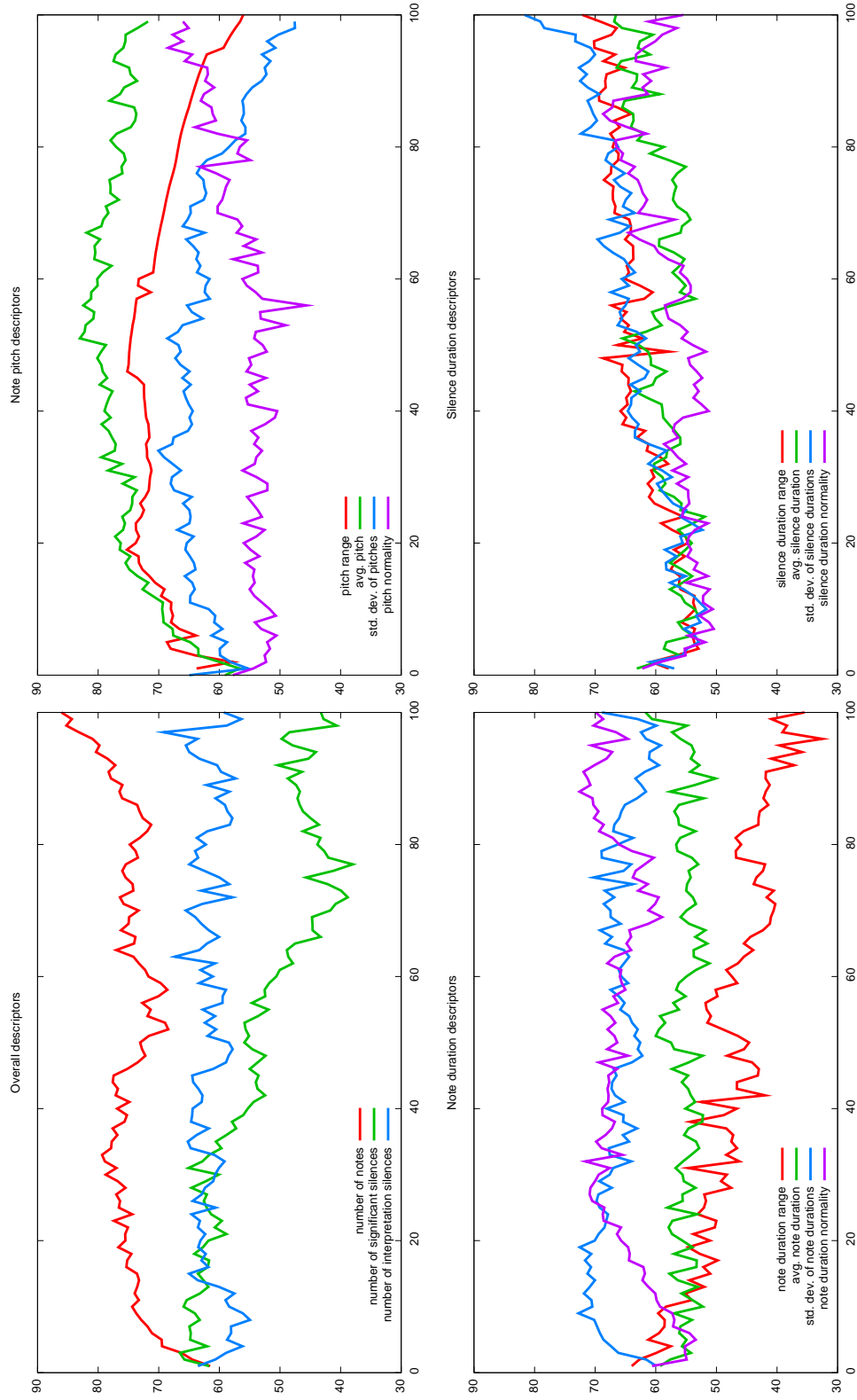


Figure 4.18: Average success rate for 1R rules using a single feature.

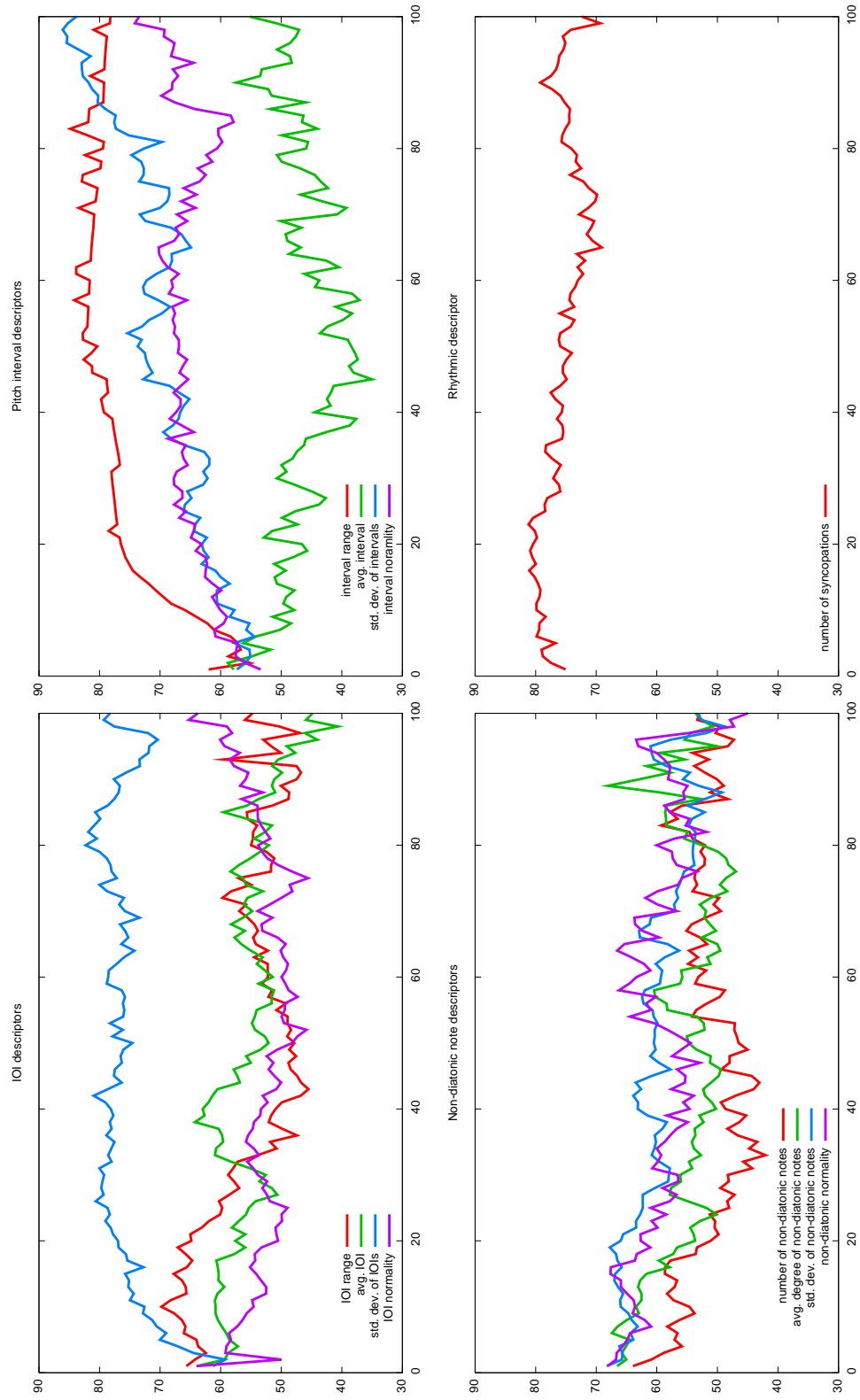


Figure 4.18: Average success rate for 1R rules using a single feature (cont.).

4.3. SUPERVISED MUSIC GENRE RECOGNITION

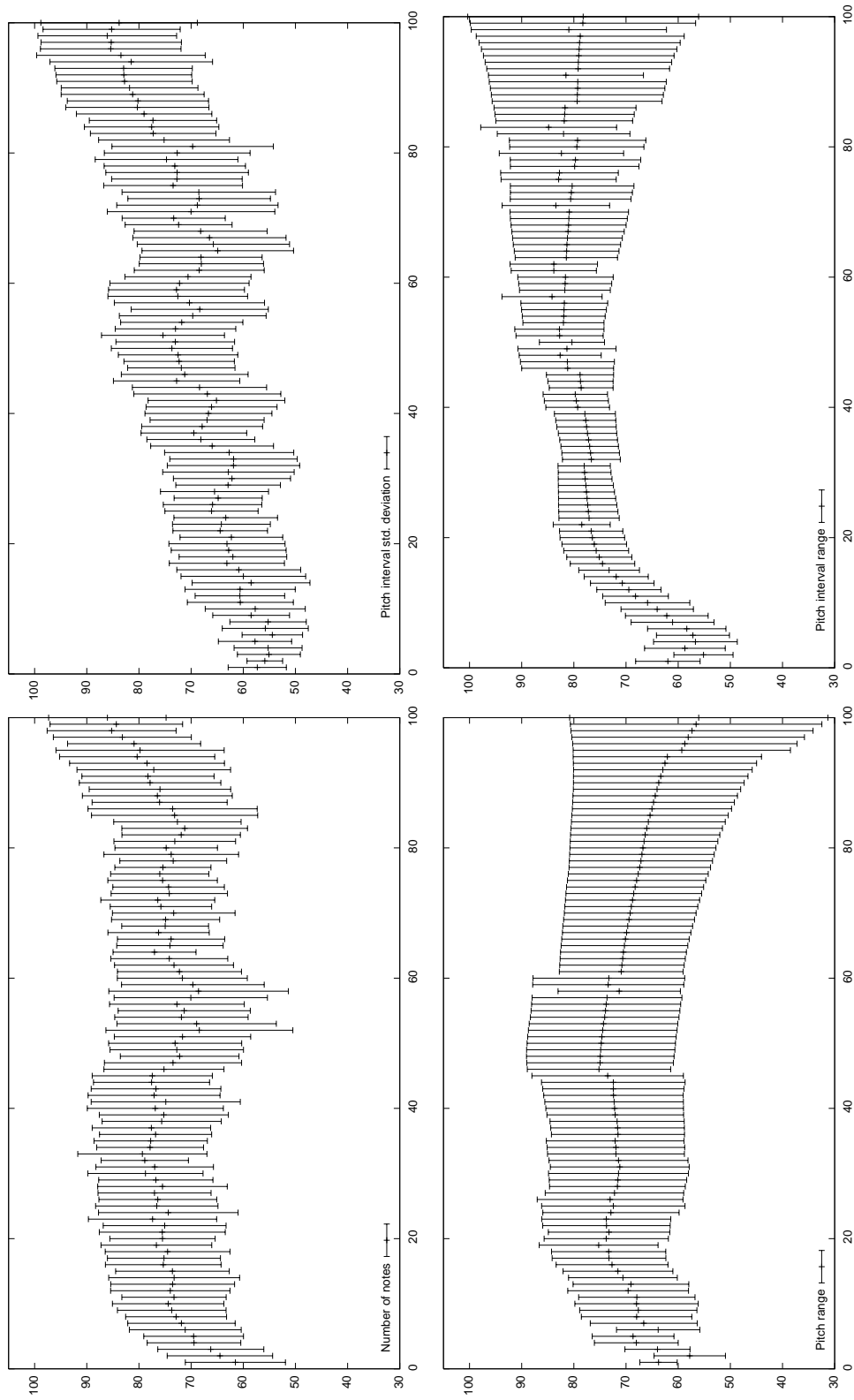


Figure 4.19: Average success and standard deviation for 1R rules for selected descriptors.

CHAPTER 4. MUSIC GENRE RECOGNITION

Number of notes Above 70% for $\omega \geq 6$, and above 80% for $\omega \geq 94$

Average pitch Over 70% for $\omega \geq 13$.

Note duration descriptors Below 70% for almost every ω .

Silence duration descriptors Below 70% for almost every ω . Std. dev. of silence duration raises above 70% from $\omega \geq 82$.

Std. dev. IOI Between 70% for and 80% for $\omega \geq 7$.

Pitch Interval range Above 70% for $\omega \geq 13$. Quite stable around 80% for $\omega \geq 46$.

Std. dev. pitch interval Increases more or less lineally with ω .

Non-diatonic descriptors Below 70% for every ω .

Num. syncopations Between 70% and 80% for every ω . Slightly better accuracy for small ω values.

It seems like some pitch related descriptors (average pitch, pitch interval range) outperform duration-based descriptors. The ‘number of notes’ descriptor perform surprisingly well at every ω , and gets an accuracy boost for very large ω values. Some descriptors related to note onsets (IOI range, num. of syncopations) perform also reasonably well across ω . In the context of melody matching, (Suyoto and Uitdenbogerd, 2005) have shown that note duration descriptors do not improve performance based on note pitches, even when used in combination. On the other hand, the same authors, (Suyoto and Uitdenbogerd, 2008), found that IOI descriptors did it when combined with pitch descriptors. Nevertheless, in our context of music genre recognition, less accurate descriptors should not be discarded for further research, as they can help in a combined description approach. This is investigated in the following sections.

Feature selection results

The feature selection test presented in section 4.3.1 has been applied to datasets corresponding to 100 randomly selected points of the $\omega\delta$ -space. This is motivated by the fact that the descriptor computation is different for each ω and the set of values is different for each δ , so the best descriptors may be different for different $\omega\delta$ -points. Thus, by choosing a set of such points, the sensitivity of the classification to the feature selection procedure can

4.3. SUPERVISED MUSIC GENRE RECOGNITION

descriptor	\bar{z}	passed tests	models
Number of notes	22.5	100%	6,10,12
Average pitch	22.3	100%	6,10,12
Pitch range	22.2	100%	6,10,12
Interval range	20.3	100%	6,10,12
Syncopation	19.6	100%	6,10,12
Dev. pitch	18.7	100%	6,10,12
number of significant silences	14.2	100%	10,12
Interval distrib. normality	14.2	100%	10,12
Dev. interval	14.0	100%	10,12
Dev. IOI	13.2	97%	10,12
Dev. note duration	9.3	95%	12
Dev. non-diatonic degrees	9.1	100%	12
Dev. silence duration	6.3	94%	–
Silence duration range	6.1	87%	–
Note duration distrib. normality	6.0	89%	–
Avg. note duration	5.6	71%	–
Avg. silence duration	5.1	85%	–
Avg. non-diatonic degrees	4.9	66%	–
IOI range	4.7	53%	–
number of non-significant silences	4.5	76%	–
Silence duration distrib. normality	4.3	45%	–
Avg. IOI	4.2	53%	–
Non-diatonic degree distrib. normality	3.5	39%	–
Note duration range	3.3	34%	–
Pitch distrib. normality	2.6	25%	–
Num. non-diatonic notes	2.5	32%	–
IOI distrib. normality	2.2	20%	–
Avg. interval	1.7	14%	–

Table 4.10: Feature selection results

be analysed. Being a random set of points is a good trade-off decision to minimise the risk of biasing this analysis.

The descriptors were sorted according to the average z value (\bar{z}) computed for the descriptors in the tests. The list of sorted descriptors is shown in table 4.10. The \bar{z} values for all the tests and the percentage of passed tests for each descriptor are displayed. In order to select descriptors, a threshold on the number of passed tests has been set to 95%. This way, those descriptors which failed the separability hypothesis in more than a 5% of the experiments were discarded from the reduced models. Only 12 descriptors out of 28 were selected. In the rightmost column, the reduced models in which the descriptors were included are presented. Each model is denoted with the number of descriptors included in it. Not surprisingly, descriptors that did best in the feature analysis stage have obtained good scores in the feature selection test and were selected to become part of reduced models.

Three reduced size models have been chosen, with 6, 10, and 12 descriptors. These models are built according to the \bar{z} value as displayed in figure 4.20. The biggest gaps in the \bar{z} values for the sorted descriptors led

CHAPTER 4. MUSIC GENRE RECOGNITION

us to group the descriptors in these three reduced models. Note also that the values for \bar{z} show a small deviation, showing that the descriptor separability is quite stable in the $\omega\delta$ -space.

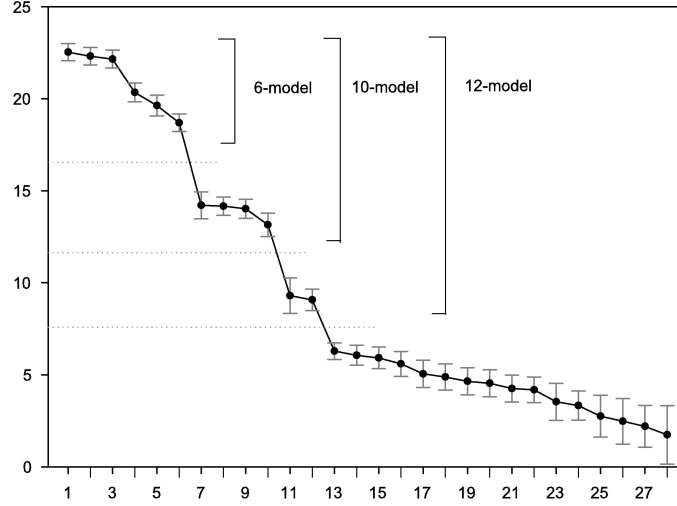


Figure 4.20: Values for \bar{z} for each descriptor as a function of their order numbers. The relative deviations for \bar{z} in all the experiments are also displayed. The biggest gaps for \bar{z} and the models are outlined.

It is interesting to remark that at least one descriptor from each category of those defined in section 4.3.1 were selected for a reduced model. The best represented categories were pitches and intervals, suggesting that the pitches of the notes and the relation among them are the most influent features for this problem. From the statistical point of view, standard deviations were the most important features, since five from six possible ones were selected.

4.3.3 Experiments and results on music genre classification

The $\omega\delta$ -space framework

The melodic segment parameter space has been established as follows:

$$\omega = 1, \dots, 100 \quad (4.3)$$

and, for each ω

4.3. SUPERVISED MUSIC GENRE RECOGNITION

$$\delta = \begin{cases} 1, \dots, \omega & \text{if } \omega \leq 50 \\ 1, \dots, 20 & \text{otherwise} \end{cases} \quad (4.4)$$

The range for δ when $\omega > 50$ has been limited to 20 due to the very few number of samples obtained with large δ values for this ω range. This setup produces a total of 2275 points $\langle \omega, \delta \rangle$ in the $\omega\delta$ -space. A number of experiments have been made for each of these points: one with each classifier (Bayes, NN) for each of the four description models discussed in section 4.3.2. Therefore, 12 different experiments for each $\omega\delta$ -point have been made, denoted by $(\omega, \delta, \mu, \gamma)$, where $\mu \in \{6, 10, 12, 28\}$ is the description model and $\gamma \in \{\text{Bayes}, \text{NN}\}$ the classifier used.

In order to obtain reliable results, a ten-fold crossvalidation scheme has been carried out for each of the $(\omega, \delta, \mu, \gamma)$ experiments, making 10 sub-experiments with about 10% of samples saved for test in each sub-experiment. The success rate for each $(\omega, \delta, \mu, \gamma)$ experiment is averaged for the 10 sub-experiments.

The partitions were made at the MIDI file level, to make sure that training and test sets do not share segments from any common melody. Also the partitions were made in such a way that the relative number of measures for both genres were equal to those for the whole training set. This permits us to estimate the prior probabilities for both genres once and then use them for all the sub-experiments. Once the partitions have been made, segments of ω measures are extracted from the melody tracks, and labeled training and test datasets containing μ -dimensional descriptor vectors are constructed.

To summarise, 27 300 experiments consisting of 10 sub-experiments each, have been carried out. The maximum number of segments extracted is $s = 9339$ for the $\omega\delta$ -point $\langle 3, 1 \rangle$. The maximum for s is not located at $\langle 1, 1 \rangle$ as expected, due to the fact that segments not containing at least two notes are discarded. The minimum is $s = 203$ for $\langle 100, 20 \rangle$. The average number of segments in the whole $\omega\delta$ -space is 906. The average proportion of jazz segments is 36% of the total number of segments, with a standard deviation of about 4%. This is a consequence of the classical MIDI files having a greater length in average than jazz files, although there are less classical files than jazz files.

Each $(\omega, \delta, \mu, \gamma)$ experiment has an average success rate, obtained from the crossvalidation scheme discussed above. The results presented from herein are based on those rates.

CHAPTER 4. MUSIC GENRE RECOGNITION

Bayes classifier

For one sub-experiment in a point in the $\omega\delta$ -space, all the parameters needed to train the Bayesian classifier are estimated from the particular training set, except for the priors of each genre, that are estimated from the whole set, as explained above.

Figure 4.21 shows the classification results for the Bayesian classifier over the $\omega\delta$ -space for the 12-descriptor model. This was one of the best combination of model and classifier (89.5% of success) in average for all the experiments. The best results for this classifier were found around $\langle 58, 1 \rangle$, where a 93.2% average success was achieved.

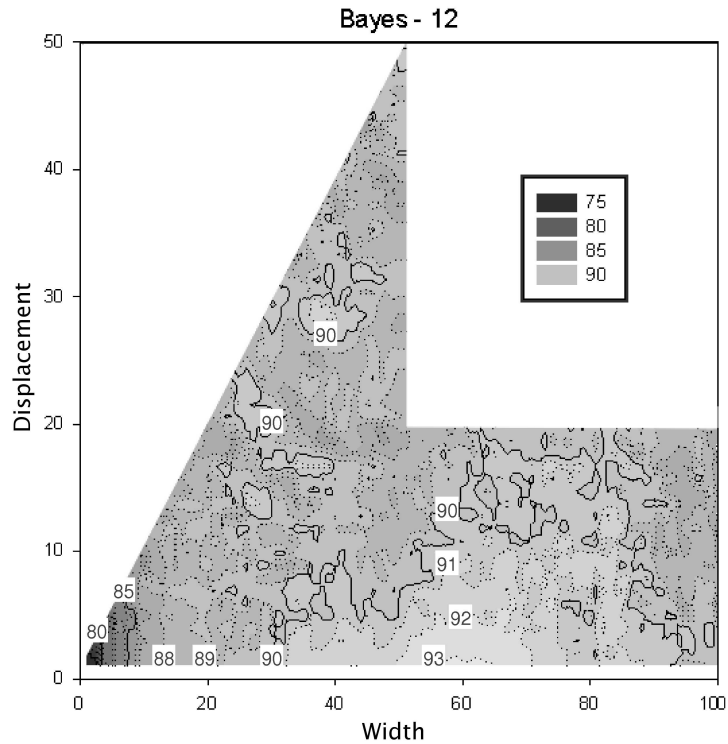


Figure 4.21: Illustration of the recognition percentage in the $\omega\delta$ -space for the Bayesian classifier with the 12-descriptor model. Numbers on top of level curves indicate the recognition percentage at places on the curve. The best results (around 93.2%) are found in the lighter area, with large widths and small displacements.

The best results for genre classification were expected to be found for moderate ω values, where enough musical events to calculate reliable statistical descriptors are contained in a segment, while musical events

4.3. SUPERVISED MUSIC GENRE RECOGNITION

located in other parts of the melody are not mixed in a single segment. But the best results are generally obtained with a combination of large ω values and small δ . Experiments for $\omega = \infty$ (taking the whole melody as a single segment) are discussed in section 4.3.3.

The worst results occurred for small ω , due to the few musical events at hand when extracting a statistical description for such a small segment, leading to non-reliable descriptors for the training samples.

All the three reduced models outperformed the 28-descriptor model (see Fig. 4.22 for a comparison between models for $\delta = 1$), except for $\omega \in [20, 30]$, where the 28-descriptor model obtains similar results for small values of δ . For some reason, still unknown, the particular combination of ω and δ values in this range results in a distribution of descriptor values in the training sets that favours this classifier.

The overall best result (95.5% of average success) for the Bayesian classifier has been obtained with the 10-descriptor model in the point $\langle 98, 1 \rangle$. See Table 4.11 for a summary of best results – indices represent the $\langle \omega, \delta \rangle$ values for which the best success rates were obtained. About 5% of the sub-experiments (4 556 out of 91 000) for all models yielded a 100% classification success.

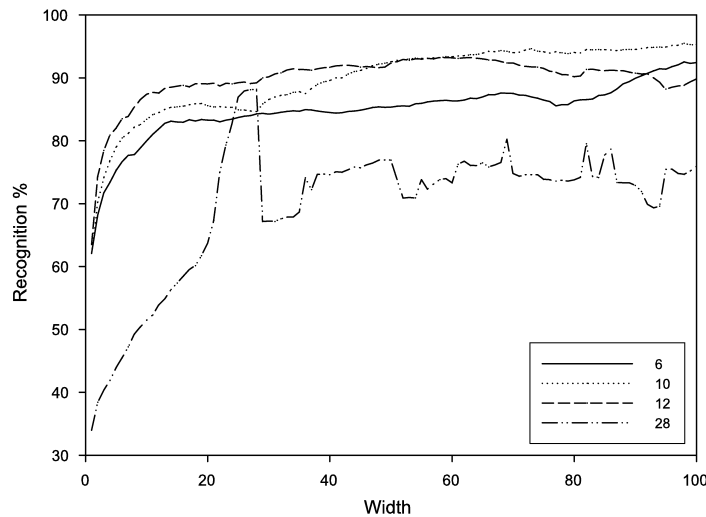


Figure 4.22: Bayes recognition results for the different models versus the window width, with a fixed $\delta = 1$.

k-NN classifier

Before performing the main experiments for this classifier, a study of the evolution of the classification as a function of k has been designed, in order to test the influence of this parameter in the classification task. The results are displayed in Fig. 4.23. Recognition percentage is averaged for all $\langle \omega, 1 \rangle$ points. Note that there is almost no variation in the recognition rate as k increases, except a small improvement for the 6-descriptor model. Thus, the simplest classifier was selected: $k = 1$, to avoid unnecessary time consumption due to the very large number of experiments to be performed.

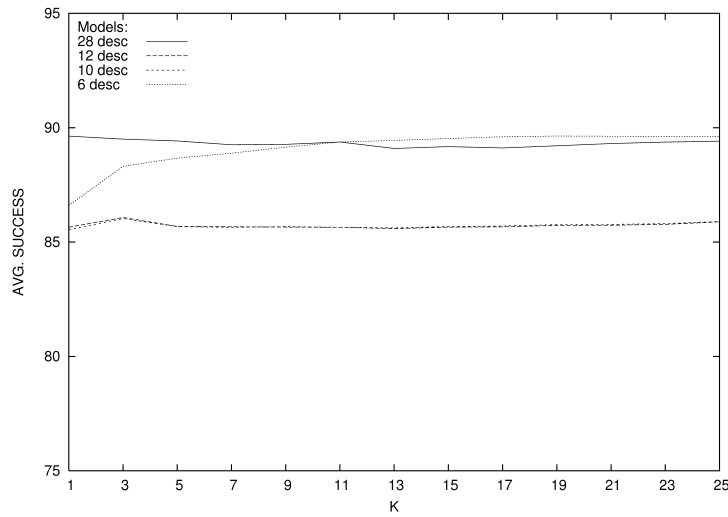


Figure 4.23: Evolution of k -NN recognition for the different models against values of k .

Once the classifier has been set, the results for the different models were obtained and are displayed in Fig. 4.24 for $\delta = 1$. All models performed comparatively for $\omega \leq 35$. For $\omega > 35$, the 28-descriptor model begins to perform better than the reduced models. Its relatively high dimensionality and a greater dispersion in the samples (the larger the ω , the higher the probability of different musical parts to be contained in the same segment) causes larger distances among the samples, making the classification task easier for the k -NN.

The best results (96.4%) were obtained for the point $\langle 95, 13 \rangle$ with the 28-descriptor model. The best results for all the models have been consistently obtained with very large segment lengths (see Table 4.11). The percentage of perfect (100%) classification sub-experiments amounts to 18.7% (17 060 out of 91 000).

4.3. SUPERVISED MUSIC GENRE RECOGNITION

model	Bayes	NN
6	93.2 _(100,2)	94.0 _(91,16)
10	95.5 _(98,1)	92.6 _(99,19)
12	93.2 _(58,1)	92.6 _(98,19)
28	89.5 _(41,33)	96.4 _(95,13)

Table 4.11: Best success rates

For the whole $\omega\delta$ -space, the NN classifier obtained an 89.2% in average with the 28-descriptor model, while the other models yielded similar rates, around 87%. The behavior of the 10- and 12-descriptor models was almost identical over the parameter space (Fig. 4.24) and for the different tested values for k (Fig. 4.23).

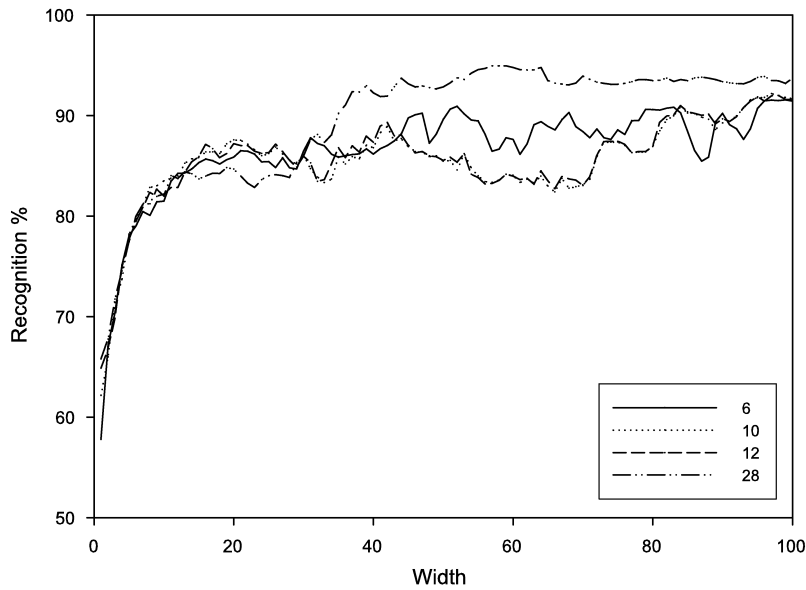


Figure 4.24: NN recognition results for the different models versus the window width, with a fixed $\delta = 1$.

Whole melody segment classification

The good results obtained for large ω called our attention to the question of how good would be the results of classifying whole melodies, instead of fragments, as presented so far. The first problem is the small number of samples available this way (110 samples for training and test). The results

CHAPTER 4. MUSIC GENRE RECOGNITION

model	Bayes	NN
6	84.2 \pm 2.0	87.4 \pm 2.9
10	88.5 \pm 3.2	86.9 \pm 2.5
12	89.5 \pm 1.7	87.1 \pm 2.5
28	71.1 \pm 6.3	89.2 \pm 4.5

Table 4.12: Averages and standard deviations of success rates

model	Bayesian	NN
6	88.0	87.0
10	91.0	88.0
12	91.0	88.0
28	79.0	93.0

Table 4.13: Average success rates for whole melody segment length ($\omega = \infty$)

of these experiments are displayed in Table 4.13. The same 10-fold cross-validation scheme described in section 4.3.3 was used here. The results are comparable or even better than the average in the $\omega\delta$ -space for both classification paradigms.

In spite of this good behavior for Bayes and k -NN, this approach has a number of disadvantages. Training is always more difficult due to the smaller number of samples. The classification cannot be performed *on-line* in a real-time system, because all the piece is needed in order to take the decision. There are also improvements to the presented methodology, like cooperative decisions using different segment classifications that can not be applied to the complete melody approach.

Results comparison

Bayesian and NN classifier performed comparatively. There were, in general, lower differences in average recognition percentages between NN models than those found with the Bayesian classifier (see Table 4.12), probably due to its non-parametric nature.

An ANOVA test with Bonferroni procedure for multiple comparison statistics (Hancock and Klockars, 1996) was used to determine which combination of model and classifier gave the best classification results in average. According to this test, with the number of experiments performed, the required difference between any two recognition rates in Table 4.12 must be at least 0.45123 in order to be considered statistically different at the

4.3. SUPERVISED MUSIC GENRE RECOGNITION

95% confidence level. Thus, it can be stated that Bayes classifier with 12-descriptor model and NN classifier with 28-descriptor model perform comparatively well, and both outperform the rest of classifier and model combinations. The Bayes classifier has the advantage of using a reduced size description model.

In (Pérez-Sancho et al., 2004), the JvC1 dataset was also used to perform genre recognition from whole melodies, using several text categorization algorithms. In particular, a naive Bayes classifier with several multivariate Bernoulli and multinomial models are applied to binary vectors indicating the presence or absence of n -length words (sequences of n notes) in a melody. The work reported around 93% of success as the best performance. This is roughly the same best result reported here for the whole melody, although it is outperformed by the window classification results.

Results for the $\omega\delta$ -space are hardly comparable with those by other authors, due to our use of segments instead of complete melodies, and mainly due to the different datasets put under study by different authors. Nevertheless a comparison attempt can be made with the results found in (Tzanetakis et al., 2003) for pair-wise genre classification. The authors use information from all the tracks in the MIDI files except tracks playing on the percussion channel. In that work, a 94% accuracy for Irish Folk music and Jazz identification is reported as the best result. Unfortunately, they did not use Classical samples. This accuracy percentage is similar to our results with whole melody length segments and the NN classifier (93%). A study on the classification accuracy as a function of the input data length is also reported, showing a behavior similar to the one reported here: classification accuracy using statistical information reaches its maximum for larger segment lengths, as they reported a maximum accuracy for five classes with 4 minute segment length. Our best results were obtained for $\omega > 90$ bars (see Table 4.11).

4.3.4 A graphical interface for music genre recognition

An experimental graphical user interface has been developed to facilitate working on the problem of music genre recognition. The main motivation for such a tool is to allow investigate why classification errors occur. The interface allows to select a model $(\omega, \delta, \mu, \gamma)$ for classifying selected tracks from MIDI files. The classification of each extracted window is shown in a row and encoded by colors. Each window content can be played individually and its description visualized. A more detailed description of the interface can be found in appendix A.3. A snapshot of the application can be seen in Figure 4.25.

CHAPTER 4. MUSIC GENRE RECOGNITION

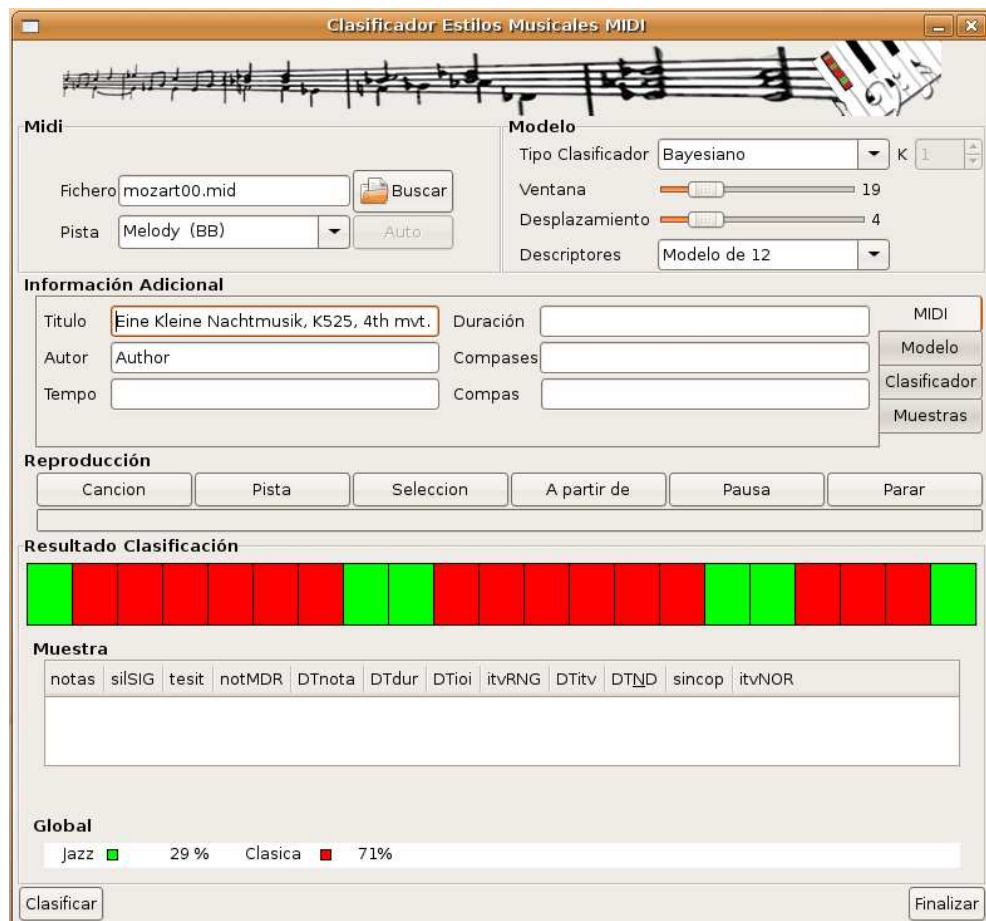


Figure 4.25: Snapshot of a GUI application for music genre recognition.

4.4 Classifier ensembles for music genre recognition

Work done so far in genre recognition has taken our research to the use of classifier ensembles, trying to improve the performance of single models. First, section 4.4.1 demonstrates the feasibility of a type of ensembles known as voting methods, described in 2.4.1. This is a *late fusion* technique that combine classifier outcomes using several voting strategies in order to obtain a *consensus* output. Their performance on a series of classification benchmarking tasks is shown to be precise and robust, revealing voting ensembles, and in particular the voting rules proposed in this work, as a good choice for classification tasks.

Once the classifier ensembles have been thoroughly tested, section 4.4.2 discusses the application of such technique to music genre recognition, to corroborate that conclusions drawn out in the previous section remain sound in this application domain. Base classifiers are trained using statistical description models of melodies, as in previous sections, in order to improve both the accuracy and robustness of single classifier systems in the genre recognition task.

4.4.1 Evaluation of voting schemes

In (Moreno-Seco et al., 2006), the performance of several classifier fusion methods, some of them proposed in the context of this research, is compared. Results from this benchmarking experiments are presented and discussed in this section. The voting schemes presented in section 2.4.1 have been tested on 19 benchmarking data sets from the UCI repository (Blake and Merz, 1998). The performance of the *RSWV*, *BWWV*, and *QBWWV* methods is assessed, using the *PV*, *SWV*, and *WMV* methods as reference voting schemes. The ensembles are built using five conceptually different base classifiers: the naïve Bayesian classifier (NB), support vector machine (SVM), multi-layer perceptron (MLP), 3-nearest-neighbour classifier (3-NN) and the C4.5 decision tree. They have been trained using the default parameters established for them in the WEKA framework.

Two methods have been used to train the classifiers, and the ensembles: first, for the UCI repository data sets, a total of 50 pairs of train/test sets were generated, using 10 random seeds for generating 5 cross-validation pairs (with approximately an 80% of the data for training, and the rest for testing). The base classifiers have been run 50 times with different train and test sets from the same data (each data sample has been classified 10 times). The error rate of the classifier has been estimated by counting the total number

CHAPTER 4. MUSIC GENRE RECOGNITION

of errors over the 50 experiments, divided by the total number of samples classified (that is 10 times the size of the data set).

Once the ensembles have been trained with the UCI project data sets, a validation experiment has been run, using a new random seed for generating another 5 pairs of train/test sets. The base classifiers have also been run with the validation data, in order to obtain a reference.

Table 4.14 presents the error rates of the validation experiments for the UCI datasets, with the best results for each data set emphasized in boldface. The result for the best single classifier classifier is showed as a reference.

4.4. ENSEMBLES FOR MUSIC GENRE RECOGNITION

DATA SET	PV	SWV	RSWV	BWWV	QBWWV	WMV	BEST
australian	13.04	13.04	13.04	13.62	14.64	13.04	14.64 (SVM)
balance	12.64	11.36	11.36	10.56	10.56	11.20	8.80 (MLP)
cancer	3.37	3.37	3.37	3.37	3.37	3.37	3.22 (SVM)
diabetes	23.18	23.18	23.18	23.44	22.66	23.18	22.66 (SVM)
german	24.30	24.30	24.30	23.5	23.70	24.30	23.70 (SVM)
glass	32.71	30.84	30.84	28.51	29.91	28.51	32.24 (MLP)
heart	15.93	15.93	15.93	15.19	15.19	15.93	14.07 (NB)
ionosphere	9.12	9.12	9.12	11.11	11.11	9.12	9.40 (MLP)
liver	36.81	36.81	35.36	33.62	31.88	35.36	31.88 (MLP)
monkey1	3.60	3.60	0	0	0	0	0 (MLP)
phoneme	16.78	16.78	16.78	13.53	12.31	16.78	12.31 (3-NN)
segmen	3.51	3.07	3.07	2.55	2.55	3.07	3.77 (C4.5)
sonar	24.04	24.04	24.04	23.08	23.08	24.04	22.12 (MLP)
vehicle	21.75	21.04	21.04	20.33	20.33	20.33	18.91 (MLP)
vote	4.37	4.37	4.37	3.69	4.14	4.37	4.14 (C4.5)
vowel	14.02	11.74	11.74	5.87	4.92	5.68	4.92 (3-NN)
waveform21	14.74	14.70	14.70	13.36	13.3	14.70	13.30 (SVM)
waveform40	14.50	14.50	14.50	13.96	13.74	14.50	13.74 (SVM)
wine	1.69	1.69	1.69	2.25	2.25	1.69	1.69 (NB)

Table 4.14: Error rates (in %) of the different ensembles with the UCI/Statlog data sets, together with the result of the best individual classifier (BEST) column. The winning classifications schemes in terms of accuracy for each data set have been highlighted.

CHAPTER 4. MUSIC GENRE RECOGNITION

To summarize the results, the ensembles perform equal or better than the best single model in 14 out of 19 data sets. The approaches based on scaling the weights to a range established by the best and the worst classifiers (*BWWV* and *QBWWV*) have shown the best classification accuracy on 11 out of 19 datasets from the UCI repository. Note that the *QBWWV* voting method has performed the best overall, being 8 times in the set of winner schemes. Taking only single models, every classification scheme has been at least twice the winning choice, so no one is consistently superior on these datasets. No analytic methods exist that are able to decide which is the best classifier to be used according to the data. On the other hand, the trained ensemble is essentially the same system on a per corpus basis, where single model decisions are “averaged”, thus looking like an *a priori* better choice, reducing the risk of selecting the wrong classifier for a given problem.

Following the performance results in table 4.14, ensembles are not consistently better than single models over all datasets. However, note that the ensemble is made up of only five trained models. In order to improve the ensemble performance, a greater and more diverse number of models would be desirable. Work in this direction is presented in section 4.5.4.

4.4.2 Evaluation on music genre recognition

Given the results in the previous section, the question arise whether the ensemble is a good choice for a symbolic music genre recognition task. This section presents a set of experiments performed to find evidence in this direction. The JvC1+2⁷ corpus is used here. Only melody tracks are used as input data. They are described using the full descriptor model described in section 4.3.1 (28 features).

From the set of MIDI files two datasets have been built. The first one composed of 150 samples, one sample per melody track. This is equivalent to consider a sliding window width $\omega = \infty$. The second one is built applying a $\langle 50, 1 \rangle$ sliding window procedure to the corpus, so each sample describes a 50-bar melody fragment. This second dataset is made up of 7125 samples.

The experiments with the JvC data sets have been carried out using a train, test, and validation scheme. Random partitions are not advisable since for the $\omega = 50$ case attention has to be paid to samples belonging to the same melody do not appear in both training and test or validation. Each data set has been split into five partitions (keeping in the same partition those samples belonging to the same MIDI file). Three of them have been used for training, one for test, and the remaining one for validation. The experiment has been

⁷It is abbreviated as JvC in what follows.

4.4. ENSEMBLES FOR MUSIC GENRE RECOGNITION

repeated five times, rotating the partitions. The results of the validation presented in table 4.15 are average error rates over the experiments.

ENSEMBLE/CLASSIFIER	DATA SET	
	JvC, $\omega = \infty$	JvC, $\langle 50, 1 \rangle$
<i>PV</i>	7.33	9.28
<i>SWV</i>	7.33	9.28
<i>RSWV</i>	7.33	9.16
<i>BWWV</i>	6.00	6.31
<i>QBWWV</i>	6.00	8.29
<i>WMV</i>	6.00	9.46
3-NN	6.00	11.80
MLP	8.00	13.30
SVM	10.67	11.08
C4.5	13.33	15.66
NB	16.00	15.56

Table 4.15: Average error rates (in %) of the different ensembles with the JvC data sets, together with the results of the base classifiers.

The results for $\omega = \infty$ show that even when the base classifiers are not as good as the best one (the 3-NN classifier), the ensembles still perform comparably to it. For the $\omega = 50$ data set, the ensembles perform much better than any base classifier, specially the BWWV, which obtains an error rate 4.5% below the rate of the best classifier (SVM). The results shown in table 4.15 confirm that the ensembles' performance is better in the general case. This is taken as an evidence that classifier ensembles of this kind are a good choice for a symbolic music genre recognition task. The BWWV performed best on the two music genre recognition experiments.

The use of voting ensembles has proven feasible to improve single classifier results on music genre recognition. In this section, all base models were trained using the same set of features. However, this is not mandatory. Each base classifier can be trained on a different set of features describing the objects in a given corpus. The next section presents some experiments on this type of information fusion, or multimodal pattern recognition, in the classification context of music genre recognition. Section 4.5 discusses an extensive set of experiments on information fusion from the audio and symbolic domains, again in the context of music genre recognition.

4.4.3 Fusion of global and local symbolic music information

Experiments on late fusion (sec. 2.3.2) of musical information are presented in this section. The *JvC* corpus is used as input data. Two kind of features are extracted from whole melody tracks (no sliding window segmentation is applied) in the corpus: *global* statistical features, as described in section 4.3.1, and *local* statistical features (Pérez-Sancho, 2009b). The latter describe melodic content in terms of strings of symbols corresponding to melody subsequences or *n*-words. They are briefly described below.

Given these two set of features, a multiple viewpoint classifier ensemble is built, whit some base models trained on global features while others are trained on local ones. It is expected that the diversity of the ensemble will increase, as models based on different feature sets should, in general, make their wrong decisions on different objects.

n-word based melody representation and classification

The *n*-word based models make use of text categorization methods to describe melodic content. The technique encodes note sequences as character strings, therefore converting a melody in a text to be categorized. Such a sequence of *n* consecutive notes is called an *n*-word. All possible *n*-words in a melody are extracted, except those containing a silence lasting four or more beats. The encoding for *n*-words used here has been derived from the method proposed in (Doraisamy and Rüger, 2003). This method generates *n*-words by encoding pitch interval and duration information. For each *n*-note sequence, all pitch intervals and duration ratios (inter-onset interval ratio) are calculated using Eqs. (4.5) and (4.6) respectively:

$$I_i = \text{Pitch}_{i+1} - \text{Pitch}_i \quad (i = 1, \dots, n-1) \quad (4.5)$$

$$R_i = \frac{\text{Onset}_{i+2} - \text{Onset}_{i+1}}{\text{Onset}_{i+1} - \text{Onset}_i} \quad (i = 1, \dots, n-2) \quad (4.6)$$

and each *n*-word is defined as a string of symbols:

$$[I_1 \ R_1 \ \dots \ I_{n-2} \ R_{n-2} \ I_{n-1} \ R_{n-1}] \quad (4.7)$$

where the pitch intervals and duration ratios have been mapped into alphanumeric characters (see (Pérez-Sancho et al., 2004) for details).

This method represents a musical piece as a vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{i|\mathcal{V}|})$, where each component represents the presence of the word w_t in the melody, being $|\mathcal{V}|$ the size of the vocabulary, that is, the total number of different *n*-words extracted from the corpus.

4.4. ENSEMBLES FOR MUSIC GENRE RECOGNITION

A common practice in text classification is to reduce the dimensionality of those vectors (usually very high) by selecting the words that contribute most to discriminate the class of a document (a melody here). The *average mutual information* measure (AMI) (Cover and Thomas, 1991) is used to rank the words. This measure gives a high value to those words that appear often in melodies of one genre and are seldom found in melodies of the other genres. The n -words are sorted using this value, so only information about the first N words are provided to the classifiers. The Naïve Bayes classifier is commonly used with this kind of representation (Domingos and Pazzani, 1997). The class-conditional probability of a melody is given by the probability distribution of note sequences (n -words) in a genre.

Two different distribution models have been used. The first one is a Multivariate Bernoulli (MB) model, where the components of a sample vector \mathbf{x}_i are $x_{it} \in \{0, 1\}$. Each class follows a multivariate Bernoulli distribution where the parameters to be learned from the training set are the class-conditional probability of each word in the vocabulary. The second choice is a Multinomial (MN) model, where components $x_{it} \in \{0, 1, \dots, |\mathbf{x}_i|\}$, being $|\mathbf{x}_i|$ the number of n -words extracted from melody \mathbf{x}_i . Each component x_{it} is the number of occurrences of word w_t in the melody. The probability that a melody has been generated from a genre c_j is a multivariate multinomial distribution, where the melody length is assumed to be class-independent. Both MB and MN distributions have proven to achieve quite good results in text classification (McCallum and Nigam, 1998).

Results

A set of base classifiers has been built by combining the different description models and classification paradigms: four k -nearest neighbors, using $k = 7$, with the different feature combinations discussed in section 4.3.2, and four Bayesian classifiers with the same feature combinations, and two naïve Bayes classifiers using Bernoulli and Multinomial probability distributions. For the latter, a vocabulary size of 100 and 170 2-words have been used respectively, according to their AMI values. This makes a total of ten classifiers for building ensembles. Table 4.16 presents the estimated accuracy of the individual classifiers, α_k , obtained using a leave-one-out validation method on the training set.

Six voting ensembles have been constructed using the voting methods from sec. 2.4.1. The decisions of the ensembles are summarised in Table 4.17 (*# errors all* column), and graphically depicted in Fig. 4.26 against the best individual classifier score. Ties in the ensemble decisions are considered as errors. Note that the ensemble's performance using the *WMV* strategy

CHAPTER 4. MUSIC GENRE RECOGNITION

Classifier	Statistical model	# features	# errors	α_k
7-NN	Global	6	7	0.936
	Global	10	12	0.891
	Global	12	12	0.891
	Global	28	3	0.973
Bayes	Global	6	10	0.909
	Global	10	9	0.918
	Global	12	10	0.909
	Global	28	22	0.746
Naive Bayes	Bernoulli	$N = 100$	8	0.923
	Multinomial	$N = 170$	16	0.855

Table 4.16: Working parameters and accuracy of the different classifiers selected.

Voting method	# errors all	% OK	# errors all-but-best	% OK
<i>PV</i>	6	94.5	9	91.8
<i>SWV</i>	3	97.3	6	94.5
<i>RSWV</i>	3	97.3	6	94.5
<i>BWWV</i>	3	97.3	4	96.4
<i>QBWWV</i>	3	97.3	4	96.4
<i>WMV</i>	2	98.2	6	94.5

Table 4.17: Ensemble's performance.

improves the behavior of the best of the individual classifiers: just two errors against the three errors made by 7-nearest neighbour classifier based on the whole set of global descriptors. Also, it is worth mentioning that for the *PV* method, 4 out of 6 errors are due to ties (5 out of 9 when removing the best classifier), while there are no ties with other voting strategies. The probability of decision ties in the ensemble is lowered by using weighed voting rules.

The question arises of how sensitive is the ensemble success to its own structure. In addition, is it worth to build an ensemble for avoiding just one error? The answer for both questions could be approached removing from the ensemble the best of the classifiers and analysing how much the performance is degraded. Thus, the best base model was dropped from the ensemble, and the new results were those also shown in Table 4.17 (*# errors all-but-best* column).

Note how, although the results are not as good as earlier, some ensembles maintain a high standard of precision, with just 4 errors. This clearly

4.4. ENSEMBLES FOR MUSIC GENRE RECOGNITION

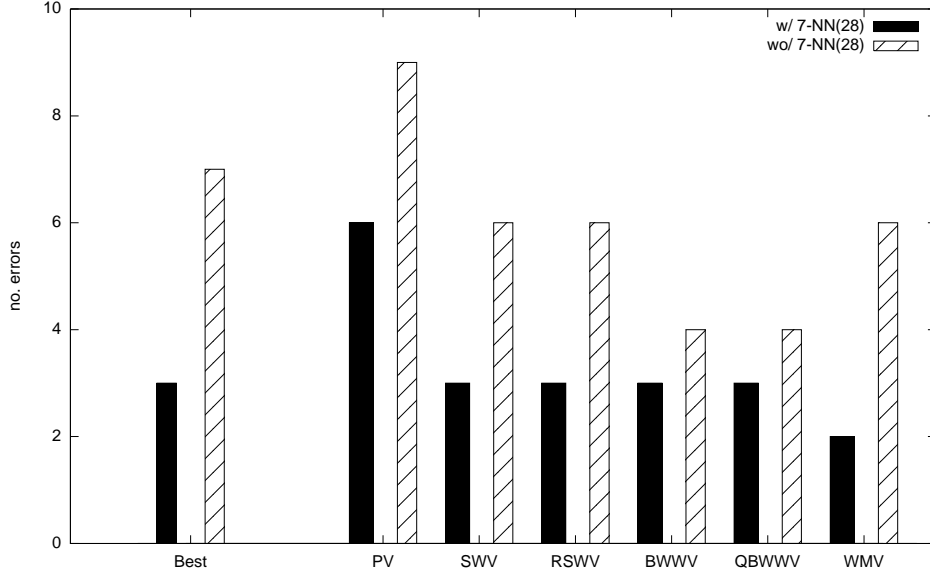


Figure 4.26: Number of errors made by the different ensembles (with the performance of the best classifier on the left). Bars in black correspond to the ensemble of all the classifiers and pattern-filled bars to the ensemble of all but the best.

improves the performance of the current best classifier (7 errors), so the ensemble seems quite robust and performs well, specially with the best-worst strategies (*BWWV* and *QBWWV*) proposed in this work.

As this probably has to do with the diversity of the ensemble, the classifiers' diversity has been assessed by the *inter-rater agreement*, or κ measure (sec. 2.4.2). The goal is to find which are the most 'disagreeing' classifiers among those in the ensembles. The diversity measure evaluate classifiers in pairs. The less the κ value of a pair, the more their components disagree. Given a list of all pairs sorted in ascending κ values, a variant of the *Borda count* ranking aggregation method (sec. 2.8.3) is used to obtain a ranking of single classifiers:

Given $D = \{D_1, \dots, D_j, \dots, D_K\}$, the set of K classifiers in the ensemble, define

$$P = \{\{D_i, D_j\} \mid 1 \leq i, j \leq K, i \neq j\} \quad (4.8)$$

as a subset of all unordered pairs of classifiers, $p_l = \{D_i, D_j\} \in P$, and $\kappa(p_l)$ as the inter-rater-agreement value of p_l . P is a sorted set such that,

$$\kappa(p_l) \leq \kappa(p_m), \forall l < m \quad (4.9)$$

The Borda count ranking value of a classifier D_i , is computed as

$$r_i = \sum_{l=1}^{|P|} \delta(D_i, p_l)(|P| - l + 1) \quad (4.10)$$

where

$$\delta(D_i, p_l) = \begin{cases} 1 & \text{if } D_i \in p_l \\ 0 & \text{otherwise} \end{cases} \quad (4.11)$$

This assigns the greater ranking value to classifiers in p_1 , and the lower value to classifiers in $p_{|P|}$. The results of this ranking procedure are shown in Table 4.18.

Classification paradigm	Feature selection	r_i
Bayes	28	367
Naive Bayes Multinomial	$ \mathcal{V} = 170$	298
Naive Bayes Bernoulli	$ \mathcal{V} = 100$	207
Bayes	12	195
7-NN	10	193
7-NN	12	185
Bayes	6	181
Bayes	10	167
7-NN	6	154
7-NN	28	123

Table 4.18: Classifiers ranked by diversity.

Interestingly enough, the best classifier (7-NN with 28 features) is the last one in the ranking. That is, it is the one that less disagrees with the rest of the ensemble. This could explain the fact that removing it from the ensemble doesn't worsen too much its performance, at least with the best-worst scaling voting methods. Another interesting fact from this ranking is that the two classifiers based on local features are in top positions. This supports the fact that training models on different feature spaces provides a more diverse ensemble of classifiers, which is a premise for a successful performance.

4.5 Fusion of musical information from audio and symbolic domains

In 2007 a fruitful research collaboration started between our team and the *Information & Software Engineering Group* of the *Vienna University of Technology*, led by Andreas Rauber⁸. This research opened the application field of symbolic music classification to the domain of audio musical data through its symbolic representation, obtained by means of automatic transcription systems. The experiments performed during such collaboration, and the results they provided (Lidy et al., 2007, 2008, 2009, 2010a; Mayer et al., 2010) are discussed in this section.

The corpora used in that research was made up of audio recordings. They are shown in Table 4.19, and discussed more in depth in section 2.8.4. Given that the symbolic feature extraction application is devised to deal with symbolic formats only, notably MIDI files, it was necessary to incorporate an audio music transcription stage. The transcription system presented in (Pertusa, 2010) was used for this task. It is briefly described in section 4.5.2.

dataset	files	genres	file length	ref.
9GDB	856	9	full	(Perez-Sancho et al., 2009)
GTZAN	1000	10	30 sec	(Tzanetakis, 2002)
ISMIRgenre	1458	6	full	(Cano et al., 2006)
ISMIRrhythm	698	8	30 sec	(Cano et al., 2006)
LatinMusic	3225	10	full	(Silla Jr. et al., 2008)
Africa	1024	var.	full	(Cornelis et al., 2005)
MIREX2007	7000	10	30 sec	(MIREX, 2007b)

Table 4.19: Audio datasets.

State-of-the art transcription systems have limited capabilities for transcribing multi-part music pieces, as source separation in music signals is still a largely open problem. Most current transcription systems can do reasonably good with polyphonic monotimbral recordings, as it is the case of the system used here. However, the audio music genre recognition task is defined as to predict the music genre of, in general, polyphonic and *multitimbral* recordings. So, the approach is to use a polyphonic, monotimbral transcriber to transcribe polyphonic, multitimbral music, assuming that:

⁸Funding was provided by national state agencies from both countries, for fostering research on the combination of audio and symbolic music description features for genre classification.

CHAPTER 4. MUSIC GENRE RECOGNITION

- The symbolic transcription will be monotimbral.
- The transcription will contain transcription errors.

The transcription errors are expected to be many, as the result of different circumstances: mixed timbres playing the same notes, or in the same pitch range, the presence of percussion (indefinite pitch) instruments, special effects, post-production artifacts like equalization, reverberation, compression, etc. It is important to mention that a better transcription (i.e., one with less such errors), do not necessarily imply better results in this context. The systematic errors of transcription can be helpful, in some degree, for such tasks like genre recognition. For instance, when analyzing a musical excerpt containing a distorted guitar sound, many false positive notes usually appear due to the strong and unstable harmonic content. This is an unwanted effect for the evaluation of a transcription system, but many such notes can also be an indication about the presence of strong timbres or unpitched content, which are more common in rock than in classical music, for example. Given that an accurate transcription is not the ultimate goal here, the assumption made in this research collaboration is the following:

An inaccurate but systematic monotimbral symbolic music transcription of a multitimbral audio music signal still conveys information about genre.

The symbolic transcription obtained this way consists of a MIDI file with a single polyphonic track, containing only note events extracted from the whole music signal. How much of the genre information conveyed by the audio has gone into the symbolic transcription is something hard to evaluate, and definitively out of the scope of this research. For most recordings, the transcription is barely recognizable for human ears, and definitively not what one would expect from an accurate multi-part transcriber. However, if the assumption above holds, then symbolic descriptors can be extracted from that track and complement audio descriptors extracted directly from the audio signal, in order to predict the genre of a music piece. Such combination of audio and symbolic descriptors extracted from the same source was expected to improve previous genre classification results by audio descriptors alone.

Two approaches have been taken to combine the feature sets at hand: *early fusion* of features (sec. 4.5.3) and *late fusion* of predictions in classifier ensembles (sec. 4.5.4 and 4.5.5). The first one integrates multiple feature sets in a single feature superset that describe the input signal. Then, a single classifier is build on this combined feature domain. Rather than combining

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

features, the second approach builds an ensemble of classifiers, each of them on a (possibly) different feature space, and then combines the predictions from single models to output a *consensus* decision. See section 2.3 for more details on these two information fusion techniques.

4.5.1 Audio and symbolic descriptors

Several sets of both audio and symbolic descriptors have been used to describe audio music. They are shown in Tables 4.20 and 4.21. Symbolic descriptors are gathered from those used in melody part selection (section 3.1.2) and previous music genre recognition experiments (section 4.3.1). Some of these features have been added in late fusion experiments discussed below. They are the *average notes per beat* descriptor, and the *chord type frequencies* feature set. The former is a polyphony-related descriptor, as it measures the average number of notes simultaneously active during a beat. In order to capture more detailed information about the polyphonic structure of the transcription, chord type frequency features are extracted. A chord sequence is extracted using the algorithm from (Pardo and Birmingham, 2002), and subsequently analyzed. Nine different types of chords are detected: major triad, major 7th, dominant 7th, dominant suspended 7th, dominant 7th (sharp 5th), dominant 7th (flat 5th), minor 7th, half diminished and fully diminished chords. The relative frequencies of these chords in a chord sequence are computed as symbolic features.

Audio Descriptors	# feats.	Set		
		A1	A2	A3
Rhythm Pattern (RP)	1440	•	•	•
Rhythm Histogram (RH)	60	•	•	•
Statistical Spectrum Descriptors (SSD)	168	•	•	•
Temporal Rhythm Histogram (TRH)	420		•	•
Temporal Statistical Spectrum Descriptors (TSSD)	1176		•	•
Modulation Frequency Variance Descriptors (MVD)	420		•	•
Onset Descriptors (OD)	11	•		
Relative Spectral Energy Matrix (RSEM)	112		•	
Template Descriptors (TD)	unknown		•	

Table 4.20: Audio feature sets used in fusion experiments

Audio descriptor sets are briefly explained below.

Rhythm Patterns (RP) This feature set is neither a mere description of rhythm nor does it represent plain pitch information. Rather, it

CHAPTER 4. MUSIC GENRE RECOGNITION

Category	Descriptors	Set		
		S1	S2	S3
Overall	Number of notes	•	•	•
	Number of significant silences	•	•	•
	Number of non-significant silences	•	•	•
	Track duration	•	•	•
	Occupation rate		•	•
	Polyphony rate		•	•
	Avg. notes per beat (*)			•
Pitch	Highest pitch	•	•	•
	Lowest pitch	•	•	•
	Pitch range		•	•
	Average pitch	•	•	•
	Average relative pitch	•	•	•
	Dev. pitch	•	•	•
	Pitch Normality	•	•	•
Note duration	Largest duration	•	•	•
	Smallest duration	•	•	•
	Duration range		•	•
	Avg. duration	•	•	•
	Avg. relative duration	•	•	•
	Dev. duration	•	•	•
	Duration normality	•	•	•
Silence duration	Largest duration	•	•	•
	Smallest duration	•	•	•
	Silence duration range		•	•
	Avg. duration	•	•	•
	Avg. relative duration	•	•	•
	Dev. duration	•	•	•
	Duration normality	•	•	•
Inter Onset Interval	Num. IOI	•	•	•
	Largest IOI	•	•	•
	Smallest IOI	•	•	•
	IOI duration range		•	•
	Avg. IOI	•	•	•
	Avg. relative IOI	•	•	•
	Dev. IOI	•	•	•
	IOI normality	•	•	•
Pitch interval	Num. distinct intervals	•	•	•
	Largest Interval		•	•
	Smallest interval		•	•
	Interval range			•
	Avg. interval		•	•
	Avg. relative interval		•	•
	Dev. interval		•	•
	Interval mode	•	•	•
	Interval normality		•	•
Non-diatonic degrees	Num. non-diatonic notes		•	•
	Highest non-diatonic degree	•	•	•
	Lowest non-diatonic degree	•	•	•
	Avg. non-diatonic degrees	•	•	•
	Avg. relative non-diatonic degrees	•	•	•
	Dev. non-diatonic degrees	•	•	•
	Non-diatonic degrees normality	•	•	•
Syncopation	Number of syncopes		•	•
Chord type frequencies (*)	maj. triad, maj. 7th, dom. 7th,			•
	dom. sus. 7th, dom. 7th (#5th),			•
	dom. 7th (b5th), min. 7th,			•
	half dim., fully dim.			•
Total descriptors		37	52	62

(*) descriptors added for fusion classification experiments.

Table 4.21: Symbolic feature sets used in fusion experiments

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

describes the modulation of the sensation of loudness for different bands, by means of a time-invariant frequency representation. They were introduced in (Rauber and Fröhlich, 2001), and enhanced by incorporating psycho-acoustic phenomena in (Rauber et al., 2002).

Rhythm Histogram (RH) The magnitudes of each modulation frequency bin of 24 critical bands are summed up, to form a histogram of “rhythmic energy” per modulation frequency. The histogram contains 60 bins which reflect modulation frequency between 0 and 10 Hz. These features are descriptors for general rhythmic patterns in an audio document, and were first introduced in (Lidy and Rauber, 2005).

Statistical Spectrum Descriptors (SSD) According to the occurrence of beats or other rhythmic variation of energy on a specific band, statistical measures are able to describe the audio content. The following statistical moments on the values of 24 critical bands: mean, median, variance, skewness, kurtosis, min- and max-value. These features were also introduced in (Lidy and Rauber, 2005),

Temporal features (TRH, TSSD) Feature sets are frequently computed on a per segment basis and do not incorporate time series aspects. TRH and TSSD features include a temporal dimension describing variations over time. For TRH, statistical measures (mean, median, variance, skewness, kurtosis, min and max) are computed over the individual RH extracted from segments in a piece of audio. Thus, change and variation of rhythmic aspects in time are captured. TSSD analogously capture timbral variations and changes over time in the spectrum on the critical frequency bands. Hence, a change of rhythmic patterns, instruments, voices, etc. over time is reflected by this feature set (Lidy et al., 2010b).

Modulation Frequency Variance Descriptors (MVD) This descriptor measures variations in the critical bands for a specific modulation frequency of the RP matrix, representing the amplitudes of 60 modulation frequencies on 24 critical bands. The MVD vector is computed by taking the same statistics as for TRH and TSSD, for each modulation frequency over the 24 bands (Lidy et al., 2010b).

Onset features An onset detection algorithm described in (Pertusa et al., 2005) has been used to complement audio features. As a result of the onset detection, the minimum, maximum, mean, median and standard deviation

CHAPTER 4. MUSIC GENRE RECOGNITION

of both onset energies and onset intervals have been extracted. Also, the average number of onsets per frame is computed.

Relative Spectral Energy Matrix (RSEM) This feature set contains a coarse binning of the amplitude and the power spectrum at 40, 120, 500, 2000, 6000, 11000 and 22050 Hz respectively, averaged over all frames. In addition to these simple features, two matrices are formed by dividing each of the resulting amplitude bins by each of the power bins and vice versa ([Lidy et al., 2009](#)).

Template descriptors An algorithm coming from the blind source separation domain, and described in ([Grecu, 2008](#)) was adapted for genre classification and related tasks. In the original setting, each instrument sound is represented by a frequency spectrum template. The sum of these templates at their respective onsets will ideally reconstruct the song, though this is not a perfect reconstruction. These templates are further processed to result in the template descriptor set. This set contains, for example, the mean onset amplitude, mean onset distance, mean overlap with the other templates (matrix), template count, etc.

4.5.2 Transcription system

The transcription system uses a multiple fundamental frequency (f_0) estimation method to convert the audio files to MIDI files. This is a joint estimation approach, evaluated in ([Pertusa and Iñesta, 2008b](#)), which experimentally obtained a high accuracy with a low computational cost. It is presented in great detail in ([Pertusa, 2010](#)). The version used here extends a previous work ([Pertusa and Iñesta, 2008a](#)) by adding information about neighboring frames to get a smooth temporal estimation. It does not separate instruments, therefore producing single track MIDI files without any timbral information. Rhythm detection is not considered, only note pitches, onsets, and durations are extracted.

The system has two main parameters to tune in order to improve its efficiency for the genre classification task: the maximum polyphony of the transcription, that is, the maximum number of notes to be transcribed at any given time, and the minimum duration of a note. The latter has been fixed to about 140 ms, to avoid very short note detections. The maximum polyphony parameter P has been investigated by setting up a number of preliminary late fusion experiments in music genre recognition. The GTZAN corpus was transcribed to MIDI setting P to values from 1 to 6. For each value, a

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

classification experiment has been performed, where an ensemble of classifiers has been trained and evaluated using a 10-fold crossvalidation scheme. The individual classifiers in the ensemble are Naive Bayes (NB), 3-NN, RF, SVM, and C4.5. All of them were trained using the default parameters in their WEKA 3.6.1 implementation. The S2 feature set from Table 4.21 was used to describe transcribed music samples.

Table 4.22 shows the estimated accuracy for individual classifiers on the transcribed GTZAN corpus, while Table 4.23, and Figure 4.27 shows results for the ensemble, using the voting schemes discussed in section 2.4.1. The experiments show that results are quite comparable for all values of P , except maybe $P = 1$ for some individual classifiers, and most of the ensemble voting methods. As a large value for P affects the transcriptor's performance negatively, $P = 3$ was chosen, if not otherwise indicated, as the default maximum transcription polyphony for music genre classification experiments discussed below.

P	NB	3-NN	C4.5	RF	SVM
1	35±3	28±5	41±3	47±5	48±7
2	36±3	29±5	48±3	54±4	49±5
3	35±4	34±5	46±5	55±3	49±5
4	36±2	34±4	48±4	55±5	48±4
5	36±3	33±4	49±4	55±4	48±3
6	37±3	32±4	49±3	54±4	48±4

Table 4.22: Transcriptor maximum polyphony (P) estimation results. Estimated accuracy percentages for individual classifiers on the transcribed GTZAN corpus, when using different values for P , and 10-fold crossvalidation.

P	PV	SVW	RSWV	BWWV	QBWWV	WMV
1	50 ± 7	50 ± 6	50 ± 6	51 ± 6	51 ± 6	51 ± 6
2	54 ± 2	56 ± 2	56 ± 2	56 ± 3	56 ± 3	55 ± 3
3	54 ± 4	56 ± 4	56 ± 4	56 ± 2	55 ± 2	55 ± 3
4	53 ± 4	56 ± 4	55 ± 4	55 ± 4	54 ± 4	54 ± 4
5	54 ± 3	56 ± 4	56 ± 4	57 ± 3	57 ± 4	56 ± 5
6	55 ± 2	56 ± 3	56 ± 3	56 ± 4	55 ± 4	55 ± 5

Table 4.23: Transcriptor maximum polyphony (P) estimation results. Estimated accuracy percentages for classifier ensemble on the transcribed GTZAN corpus, when using different values for P , and 10-fold crossvalidation.

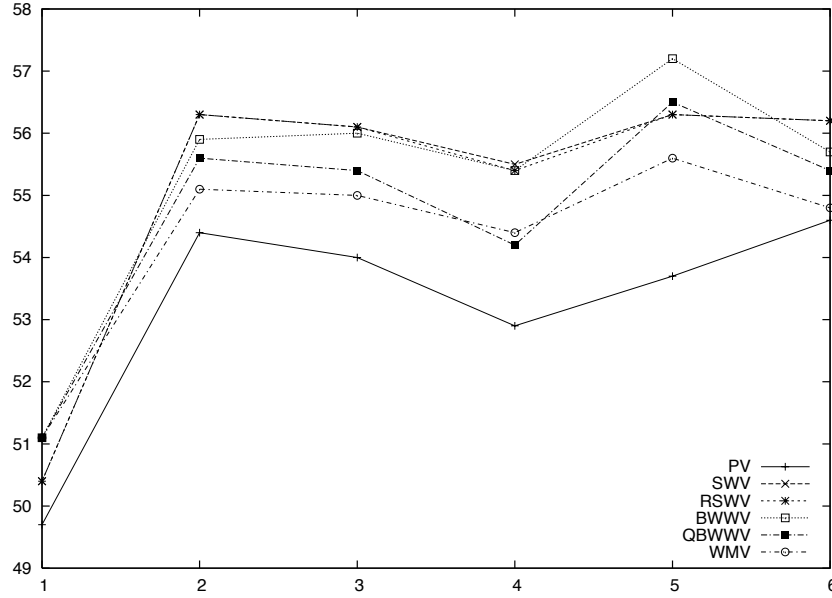


Figure 4.27: Genre classification results using the GTZAN dataset changing the maximum polyphony for the transcription method. The abscissae represent the maximum polyphony P and the ordinates the success rate for the different classifier ensembles tested.

4.5.3 Early fusion results

The first early fusion experiments performed within the collaboration research mentioned above were published in (Lidy et al., 2007). The symbolic feature set S1 was combined with audio feature set A1 (RP, RH, SSD, and onset features). The GTZAN, ISMIRrhythm, and ISMIRgenre datasets were used as evaluation corpora. The evaluation method was to perform 10-fold stratified crossvalidation. A linear SVM was chosen as the classifier to train, using the SMO implementation of the WEKA machine learning software with pairwise classification and default parameters (complexity parameter $C = 1.0$). The performance of each feature set was investigated individually, before deciding which ones to combine. Some combinations were chosen to make results comparable to those obtained in (Lidy and Rauber, 2005), where only audio feature sets were combined.

Table 4.24 reports a summary of accuracy results for individual models. Considering a “dumb classifier” attributing all pieces to the class with the highest probability (i.e. the largest class), the lower baseline would be 10% accuracy for the GTZAN data set, 15.9% for the ISMIRrhythm data set and 43.9% for the ISMIRgenre data set. Interestingly, the symbolic descriptors derived from transcribed audio surpassed in performance the RH features,

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

which are computed directly from audio, on the ISMIRgenre data set, and they also achieved remarkable performance on both other data sets.

Table 4.25 reports accuracy results for SVM trained with combined feature sets, along with results from (Lidy and Rauber, 2005), where the SVM was trained with several combinations of audio feature sets. Clearly, there is a consistent improvement on all corpora when using combined audio and symbolic feature sets, with respect to both single feature sets and combinations of audio features only. Although such improvements are not of substantial magnitude, it seems that the “glass ceiling” in audio music classification described in (Aucouturier and F., 2004) can be surpassed by combining features that describe music from different modalities. However, there is no single feature combination in Table 4.25 that outperforms the others consistently. This can be seen as a drawback of this early fusion approach, that could be circumvented using an alternative: the late fusion approach.

The authors provided some comparisons with similar studies by other authors. For example, (Li et al., 2003) reported an accuracy of 74.9% in a 10-fold crossvalidation of DWCH features on the GTZAN data set using SVM with pairwise classification, surpassed by a 76.8% average accuracy reported here. A 66.9% average accuracy in a 10-fold crossvalidation set up on the ISMIR rhythm dataset is reported in (Flexer et al., 2006), while (Dixon et al., 2004) reports achieving 85.7% accuracy on the same corpus. The proposed approach discussed here obtained 90.4% average accuracy.

Feature set	GTZAN	ISMIRrhythm	ISMIRgenre
Onsets	34.9	48.4	58.0
Symbolic	41.3	51.1	66.0
RH	44.0	82.7	64.4
SSD	72.6	59.6	78.6
RP	64.4	86.5	75.9

Table 4.24: Accuracy percentage results for a SVM trained on different feature sets (from (Lidy et al., 2007)).

This experimental set up was presented to the MIREX 2007 evaluation contest in several categories: Audio genre classification, audio music similarity, audio artist identification, classical composer identification, and audio mood classification. The evaluation was performed using a 3-fold crossvalidation. In genre classification, artist filtering was used for test and training splits, i.e. training and test sets contained different artists. In classic

CHAPTER 4. MUSIC GENRE RECOGNITION

Feature set	GTZAN	ISMIRrhythm	ISMIRgenre
Onset+Symb.	50.5	61.6	68.0
SSD+Onset	74.5	67.6	79.6
SSD+Symb.	75.7	63.6	81.0
SSD+Onset+Symb.	76.1	69.5	81.4
RH+SSD+Onset+Symb.	76.8	87.1	80.5
RP+SSD+Onset+Symb.	74.3	89.8	80.6
RP+RH+SSD+Onset+Symb.	74.0	90.4	80.9
Best results in (Lidy and Rauber, 2005)	74.9	84.2	80.3

Table 4.25: Early fusion accuracy results for a SVM trained on combined feature sets (from ([Lidy et al., 2007](#)) and ([Lidy and Rauber, 2005](#))).

composer identification, album filtering was used, i.e. training and test sets contained tracks from different albums.

The submitted system used the RP+RH+SDD+Onset+Symb combined feature set. A summary of the different contest results is shown in Table 4.26. It shows that the system yielded a high success rate for genre classification. In the music similarity contest, a system variant without symbolic features was also submitted, but was outperformed by the whole feature set combination. In the case of audio artist identification and classical composer identification, the system yielded encouraging results, and for mood classification the results were satisfactory.

Contest	Best	(Lidy et al., 2007)	Rank
Genre classification	68.29%	66.71%	2/7
Audio music similarity (*)	0.568	0.519	5/12
Audio artist identification	48.14%	38.76%	4/7
Classical composer identification	53.72%	47.26%	5/7
Audio mood classification	61.50%	59.67%	3/9

(*) F-score measures.

Table 4.26: MIREX 2007 average raw accuracy summary. The second column shows the result of the best system. The third column shows the results of the system discussed here, and the last one shows its rank, in terms of accuracy, with respect to the total number of systems submitted.

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

Several areas of improvement were identified, notably on the symbolic descriptors side. A better transcription system is expected to help symbolic descriptors to improve their performance. Also, the addition of new symbolic features could provide a better representation of the transcribed signal, as well as the addition of new audio feature sets.

All these suggestions were investigated, and a new system was presented to the MIREX 2008 evaluation contest (Lidy et al., 2008). It included the A3 audio feature sets, the S2 symbolic descriptor set, and a tweaked transcription system that used gaussian smoothness and short context to improve the transcription. The system was submitted to the audio classification tasks, which included the genre, artist, and mood classification contests, with the same evaluation conditions used in MIREX 2007. The corpora used in each task were the same as in the previous edition. For the genre classification task, the LMD corpus was used for a new ‘latin music’ genre classification task, besides the ‘western music’ task defined on the previous edition genre dataset.

Four variants of the system, using different feature set combinations, were submitted:

1. RH+SSD,
2. RP+SSD+MVD,
3. RP+SSD+TRH+Symb,
4. RP+RH+SSD+TRH+TSSD+MVD+Symb.

A summary of results is shown in Table 4.27. MIREX 2008 used the ANOVA significance test with Tukey-Kramer HSD (Honestly Significant Difference) for multiple comparisons. The performance of the system on the artist, mood, and ‘western music’ genre was unexpected, as the results were in general poorer than those obtained in the previous edition on the same corpora. Also, the trend of the audio and symbolic combined feature sets outperforming the combination of audio only features was inverted.

Due to the differences in the combinations used in this evaluation contest with respect to MIREX 2007, and the fact that the evaluation data are not distributed by the IMIRSEL⁹, it is hard to draw any conclusion on why the system performance was not improved as expected. This in part supports the intuitive claim that early fusion approaches are very sensitive to which particular feature set combination is used. It is also an evidence in favor

⁹The *International Music Information Retrieval Systems Evaluation Laboratory* runs the MIREX evaluation contests.

CHAPTER 4. MUSIC GENRE RECOGNITION

of using alternative approaches, like late fusion methods, to try to improve results in a more robust way.

Contest	Best	(Lidy et al., 2008)	Rank	TK-HSD
Genre classification (western)	66.41%	65.06% (1)	6/13	false
Genre classification (latin)	65.17%	62.23% (2)	4/13	false
Audio artist identification	47.65%	35.42% (1)	5/11	false
Classical composer identif.	53.25%	39.54% (4)	7/11	false
Audio mood classification	63.67%	56.00% (1)	3/13	false

Table 4.27: MIREX 2008 average raw accuracy summary. ‘Best’ column shows the result of the best system. ‘Ours’ column shows the results of the best submitted early fusion system variant. The variant number is shown in parenthesis. The ‘rank’ column shows the system rank, in terms of accuracy, with respect to the total number of systems submitted. The last column, TK-HSD, indicates whether the proposed system average accuracy is significantly different from the best system average accuracy.

4.5.4 Late fusion: the cartesian ensemble

With the availability of multiple feature sets as a source of music description, and potentially also multiple classifiers, there are several alternatives of how to design a music multiple classification system. The early fusion approach taken in the previous section pointed out the need for a more robust classification scheme. Taking a late fusion approach is a natural step forward in the attempt to combine multiple feature subspaces that describe the same musical objects. Moreover, results discussed in section 4.4 provided hints about the robustness of voting classifier ensembles in music genre recognition. Also, the diversity of the ensemble has been identified as a key factor for performance improvement over single classifiers.

Late fusion techniques always involve training several base models, from which predictions on test objects are later combined. For voting ensembles, several approaches can be taken to obtain a set of base trained models:

- build a classifier for each class using a *one-against-all* strategy,
- train a model for each pair of classes in the problem,
- train different learning schemes are trained in the same feature space, or

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

- if several feature spaces are available to describe input objects, build different models using the same classification method on different feature sets.

Our approach is to apply the last two strategies simultaneously. It has been named a *cartesian ensemble* because the set of models used as base classifiers is the cartesian product of D feature subspaces and C classification schemes. A model is built by training a classification scheme c_i on a feature subspace, d_j . This produces a total of $D \times C$ base models as the ensemble. The aim of this approach is to obtain a sufficiently *diverse* ensemble of models that will guarantee, up to a certain degree, an improvement of the ensemble accuracy over the best single model trained. Moreover, the ensemble abstracts from the selection of a particular classifier and feature set to use for a particular problem. Selecting sufficiently different schemes (different classification paradigms, methods,...) the ensemble provides results that are at least comparable to the best single scheme. For selecting the most diverse models within the ensemble the *Pareto-optimal* selection strategy is applied in order to discard models not diverse or not accurate enough.¹⁰

When a new music instance is presented to the trained ensemble, predictions are made by selected models, which are then combined to produce a single category prediction outcome. A number of decision *combination* (or label fusion) *rules*, can be used for this final prediction, like the ones presented in section 2.4.1.

The cartesian ensemble system is depicted in Figure 4.28. It is built on top of the WEKA toolkit. The ensemble is a WEKA classifier itself, so it can be plugged into any system using this toolkit. Any classification algorithm available on the toolkit can be used with arbitrary parameters in the ensemble. Further, an arbitrary number of feature files can be used. The system provides a number of combination and voting rules, which are employed to obtain the final prediction of the classifier ensemble. The system is not limited to MIR applications. With regard to the original motivation of this thesis, however, let us focus on the scenario of music classification into genre categories, in order to show the applicability of the system and the progress in this domain.

Instance alignment

A particular challenge when using a variety of feature sets that were generated by different feature extractors and written to different file formats is the correct alignment of instances from different feature subspaces in memory,

¹⁰cf. sec. 2.4.2

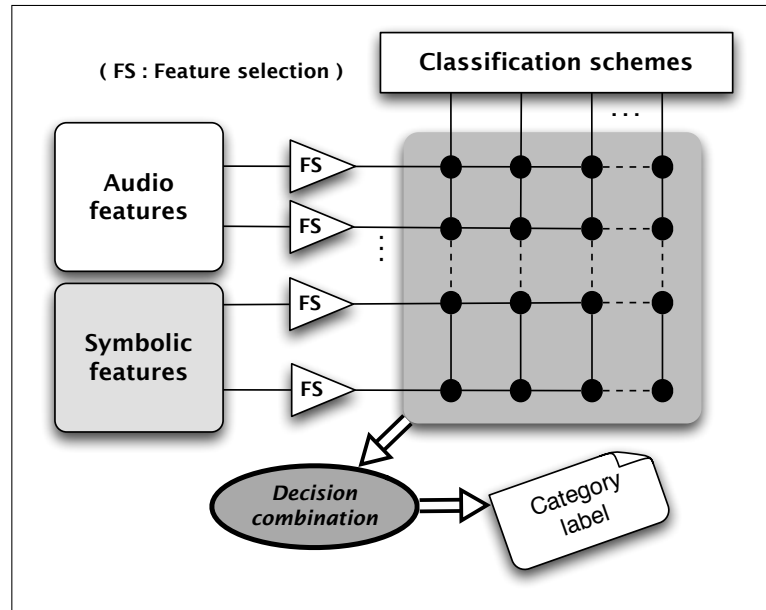


Figure 4.28: Overview of the cartesian ensemble system

when processing these diverse files. An *instance aligner* preprocessing component of the cartesian ensemble takes care of this. It accepts a range of different feature sets as inputs from different feature files and – based on a common ID field, which here is the filename of the piece of music – checks for completeness of instances in all files, aligns the instances to a common order, and constructs a convenient joint data structure, which is the prerequisite for all further ensemble classification steps.

The instance aligner takes one of the feature subspace sets as its *reference dataset*. This set must be fully tagged and must contain the ID field. Clearly, only one instance per ID is allowed. Further feature subspace datasets must also contain the ID field, and instances for the same IDs as in the reference dataset. However, these datasets can contain additional instances (with other IDs) and need not to be tagged. Only the tags from the reference dataset will be used for training and testing purposes. The ID field is obviously removed for training and testing, and it is used only for keeping instances aligned.

Feature selection

The cartesian ensemble provides the possibility of performing a feature selection step for each feature subspace at each fold, using outer training data. The feature selection method used here is a *fast correlation-based filter (FCBF)* described in section 2.5.2. FCBF has been shown to efficiently

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

achieve a high degree of dimensionality reduction for high-dimensional data, while enhancing or maintaining predictive accuracy with selected features.

Ensemble evaluation

The classifier ensemble crossvalidation scheme described in section 2.8.2 is used in the experiments that follow. Recall that this procedure involves an *inner* crossvalidation to estimate base model accuracies, needed to set up model weights, and perform the model selection step. This step is performed within each *outer* crossvalidation fold, aimed at estimating the whole ensemble performance. This means that, for each outer training fold, each base model,

- is crossvalidated by partitioning the training fold.
- its estimated accuracy is recorded and used to compute its weight in the ensemble.
- it is trained with the whole outer training fold.

So, for each iteration of the outer crossvalidation process, all base models are retrained and their respective weights recomputed.

4.5.5 Late fusion results

The first evaluation of the cartesian ensemble was performed at the MIREX 2009 evaluation contest (Lidy et al., 2009). The A2 and S2 set of feature subspaces were used, without the feature selection step. The base classification schemes used are listed in Table 4.28. A 3-fold inner crossvalidation was used. Two variants of the system, using different ensemble voting rules, were evaluated: (1) WMV, and (2) BWWV voting rule, based on pre-contest experimental results.

The system was submitted to the so called the ‘Train-Test Task Set’:

- Genre Classification (western music corpus)
- Genre Classification (latin music corpus)
- Music Mood Classification
- Classic Composer Identification

CHAPTER 4. MUSIC GENRE RECOGNITION

Scheme	Parameters
Naïve Bayes (NB)	default
HyperPipes (HP)	default
3-NN	Euclidean dist.
1-NN,	Manhattan dist.
1-NN	Chebichev dist.
RIPPER	default
C4.5	default
REPTree	default
Random Forest (RF)	default
SVM, linear kernel (SVM-lin)	default
SVM, quadratic kernel (SVM-quad)	default
SVM, Puk kernel (SVM-Puk)	$C = 4, \omega = 3.2, \sigma = 13.0$

Table 4.28: Base classifiers for the cartesian ensemble submission to the MIREX 2009 evaluation contest. ‘default’ parameters means the default Weka 3.6.1 implementation parameters are used.

The corpora used for evaluation were the same as in the previous edition. The evaluation was performed using a 3-fold crossvalidation, that corresponds to the outer crossvalidation of the ensemble. In genre classification, artist filtering was used. Album filtering was used in the classic composer identification task. As in other editions of MIREX, the ANOVA significance test with Tukey-Kramer HSD for multiple comparisons was applied to the results. A summary of the ‘Train-test task set’ results in MIREX 2009 is presented in Table 4.29.

Contest	Best	Cartesian ensmb.	Rank	TK-HSD
Genre classification (western)	73.33%	68.84% (1)	8/31	false
Genre classification (latin)	74.66%	58.37% (1)	12/32	false
Classical composer identif.	62.05%	53.57% (1)	11/29	false
Audio mood classification	65.67%	60.17% (2)	9/28	false

Table 4.29: MIREX 2009 average raw accuracy summary.

The cartesian ensemble obtained results consistently higher than average MIREX 2009 results. When compared to early fusion results obtained in previous MIREX contest (see Table 4.30), the late fusion approach is superior, in general, to early fusion methods. The only exception is the LMD corpus, where the accuracy of the cartesian ensemble was expected to be

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

higher. Expectations were based on previous results obtained on this dataset. Table 4.31 shows these results, obtained using a 5-fold outer/3-fold inner crossvalidation, without artist filtering. The system was configured with the same feature subspaces used for MIREX 2009, but only five classification schemes: Naïve Bayes, 3-NN, C4.5, Random Forest, and SVM with default WEKA 3.6.1 parameters. Up to a 92% average accuracy was obtained with the SVW and RSWV voting rules. The MIREX 2009 evaluation result on LMD was 58.37%, almost a 34% drop in performance.

This issue was investigated by Andrei Greçu, from our partner research group in Vienna, who also submitted a system for music genre recognition to the contest, which suffered similar drops in performance. That system is based on SMVs trained using audio features only (Greçu et al., 2009). Using a 3-fold crossvalidation without artist filtering, it reached a 92.6% average accuracy, a result very similar to that obtained with the cartesian ensemble system. After applying an artist filter to the data, performance dropped to 77.6%. However their result in MIREX was 62.79%. So the particular artist filter set up used at MIREX seems in part responsible for the lower performance. After informal conversations with people from IMIRSEL (the MIREX evaluation laboratory), the drop is more or less explainable. They agreed that some genre train/test splits get highly deteriorated using an artist filter, due to a highly unbalanced number of songs per artist in some genres.

Take, for example, the genre ‘Tango’, from LMD. It has only five artists, one of which, *Carlos Gardel*, is the artist in 335 out of 408 files. Using an artist filter the train/test splits get deteriorated, making it essentially impossible to recognize that genre. There is the additional circumstance that *Carlos Gardel* is arguably the most important tango artist. Training a tango genre recognizer without Gardel’s songs risks of not capturing the essential traits of such genre, and will probably have a rather low accuracy in identifying his songs. The same issue can probably appear whenever a strong artist-genre relation exists, even if the number of songs from this artist is not particularly high in the genre dataset.

A benchmarking study of the cartesian ensemble performance

An extensive benchmarking study of the cartesian ensemble’s performance has been carried out, without considering artist filtering, due to issues mentioned above. Moreover, the focus in this study was put on whether the ensemble performed consistently better than single classifiers in the music genre recognition task.

CHAPTER 4. MUSIC GENRE RECOGNITION

Task	Early fusion		Late fusion
	2007	2008	2009
Genre classification (western)	66.71	65.08	68.84
Genre classification (LMD)	–	62.23	58.37
Classical composer identif.	47.26	39.54	53.57
Audio artist identification	38.76	35.42	–
Audio mood classification	59.67	56.00	60.17

Table 4.30: Comparison between early and late fusion approaches. Average raw accuracy results from MIREX evaluation contests are shown.

Voting rule	Acc.	(stdev)
PV	91.88	(1.58)
SWV	92.00	(1.03)
RSWV	92.06	(0.87)
BWWV	91.94	(0.90)
QBWWV	91.60	(1.17)
WMV	91.88	(0.82)

Table 4.31: Cartesian ensemble accuracy on the LMD dataset (without artist filtering).

The 9GDB, GTZAN, ISMIRgenre, ISMIRrhythm, LMD, and Africa have been used in this study. For the Africa dataset, various meta-data categories are available, including 27 different *functions*, 11 different *instrument families*, 11 different *countries* and 40 *ethnic groups* (Lidy et al., 2010b). The number of files varies according to the number of meta-data available in each category.

System and evaluation parameters The A3 audio feature sets, and the S3 symbolic feature set have been used as feature subspaces for every dataset. In order to correctly extract the chordal features in S3, the transcriptor maximum polyphony parameter has been set to $P = 5$. The classification schemes listed in Table 4.28 have been utilized, with the exception of the 1-NN with Chebyshev distance. It produced classifications results very close to those produced using the Manhattan distance, and thus has been removed.

The voting methods described in section 2.4.1 have been tested. In addition, three other unweighted methods have been also tested, for comparison purposes. They are listed in Table 4.32, where $P(\omega_k|\mathbf{x}_i)$ is the posterior probability of instance \mathbf{x} to belong to category ω_k , given by model h_i . \mathbf{x}_i is

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

what h_i knows about \mathbf{x} , i.e., feature values that correspond to the feature subspace h_i was trained on. The posterior probability that instance \mathbf{x} belongs to class ω_k given by the ensemble is denoted by $P(\omega_k|\mathbf{x})$, and is computed as indicated in the second column of the table. The ensemble predicts class ω_j for instance \mathbf{x} if

$$P(\omega_j|\mathbf{x}) = \max_{k=1}^M P(\omega_k|\mathbf{x}) \quad (4.12)$$

where M is the number of classes in the problem. These unweighted combination rules are further described in (Kittler et al., 1998), and used through their implementation in WEKA.

Rule mnemonic	$P(\omega_k \mathbf{x})$
AVG	Average of $P(\omega_k \mathbf{x}_i)$
MAX	Maximum of $P(\omega_k \mathbf{x}_i)$
MED	Median of $P(\omega_k \mathbf{x}_i)$

Table 4.32: Additional unweighted voting methods used with the cartesian ensemble.

Two sets of experiments have been performed, one without feature selection, the other applying a feature selection stage at the feature subspace level. A 10-fold outer / 3-fold inner crossvalidation has been set up for every experiment.

Experiment results without feature selection

Dataset	Classifier	Featureset	Accuracy
9GDB	SVM-Puk	TSSD	78.15
GTZAN	SVM-lin	SSD	72.60
ISMIRgenre	SVM-quad	TSSD	81.28
ISMIRrhythm	SVM-lin	RP	87.97
LatinMusic	SVM-Puk	TSSD	89.46
Africa/country	SMO-quad	SSD	86.29
Africa/ethnic group	SVM-lin	TSSD	81.10
Africa/function	1-NN	SSD	51.06
Africa/instrument	SVM-Puk	TSSD	69.90

Table 4.33: Best results on individual classification of feature sets and classifiers on different datasets

CHAPTER 4. MUSIC GENRE RECOGNITION

$D = 7$ feature subspaces have been extracted: six audio feature sets (A3), plus one symbolic feature set (S3). $C = 11$ classification schemes were set up for the ensemble, making a total of $D \times C = 77$ base models. To have a baseline for the cartesian ensemble, all the models have been trained separately using a 10-fold crossvalidation. Table 4.33 gives an extract of the accuracies achieved with these single models. Only the best combination of an algorithm and a feature set is given. It can be observed that there is no clear trend, neither for a classifier, nor a feature set. While SVMs clearly dominate the results, the choice of the kernel is not obvious, and results can vary by several percent points. Also the feature sets do not show a clear trend – in approximately half of the cases, TSSDs are the best set to use, while also SSD and RP features sometimes yield clearly better results. These results nourish the hypothesis that ensemble classifiers may provide means to release the user from the difficult choice of the proper feature set and classifier combination.

The accuracy results for the classifier ensembles are shown in Table 4.39, with the best single classifier as our assumed baseline to improve on. Note that achieving the baseline result would require to know the best combination of feature set and classifier in advance. On each of the datasets, we can observe higher classification accuracies with the ensembles than with the baseline. The improvements are three percent points on average. The highest gains are on the GTZAN dataset, with five percent points, while the improvements on the ISMIRrhythm dataset are of 1.14%. However, the baseline on this dataset is already very high, around 88%.

Out of the nine classification tasks, the QBWWV rule performed five times the best, followed by WMV which was three times the best rule. AVG and BWWV are both once the highest ranked combination rule. In the tasks where QBWWV is not the rule with the highest accuracy, the relative difference to the top rule is minimal – the largest margin is 0.7% percent points, or 0.86% relative difference.

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

Rule	9GDB	GTZAN	ISMIR genre	ISMIR rhythm	Latin Music	Africa country	Africa ethnic group	Africa function	Africa instrument
Best	78.15 (2.25)	72.60 (3.92)	81.28 (3.13)	87.97 (4.28)	89.46 (1.62)	86.29 (2.30)	81.10 (2.41)	51.06 (6.63)	69.90 (4.69)
PV	79.56 (4.78)	72.60 (3.31)	77.78 (2.15)	88.25 (5.08)	89.33 (1.55)	85.31 (4.04)	71.86 (3.41)	37.37 (7.36)	59.63 (5.79)
MAX	60.05 (6.67)	44.00 (6.60)	60.97 (6.71)	54.87 (8.95)	50.64 (2.06)	77.67 (9.16)	73.16 (6.40)	40.38 (7.10)	61.32 (5.88)
MED	74.30 (4.32)	55.90 (3.84)	72.02 (2.74)	77.79 (4.27)	73.64 (2.37)	83.84 (3.77)	70.71 (3.62)	39.49 (5.22)	60.34 (4.67)
AVG	81.66 (3.96)	68.40 (2.37)	79.70 (3.35)	86.82 (4.29)	86.85 (1.96)	87.66 (2.28)	78.21 (3.50)	53.73 (5.35)	70.60 (3.82)
SWV	81.31 (3.32)	77.10 (3.98)	78.33 (2.48)	88.97 (5.39)	92.00 (1.34)	86.97 (2.98)	75.47 (3.62)	46.83 (4.44)	67.09 (3.99)
RSWV	80.96 (3.26)	77.40 (4.22)	79.22 (2.38)	88.97 (4.94)	92.25 (1.16)	87.17 (2.77)	75.47 (3.62)	48.39 (5.63)	68.35 (4.22)
BWWV	81.54 (3.17)	77.40 (4.22)	82.03 (1.83)	89.11 (4.62)	92.25 (1.16)	88.34 (2.22)	79.37 (3.95)	52.61 (5.76)	72.71 (3.47)
QBWWV	80.96 (2.94)	77.50 (4.30)	84.02 (1.50)	88.97 (3.86)	92.71 (0.99)	89.03 (1.63)	82.68 (3.18)	54.84 (6.29)	72.86 (3.52)
WMV	80.84 (2.90)	76.10 (4.20)	84.02 (2.02)	87.97 (3.92)	92.59 (1.29)	88.93 (1.76)	82.97 (3.30)	51.28 (6.93)	73.00 (4.25)

Table 4.34: Results of the ensemble classification on different datasets (Standard deviations are given in parentheses). Second row shows single classifier best result.

CHAPTER 4. MUSIC GENRE RECOGNITION

Experiment results applying feature selection

The *fast correlation-based filter (FCBF)*, described in section 2.5.2, has been selected as a feature selection method for the cartesian ensemble. Only the GTZAN, 9GDB, ISMIRgenre, and ISMIRrhythm datasets were used in these experiments. The same audio and symbolic feature sets, classification schemes, and other system parameters as in the previous section are used here.

Feature selection is performed independently for each subspace at each fold, using outer training data. Table 4.35 presents a summary of the best single model classification results. All accuracies are average values over folds, except for the last column. The fourth column shows the average number of features selected per fold, also as a percentage of the whole feature subspace. The last column shows performance results without feature selection for the given classifier/subspace combination. Table 4.36 shows the best classifier/subspace combination results without feature selection. This comparison suggests that SVM classifiers' performance on these datasets deteriorates more than schemes based on decision trees.

Dataset	Classifier	Subspace	avg. feat. sel.	Acc. (%)	Acc. (%) (all feat.)
9GDB	RF	TSSD	42.6 (3.6%)	73.2	76.5
GTZAN	RF	TSSD	19.1 (1.6%)	53.1	58.2
ISMIRgenre	RF	TSSD	20.5 (1.7%)	72.4	73.2
ISMIRrhythm	SVM-lin	RP	38.0 (2.6%)	81.6	88.0

Table 4.35: Best results on single subspace/classifier combinations with feature selection.

Dataset	Classifier	Subspace	Acc. (%)
9GDB	SVM-Puk	TSSD	78.2
GTZAN	SVM-lin	SSD	72.6
ISMIRgenre	SVM-quad	TSSD	81.3
ISMIRrhythm	SVM-lin	RP	88.0

Table 4.36: Best results on single subspace/classifier combinations without feature selection.

Table 4.37 provides further insight on the feature selection step on the GTZAN corpus. These results are averaged over cross-validation folds. The third column shows that the greatest dimensionality reduction is obtained for large audio feature subspaces. The fourth column indicates how many

4.5. FUSION FROM AUDIO AND SYMBOLIC DOMAINS

Features	# feats.	avg. selected	at least once	always	best avg. acc. (%)
RH	60	2 (3.3%)	7 (11.7%)	0	27.5 (NB)
RP	1440	9.1 (0.6%)	49 (3.4%)	1	41.2 (RF)
SSD	168	7.4 (4.4%)	15 (8.9%)	4	49.3 (RF)
TRH	420	2 (0.5%)	7 (1.7%)	0	28.1 (NB)
TSSD	1176	19.1 (1.6%)	71 (6%)	4	53.1 (RF)
MVD	420	4.6 (1.1%)	22 (5.2%)	1	31 (RF)
SYMB	63	7.3 (11.8%)	14 (22.6%)	3	44.2 (SVM-quad)

Table 4.37: Feature selection results for the GTZAN corpus.

features have been selected at least once. It is worth noting that for the larger subspace, only 3.4% of the features have been selected once. The next column shows how many features were always selected. Here, the symbolic feature subspace shows the best ratio of very predominant features. The accuracy column indicates average accuracy for the best single model trained with selected features. The Random Forest scheme shows best performance most of the time.

Feature selection times are negligible when compared to training times, as shown in Table 4.38. Moreover, training and testing times are an order of magnitude lower than those obtained using the ensemble with all features available. Though, in general, the accuracy of single models decreases when applying a feature selection step, it remains reasonably good for most of the benchmarking corpora.

Corpus	all features		with feature selection		
	train	test	train	feat. sel.	test
9GDB	6645	140	905	93	18
GTZAN	10702	345	1247	96	23
ISMIRgenre	12510	275	1244	133	23
ISMIRrhythm	5466	185	707	60	8

Table 4.38: Ensemble cross-validation execution times (in seconds) with and without feature selection (test times are averaged over combinations methods).

The integration of a feature selection stage aims at speeding up training and testing phases, while maintaining a good accuracy level on the task of music genre classification. The concern in this experiments was to integrate a feature selection step in the cartesian ensemble and evaluate its impact on the ensemble performance on several datasets. The size of feature subspaces

CHAPTER 4. MUSIC GENRE RECOGNITION

Rule	9GDB	GTZAN	ISMIR genre	ISMIR rhythm
Single best	73.2 (4.9)	53.1 (4.9)	72.4 (3.2)	81.6 (4.3)
MAJ	71.38 (5.16)	61 (5.01)	55.83 (1.4)	81.09 (4.8)
MAX	63.9 (5.3)	45.9 (3.41)	67.97 (2.55)	59.46 (7.71)
MED	64.72 (4.88)	42.8 (4.18)	54.39 (2.36)	66.19 (5.56)
AVG	80.02 (5.88)	65.2 (3.58)	70.23 (2.21)	83.52 (4.84)
SWV	77.69 (4.36)	61.7 (5.54)	56.1 (2.28)	82.95 (5.37)
RSWV	78.27 (4.71)	62 (4.69)	57.96 (3.13)	83.81 (5.07)
BWWV	78.97 (5.14)	61.8 (4.73)	67.08 (3.73)	83.67 (5.16)
QBWWV	79.67 (5.23)	60.7 (4.62)	71.81 (1.82)	84.53 (4.93)
WMV	77.92 (5.17)	53.2 (3.97)	72.43 (1.98)	84.24 (4.29)
Best wo/ FS	81.66 (3.96)	77.50 (4.30)	84.02 (1.50)	89.11 (4.62)

Table 4.39: Results of the ensemble classification (standard deviations in parentheses)

has been greatly reduced to less than 4% of their original size, on average. This has been achieved while maintaining the classification accuracy at a reasonable good level at least for two of the benchmarking corpora, 9GDB and ISMIRrhythm, as shown in Table 4.39. The ‘Single best’ row corresponds to the fifth column of Table 4.35, i.e., the best performing base model trained on selected features, in average. The experiments were designed to ensure that, given a feature subspace and a dataset fold, the feature selection algorithm would select the same features when training single models or the whole ensemble. The last row shows the best results obtained without feature selection, from Table 4.39. From these results, the final word would be to say that there should exist a tradeoff between the gain in training time and the loss of discrimination power in the ensemble. This should be further investigated, for example, tuning the δ parameter of the FCBF algorithm, or choosing a different feature selection method.

4.6 Conclusions on music genre recognition

The main goal in this work has been to assess the capability of melodic, harmonic, and rhythmic statistical descriptors to perform musical genre recognition of symbolically encoded music. An important feature of the approach is that the use of timbral information is purposely avoided. A methodology for feature extraction, selection and music genre classification experiments was developed, where new corpora, description models, and classifiers can be easily incorporated, and tested in a systematic way on

4.6. CONCLUSIONS ON MUSIC GENRE RECOGNITION

a given range of melody fragment sizes. Both single classifiers and ensembles have been evaluated on a two-genre problem (jazz vs. classical music). This statistical pattern recognition approach has been also tested in combination with audio features over a broader domain of music genres.

In order to establish some reference baseline for music genre recognition, a survey on genre recognition by humans in the absence of timbral information has been performed. A 16% average human recognition error rate (84% accuracy) has been obtained for a two-genre problem (jazz and classical music), when dealing with short (3 bars) to medium sized (16 bars) melody fragments. This result has been taken as a base line for comparison with automatic music genre recognition approaches discussed in this chapter.

The feasibility of the proposed categories of descriptors has been assessed through several single feature evaluation experiments. Pitch and pitch interval descriptors have shown to be the most discriminant features. Other important features have been the number of notes and the rhythm syncopation. From the statistical point of view, standard deviations were very relevant. Though descriptors based on note onset and duration performed worse when used alone, their use in combination with pitch related descriptors should not be discarded.

The music genre recognition experiments have been carried out over a parameter space defined by ω , the size of segments extracted from melody tracks of MIDI files. The two classifier schemes tested were a quadratic Bayes and a nearest neighbor (NN) classifier. They performed comparatively well on feature sets of different sizes. The parametric approach preferred the reduced size feature sets but NN performed well with all models. The best average recognition rate has been found using the Bayesian classifier and a 12-descriptor model (89.5%), although the best result was obtained with the NN, that reached a 96.4% with $\omega = 95$. Note that both results are above the human recognition base line (84%) described above. In general, best results were obtained for medium to large melody fragments. However, a drop in performance has been found when very large fragments, or even the whole melody, is used, due to the reduction of the training data available in such conditions.

The experiments demonstrated that melodies encoded symbolically convey statistical information about genre, but that this information is incomplete. While quite good performance results have been obtained on the ‘jazz vs. classical’ problem, given that this is a relatively ‘easy’ task, there is evidence that a glass ceiling for classification performance exists when relying only on information provided by the melody in a music piece. Moreover, music genres have fuzzy boundaries, evidence of which has been detected

CHAPTER 4. MUSIC GENRE RECOGNITION

in the proposed feature space through its exploration and visualization by self-organising maps (SOM).

Voting ensembles of classifiers have been proposed for its use in music genre recognition. In particular, three novel decision combination approaches (RSWV, BWWV, QBWWV¹¹), have been proposed and evaluated on a series of classification benchmarking tasks. Their performance is shown to be precise and robust, revealing voting ensembles as a good choice for general purpose classification tasks, and in particular to music genre recognition, where they have improved both the accuracy and robustness of single classifier systems. The use of ensembles looks like an *a priori* better choice, reducing the risk of selecting the wrong classifier for a given problem.

The diversity of a voting ensemble, that is, the degree of variety in the decisions taken by its base models has been investigated in the context of music genre recognition, through the use of different feature spaces for building the ensemble. Evidence has been found that the use of different modalities (feature spaces) for training base models, increases the ensemble diversity, and improves its robustness and performance.

The proposed timbre-less, statistical symbolic music description has been used in combination with audio features in audio music genre recognition problems. They have been incorporated by means of a music transcription system. This system does not generate explicit timbre information in the transcription, so the use of such symbolic representation is specially adequate in this context. While the transcription system is designed for monotimbral, polyphonic transcriptions, it is assumed that ‘an inaccurate but systematic monotimbral symbolic music transcription of a multitimbral audio music signal still conveys information about genre’. Two feature space combination approaches have been evaluated: early and late fusion. The first one, consisting of feeding a classifier with a combination of symbolic and audio feature spaces, has shown a consistent, though not significant, improvement on all tested corpora, with respect to both single feature sets and combinations of audio features only. It also compares favorably to other authors’ results on the same datasets.

Some results, however, showed that early fusion approaches are very sensitive to which particular feature set combination is used. This is an evidence in favor of using alternative approaches, like late fusion methods, in order to try to improve results in a more robust way.

A cartesian ensemble system, which combines different feature spaces with different classification schemes, has been proposed as a late fusion approach for combining audio and symbolic descriptors for music genre

¹¹c.f. sec. [2.4.1](#)

4.7. CONTRIBUTIONS IN THIS CHAPTER

recognition. A set of benchmarking experiments on audio genre recognition have been performed, where it has been shown that the performance results obtained for the ensemble are superior, in general, to those obtained by early fusion methods. The QBWWV voting rule has emerged as the best option, in general, among other rules. These results nourish the hypothesis that ensemble classifiers may provide means to release the user from the difficult choice of the proper feature set and classifier combination.

Feature selection in the cartesian ensemble has been investigated. The symbolic feature subspace has shown a best ratio of very predominant features, when compared to audio feature sets. Training and testing times of the ensemble have been reduced an order of magnitude with respect to using all features available. The size of feature subspaces has been greatly reduced to less than 4% of their original size, on average. This has been achieved while maintaining the classification accuracy at a reasonable good level at least for two of the benchmarking corpora, 9GDB and ISMIRrhythm. These experiments in feature selection for ensembles showed that there should exist a tradeoff between the gain in training time and the loss of discrimination power in the ensemble.

4.7 Contributions in this chapter

The following are the main contributions of this research in music genre recognition, where statistical pattern recognition techniques are applied to symbolically encoded music:

- A survey on music genre recognition by humans has been conducted. Subjects were asked to recognize the genre of monophonic melodies from jazz and classical music, without timbral information. A 16% error rate baseline was suggested in this context for automatic genre recognition systems.
- A methodology for evaluating symbolic music genre recognition of melodies over a range of fragment sizes, and feature set sizes, has been presented. Using a corpus of jazz and classical melodies, the performance of a quadratic Bayesian classifier, and a nearest neighbor classifier is evaluated and discussed.
- The performance of voting ensembles of classifiers has been also investigated. Three novel voting rules are proposed, based on scaling the weights of the ensemble with respect to the best and worst classifiers. These rules have shown good performance on a series of

CHAPTER 4. MUSIC GENRE RECOGNITION

benchmarking classification experiments, and in particular for music genre recognition.

- A *cartesian ensemble* system has been introduced. It is a voting ensemble that combines different feature subspaces, and different classification schemes, build on top of the WEKA framework. It provides a feature selection stage for each feature subspace, and a model selection step that selects the most diverse models in the ensemble for the decision combination stage. It has been shown to perform superior to the best choice of a single algorithm on a single feature set. Moreover, it also releases the user from making this choice explicitly.

Besides the above contributions, some tools were developed to help research in symbolic music genre recognition. Obviously, there are the feature extraction tools, that have been implemented for its use with the WEKA toolkit. Also, a graphical user interface for symbolic music genre recognition was developed, that allows the user to visualize the result of the proposed music genre recognition models. A command line tool chain for converting MIDI files to a text representation, and vice-versa, has also been developed within my research group. It is an ubiquitous tool that showed valuable when rapid inspection and transformation of MIDI file corpora is needed.

5

Summary and future perspectives

This dissertation has presented a study on two applications of statistical pattern recognition in the field of symbolic music information retrieval: melody part selection (chapter 3), and music genre recognition (chapter 4). MIDI file corpora have been used as evaluation materials in both tasks. Music in such files is represented by vectors of statistical descriptors of note pitches, intervals, durations, etc., that summarize the content of a piece (or a fragment), while intentionally avoiding the use of any metadata. This allows us to investigate to what extent the proposed tasks can be solved relying on information extracted from the actual musical content. Or, from a different point of view, to investigate how much of music genre is conveyed in the music itself. A summary of the research done in both applications is presented below.

5.1 Melody part selection

Melody part selection is the problem of selecting the part(s) that most probably contains the melody in a multi-part music file. The problem is first approached by using random forest (RF) classifiers to categorize tracks as melody or non-melody tracks. Then this information is used to decide which track, or tracks, in a MIDI file are the ones containing the melody.

For evaluating the approach, several corpora containing MIDI files from jazz, classical, and popular music, and annotated and double checked by human experts have been built. The track categorization and melody part selection experiments yielded promising results, and showed that, given a genre, enough training data of each genre are needed in order to successfully characterize melody tracks, due to the specificities of melody and accompaniment in each genre. Classical music emerged as a particularly hard problem for this approach, because often the melody is not confined to

CHAPTER 5. SUMMARY AND FUTURE PERSPECTIVES

a single track. For this kind of music, more sophisticated schemes oriented to melodic segmentation are needed.

The RF-based system has been successfully used in commercial applications, by *Sherlock Hums LLC, Chicago*, for building mobile ringtones. The tool was used to help extracting melodies from MIDI files. It could also serve as a preprocessing step to any melody-based music processing system, such as query-by-humming systems, or the symbolic music genre recognition tools summarized in the next section.

A subsidiary goal of the melody part selection research was to answer the general question *What is a melody?* under a statistical approach. This was accomplished by providing human-readable characterizations of melody tracks automatically extracted from sample data. These models are expressed as fuzzy rules, derived from crisp rules learnt from MIDI corpora. As far as this author knows, these are the first melody models of its kind in the area of MIR. Being able to automatically obtain melody characterization rules that are easily understandable by humans could be of interest for musicologists. It would help building better tools for searching and indexing symbolically encoded music.

The approach to melody part selection can be easily adapted to the characterization of other track categories, like *bass line* tracks or *percussion* tracks. Sections 3.4 and 3.5 present detailed conclusions and contributions, respectively, of this research.

5.2 Music genre recognition

Music genre recognition has been explored from different angles. First, genre recognition of melodies encoded symbolically was considered. For this, a methodology for investigating the potential of the proposed statistical description of music content has been contributed. It relies on the definition of a music fragment length space, over which the feature extraction, feature selection, and classification stages are evaluated. Several kind of classifiers and statistical descriptor subsets have been assessed on a two-genre problem (jazz and classical music).

In order to have a reference on the expected performance of this statistical pattern recognition approach, a survey on genre recognition by humans in the absence of timbral information has been performed. The ability of the subjects to recognize the genre of jazz and classical melodies has been assessed over several population parameters.

Voting classifier ensembles have been proposed for music genre recognition in order to improve the performance and robustness of single classifiers.

5.3. PERSPECTIVES ON MELODY PART SELECTION

Three novel weighted voting rules have been proposed and evaluated. These ensembles have been extensively evaluated on several benchmarking datasets, including the two-genre corpus of jazz and classical music. The proposed voting rules showed best performance results for most of the evaluation experiments performed.

The ensemble approach allows the use of different music description modalities. In particular, the global statistical description scheme proposed in this research has been combined with a local statistical description approach based on n -grams, providing promising results. Furthermore, this same global description scheme has been combined with audio feature spaces to deal with audio genre recognition tasks. This has been accomplished by means of a transcription system. Two approaches for combining features have been evaluated: early fusion of features and late fusion of classifier decisions (through voting ensembles).

For the late fusion approach, a *cartesian ensemble* capable of combining different feature subspaces with different classification schemes has been contributed. It provides a feature selection stage for each feature subspace, which greatly decreases the training time needed for the ensemble. It also applies a model selection procedure for selecting the most diverse subset of models, in order to improve the overall performance of the ensemble.

Sections 4.6 and 4.7 present detailed conclusions and contributions, respectively, of this research on symbolic music genre recognition.

5.3 Perspectives on melody part selection

One of the problems that have been faced with the proposed melody part selection system was the case of *roving* melodies, often found in classical music. These are melodies that are not confined to a single track, but move from one track to another. The approach taken here is clearly not adequate to deal with these cases, as it is designed to propose whole tracks as melodies. While it can certainly be easily adapted to point at several tracks at once as the most probable tracks containing melodies, it will not be of much help with roving melodies, where more precise location is desirable. Though melody segmentation or extraction approaches are better suited to solve this problem, a method based on characterizing track segments could be applied: fragments from all tracks in a MIDI file would be extracted, using a sliding window. Their probability of being a melody would be computed, and then the most probable melody fragments, according to a given threshold, would be proposed, using the same methodology as applied to whole tracks.

CHAPTER 5. SUMMARY AND FUTURE PERSPECTIVES

The addition of entropy based measures proposed by other authors ([Madsen and Widmer, 2007b](#); [Margulis et al., 2008](#)) to the feature set utilized in this work is under research. Also, a feature selection process is planned to be applied to this enhanced feature set.

The annotation of corpora containing genres not considered here for melody part selection is going to be undertaken. The objective would be to evaluate the RF-based system on different genres. Also, the annotation of other kind of tracks in the melody annotated corpora, like *accompaniment*, *bass line*, or *percussion*, for example, is under development. This will allow to apply the same part selection methodology to select other kind of music parts. In particular, the method is currently being adapted to the selection of *bass line* parts. Moreover, both the detection of melody and bass line parts is planned to be combined, in such a way that the probabilities for a track to be a melody, for example, would condition the probability of other tracks to be a bass line, and vice-versa.

The characterization of melodies by fuzzy rule systems needs further investigation. There is evidence that different rules have emerged that characterize slightly different types of melodies, e.g., melodies for voice parts, instrumental melodies, sparse melodies, etc. A study of what kind of tracks are being selected by which rules would be undertaken in the near future.

Tuning the probability threshold for firing a fuzzy rule in fuzzy rule system for melody characterization is under research. Also, enforcing more than one fuzzy rule to be fired could help to improve the results. Alternative approaches for the rule fuzzification, e.g. by using information theory measures ([Makrehchi et al., 2003](#)) are envisaged. As some attributes appear to be more frequently used than other in rule systems, weighting fuzzy sets proportionally to their relative importance in the rules could be a way for improvement for the proposed fuzzy rule system.

5.4 Perspectives on music genre recognition

The proposed music genre recognition framework, based on a sliding window, is capable of applying classifiers ensembles to automatically tag MIDI track fragments. Given a MIDI file, suppose it contains P tracks t_1, t_2, \dots, t_P . Each of these tracks can be fragmented applying a sliding window method. For simplicity, let the window displacement δ be equal to the window length, ω . The number of fragments extracted from track t_i is

$$L_i = \begin{cases} 1 & \text{if } \omega \geq |t_i| \\ 1 + \left\lceil \frac{|t_i| - \omega}{\omega} \right\rceil & \text{otherwise} \end{cases} \quad (5.1)$$

5.4. PERSPECTIVES ON MUSIC GENRE RECOGNITION

where $|t_i|$ is the length of track t_i . Let us denote the set of such fragments as $\Omega(t_i) = \{t_i^1, \dots, t_i^{L_i}\}$. The application of a genre model h to each fragment produces a label l_{ij} . If the model is a trained voting ensemble of classifiers, each base model predicts a label l_{ij}^k for the fragment, and a voting method v is applied to obtain a single label l_{ij} per fragment,

$$h(t_i^j) = v(l_{ij}^1, \dots, l_{ij}^k) = l_{ij}. \quad (5.2)$$

This is what the music genre recognition methods discussed in this dissertation can already do. A further extension to the proposed scheme is under development, where a voting scheme w , using rules similar to those used in the proposed classifier ensembles, combines labels l_{ij} for each track t_i , thus producing a single output label l_i per track:

$$l_i = w(l_{i1}, \dots, l_{iL_i}). \quad (5.3)$$

If tracks are pre-tagged using some track characterization system like the ones proposed in chapter 3, and denoting that track t_i is pre-tagged with label m_i as $\langle t_i, m_i \rangle$, a piece-wise combination of track tags l_i could be further applied, thus producing a piece-wise label l for the whole music piece:

$$l = pw(\langle l_1, m_1 \rangle, \dots, \langle l_P, m_P \rangle) \quad (5.4)$$

This multi-level music genre recognition scheme is depicted in figure 5.1. The methodology would provide the possibility of ignoring windows due to lack of enough data (music content). Also, some individual windows could remain untagged due to lack of confidence in a particular classifier's decision.

A problem to be solved in this approach is how to train the models needed to classify fragments of different length, coming from different types of tracks. Clearly, training a different model for each possible combination of fragment lengths and track types is not feasible. One option would be to fix the window size system-wise, and train models only using fragments of this length. Choosing an adequate length could be a difficult decision to take, although the research discussed in chapter 4 provided some hints on which window lengths are likely to produce more accurate models.

Another alternative would be to train models with assorted fragment sizes, and apply them to tag fragments of any size. The performance of models trained in this way will be investigated in the future. From what was learned in this research, it seems like a lower bound on the window length should be applied to such variable-length, fragment-based models. Another issue to be solved is how this fragments are to be generated for training in the feature extraction stage. A method that would partition

CHAPTER 5. SUMMARY AND FUTURE PERSPECTIVES

tracks in fragments of random length looks like an obvious first approach. Alternatively, smart music segmentation methods could be applied ([Bruderer and McKinney, 2008](#))¹.

Another improvement to be investigated in the future is to provide the system with the capability to sequentially label track fragments, in a way that decisions on the previous n fragments affect the decision on the next fragment:

$$l_{ij} = h(t_i^j, \{l_{i,j-1}, l_{i,j-2}, \dots, l_{i,j-n}\}). \quad (5.5)$$

¹(cited by extended abstract)

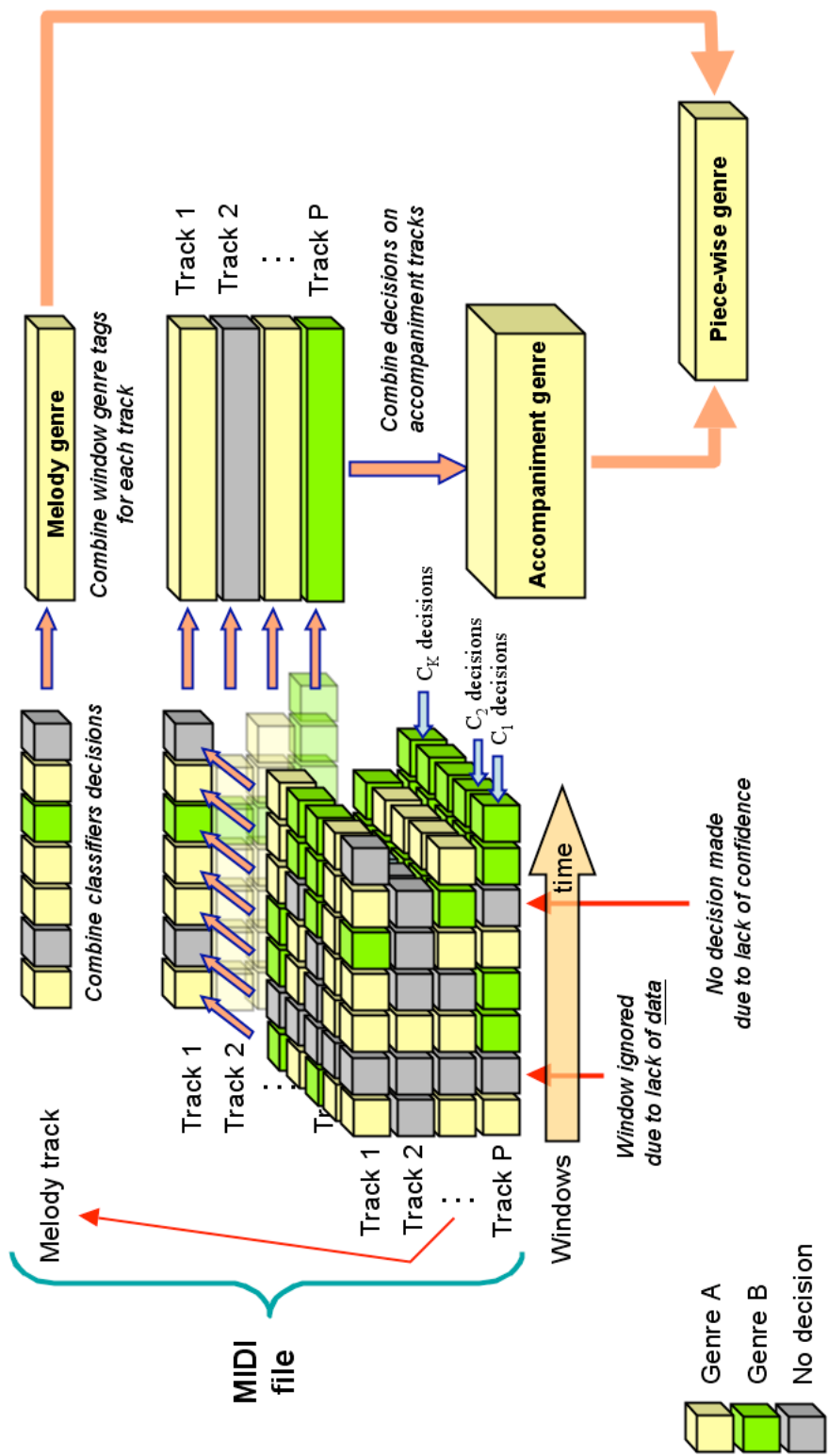


Figure 5.1: Multi-level music genre recognition system overview.

CHAPTER 5. SUMMARY AND FUTURE PERSPECTIVES

The development and evaluation of new weighted voting methods, like those discussed in section 2.4.1 is planned. In the case of audio and symbolic feature spaces fusion, several areas of improvement were identified, notably on the symbolic descriptors side. A better transcription system is expected to help symbolic descriptors to improve their performance. Also, the development of new symbolic features, adapted to the particular characteristics of the transcriptions, could provide a better representation of the transcribed signal, and thus, hopefully, a better overall classification performance.

The application of symbolic music genre recognition methods to automatic composition systems has been another field where some research effort has been invested. In particular, an automatic genre-oriented composition prototype based on genetic algorithms has been developed (Espí et al., 2007). In this system, candidate melodies represented using the description scheme discussed in section 4.3.1 are assigned a fitness measure based on distances computed by a k -NN classifier. The fitness value provided by this kind of supervisor (the music “critic”) models the affect for a certain music genre after a training phase. The early stages of this work have been encouraging since they have responded to the a priori expectations and more work has to be carried out in the future to explore the creative capabilities of the proposed system.

Other applications of the statistical pattern recognition models discussed in this dissertation are envisaged, like the fingerprinting of files containing symbolically encoded music. A hashing function based on statistical representations of the content of a music file, in particular multi-part files, could be developed to assign a unique identifier that could help in fast cataloging and retrieval of music in very large databases.

5.5 Publications

Most parts of this thesis have been published in journals and conference proceedings. Here is a list of papers in inverse chronological order (in brackets, the chapter, or chapters, to which each paper is related):

Journal articles:

- Pedro J. Ponce de León, José M. Iñesta (2007). ”A Pattern Recognition Approach for Music Style Identification Using Shallow Statistical Descriptors”. *IEEE Transactions on Systems Man and Cybernetics C*, 37(2):248-257 (2007) [**Chapter 4**]

Book chapters:

- Pedro J. Ponce de León, José M. Iñesta, David Rizo (2008). "Mining digital music score collections: melody extraction and genre recognition". *Pattern Recognition*, pp. 559-590,. Vienna, Austria. [Chapters 3 & 4]
- Pedro J. Ponce de León, Carlos Pérez-Sancho, José M. Iñesta (2006). Classifier ensembles for genre recognition. *Pattern Recognition: Progress, Directions and Applications*, pages 41–53. [Chapter 4]

Conference proceedings:

- Thomas Lidy, Rudolf Mayer, Andy Rauber, Pedro J. Ponce de León, Antonio Pertusa, José M. Iñesta (2010). "A Cartesian Ensemble of Feature Subspace Classifiers for Music Categorization". In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pp. 279-284. Utrecht, Netherlands. [Chapter 4]
- Rudolf Mayer, Andy Rauber, Pedro J. Ponce de León, Carlos Pérez-Sancho, José M. Iñesta (2010). "Feature Selection in a Cartesian Ensemble of Feature Subspace Classifiers for Music Categorisation". In *Proc. of ACM Multimedia Workshop on Music and Machine Learning (MML 2010)*, pp. 53–56. Florence (Italy) [Chapter 4]
- Thomas Lidy, Andrei Greçu, Andy Rauber, Antonio Pertusa, Pedro J. Ponce de León, José M. Iñesta (2009) "A Multi-Feature-Set Multi-Classifer Ensemble Approach For Audio Music Classification" *Music Information Retrieval Evaluation eXchange (MIREX 2009)*, held in conjunction with the ISMIR 2009 conference in Kobe, Japan. [Chapter 4]
- José M. Iñesta, Pedro J. Ponce de León, J. L. Heredia-Agoiz (2008). "A ground-truth experiment on melody genre recognition in absence of timbre". In *Proc. of the 10th International Conference on Music Perception and Cognition (ICMPC10)*, pp. 758-761. Sapporo, Japan [Chapter 4]
- Pedro J. Ponce de León, David Rizo, Rafael Ramírez. "Melody characterization by a fuzzy rule system". In *Proc. Int. Workshop on Machine Learning and Music, MML 2008*, pp. 35-36, Helsinki, Finland. [Chapter 3]

CHAPTER 5. SUMMARY AND FUTURE PERSPECTIVES

- Pedro J. Ponce de León, David Rizo, Rafael Ramirez, José M. Iñesta (2008). “Melody Characterization by a Genetic Fuzzy System” In *Proceedings of the 5th Sound and Music Computing Conference (SMC 2008)*, pp. 15-23. Berlin, Germany. [Chapter 3]
- Thomas Lidy, Andy Rauber, Antonio Pertusa, Pedro J. Ponce de León, José M. Iñesta. “Audio Music Classification Using A Combination Of Spectral, Timbral, Rhythmic, Temporal And Symbolic Features”. *MIREX 2008 - Music Information Retrieval Evaluation eXchange Genre Classification Contest*. Philadelphia, Pennsylvania, USA [Chapter 4]
- David Espí, Pedro J. Ponce de León, Carlos Pérez-Sancho, David Rizo, José M. Iñesta, Antonio Pertusa (2007). “A cooperative approach to style-oriented music composition”. In *Proceedings of the International Workshop on Artificial Intelligence and Music, MUSIC-AI*, pages 25–36. Hyderabad, India. [Chapter 4]
- Pedro J. Ponce de León, José M. Iñesta, David Rizo (2007). “Towards a human-friendly melody characterization by automatically induced rules”. In *Proceedings of the 8th Int. Conf. on Music Information Retrieval, ISMIR 2007*, pp. 437–440. Vienna, Austria. [Chapter 3]
- David Rizo, Pedro J. Ponce de León, Carlos Pérez-Sancho, Antonio Pertusa, José M. Iñesta (2006). “A Pattern Recognition Approach for Melody Track Selection in MIDI Files”. In *Proceedings of the 7th International Symposium on Music Information Retrieval, ISMIR 2006*, pages 61–66. Victoria, Canada. [Chapter 3]
- Carlos Pérez-Sancho, Pedro J. Ponce de León, José M. Iñesta (2006). “A comparison of statistical approaches to symbolic genre recognition”. In *Proceedings of the International Computer Music Conference, ICMC 2006*, pages 545–550. New Orleans, USA. [Chapter 4]
- F. Moreno-Seco, José M. Iñesta, Pedro J. Ponce de León, Luisa Micó (2006) “Comparison of Classifier Fusion Methods for Classification in Pattern Recognition Tasks”. *Lecture Notes in Computer Science v. 4109 (Structural, Syntactic, and Statistical Pattern Recognition, joint IAPR Workshop, SSPR 2006)*, pp. 705–713. Orlando, USA. [Chapter 4]
- David Rizo, Pedro J. Ponce de León, Antonio Pertusa, Carlos Pérez-Sancho, José M. Iñesta (2006). “Melody track identification in music

symbolic files”. In *Proc. of the 19th Int. Florida Artificial Intelligence Research Society Conference, FLAIRS 2006*. Florida, USA.

[Chapter 3]

- Pedro J. Ponce de León, José M. Iñesta, Carlos Pérez-Sancho (2004). “A shallow description framework for musical style recognition”. *Lecture Notes in Computer Science 3138 (Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, SSPR 2004 and SPR 2004)*, pages 871–879. Lisbon, Portugal. [Chapter 4]
- Pedro J. Ponce de León, José M. Iñesta (2004). “Musical Style Classification from Symbolic Data: A Two Styles Case Study”. *Selected Papers from the Proceedings of the Computer Music Modeling and Retrieval 2003, (Lecture Notes in Computer Science, vol. 2771)*, pp. 167-177 [Chapter 4]
- Pedro J. Ponce de León, José M. Iñesta (2004). “Statistical description models for melody analysis and characterization”. In *Proceedings of the 2004 International Computer Music Conference*, pp. 149-156. University of Miami [Chapters 4]
- Pedro J. Ponce de León, José M. Iñesta (2003). “Feature-Driven Recognition of Music Styles”. *1st Iberian Conference on Pattern Recognition and Image Analysis. (Lecture Notes in Computer Science vol. 2652)*, pp. 773-781. Puerto de Andratx, Mallorca, Spain. [Chapter 4]
- Pedro J. Ponce de León, José M. Iñesta. (2002) “Musical Style Identification Using Self-Organising Maps”. In *Second International Conference on Web Delivering of Music (WEDELMUSIC)*, pp. 82–89, Darmstadt, Germany. [Chapters 4]

Besides, during the elaboration of this thesis, the author has worked in other music information retrieval areas, collaborating in the following publications:

- Carlos Pérez-Sancho, David Rizo, José M. Iñesta, Pedro J. Ponce de León, Stefan Kersten, Rafael Ramírez. Genre classification of music by tonal harmony. *Intelligent Data Analysis*, 14:533–545.
- David Rizo, José M. Iñesta, Pedro J. Ponce de León (2006). “Tree Model of Symbolic Music for Tonality Guessing”. In *Proc. of the*

CHAPTER 5. SUMMARY AND FUTURE PERSPECTIVES

IASTED Int. Conf. on Artificial Intelligence and Applications, AIA 2006, pp. 299-304. Innsbruck, Austria

- David Rizo, José M. Iñesta, Pedro J. Ponce de León, Antonio Pertusa. “Reconocimiento automático de la tonalidad en partituras digitales”. In *Proc. I Workshop sobre Reconocimiento de Formas y Análisis de Imágenes, CEDI 2005*, pp. 53-60. Granada, Spain.

*“I’m saying: to be continued, until we meet again.
Meanwhile, keep on listening and tapping your feet.”*
Count Basie

Appendices



Software tools used/developed

A.1 Third-party tools

jFuzzyLogic

jFuzzyLogic is an open source fuzzy logic library and FCL language implementation. It is written in Java, and accepts FIS definition input files in Fuzzy Control Language (FCL). This is a language for implementing fuzzy logic systems, especially fuzzy control, that was standardized by the IEC 61131-7 norm ([International Electrotechnical Commission \(IEC\), 1997](#)). FCL allows the programmer to specify linguistic terms, fuzzy sets, and rule systems. jFuzzyLogic provides a fuzzy inference engine, parametric optimization algorithms, fuzzy set visualization, and an Eclipse¹ plugin. It is distributed under the Apache License V2.0, GNU Lesser General Public License version 3.0 (LGPLv3). It is available at <http://jfuzzylogic.sourceforge.net>.

JGAP

JGAP stands for *Java Genetic Algorithms Package*. It is a genetic algorithms and genetic programming component provided as an open source Java framework. It also allows building a grid of computers to distribute computation of fitness as well as evolution steps. *JGAP* is distributed under the GNU Lesser Public License² for non-commercial use. The project is located at <http://jgap.sourceforge.net/>.

SOM_PAK

This software package contains all programs necessary for the correct application of the Self-Organizing Map (SOM) algorithm in the visualization of complex experimental data ([Kohonen et al., 1995](#)). The last version is 3.1, from 1995, when development of the package apparently stopped. It contains

¹<http://eclipse.org>

²<http://www.gnu.org/licenses/lgpl.html>

APPENDIX A. SOFTWARE TOOLS USED/DEVELOPED

command line tools, written in C, for training, labeling, and visualizing SOMs.

This program package is copyrighted in the sense that it may be used freely for scientific purposes. However, the package as a whole, or parts thereof, cannot be included or used in any commercial application without written permission granted by its producers. It is available for download at http://www.cis.hut.fi/research/som_lvq_pak.shtml.

WEKA

WEKA is a collection of machine learning algorithms for data mining tasks, developed by the *Machine Learning Group* at the University of Waikato, New Zealand. The algorithms can either be applied directly to a dataset or called from your own Java code (Hall et al., 2009). WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. There is a book (Witten et al., 2011) on data mining associated to the software, as well as online documentation and an active mailing list. WEKA is open source software issued under the GNU General Public License., and available for download at <http://www.cs.waikato.ac.nz/ml/weka/>.

A.2 GUI for melody part selection

The RF melody track classifier presented in section 3.1 has been used as the base model for a graphical user interface for melody part selection. This application can be used to train and test the classifier. In ‘test’ mode, a list of MIDI files are opened, statistical features describing the content of their tracks are extracted and used to feed the classifier. Results from the classifier are gathered and melody part selection results are displayed in a graphical way. The tracks from each MIDI file can be played simultaneously or in ‘solo’ mode. A snapshot of the interface is shown in Figure 3.5 (page 103).

The underlying RF model is trained by just selecting a folder containing MIDI files. Of course, the melody tracks in these files should be tagged accordingly (currently, the tag ‘BBOK’ is used). The trained model can be saved to file, and reloaded subsequently. When a model is loaded in memory, MIDI files can be started to be tagged. Multiple files can be selected at once. The left part of the screen shows the list of MIDI files loaded. The right part shows the result of the classification process, where each track gets its probability to be a melody displayed as a progress bar. By default, probabilities are shown unnormalized. Tracks containing less than a pre-established number of notes are shown with a special ‘Empty’ tag and are

A.3. GUI FOR MUSIC GENRE RECOGNITION

not classified, to distinguish them for track with zero probability. The user can select the whole MIDI file, or any combination of tracks, to be played. A slider control allows to listen to a specific section of the file.

A.3 GUI for music genre recognition

An experimental graphical user interface has been developed to facilitate working on the problem of music genre recognition, using the sliding window and global description approach presented in section 4.3. The main motivation for such a tool is to allow investigate why classification errors occur. The interface allows to select a pre-trained model $(\omega, \delta, \mu, \gamma)$ for classifying selected tracks from MIDI files. Table A.1 shows the currently available choices.

Parameter	choices
ω	[1..100]
δ	[1.. ω] if $\omega < 50$ [1..20] if $\omega \geq 50$
μ	6, 10, 12, or 28 descriptors
γ	Bayes, k-NN, SOM

Table A.1: Model configuration parameter choices available in the genre recognition graphical application.

Currently, one MIDI file at a time can be loaded. Some metadata, if available, about the file and the track selected for classification is shown, including title, author, initial tempo, initial time signature, and overall duration in seconds and bars. A raw textual representation of the classification model currently loaded, the result of the classification, and the statistical description of each extracted window can be displayed. The actual classification of each extracted window is visualized as a row of boxes, and encoded by colors. Each color corresponds to a different genre. The interface allows to play the whole MIDI file, the currently selected track, a particular window, or to play the file from a certain point. A summary of the classification is displayed at the bottom, as the proportion of windows tagged with available genre labels. This is a simple way of combining predicted window labels, giving the user an approximate estimation of the genre for the whole selected track.

A snapshot of the interface (currently in spanish) is shown in Figure 4.25 (page 188).

B

Predefined fuzzy sets

Predefined fuzzy sets in Figure [B.1](#) are defined by hand, according to expert knowledge. They correspond to attributes in the RIPPER-ALL200 fuzzy rule system (see Table [3.31](#)), and are used to fuzzify this rule set according to the procedure discussed in section [3.3.2](#). The last fuzzy sets for some attributes (e.g., TrackNumNotes) are sometimes not drawn for the whole input domain. In these cases, the fuzzy value for the rest of the domain is the same as the value indicated for the last point (for rightmost fuzzy sets).

APPENDIX B. PREDEFINED FUZZY SETS

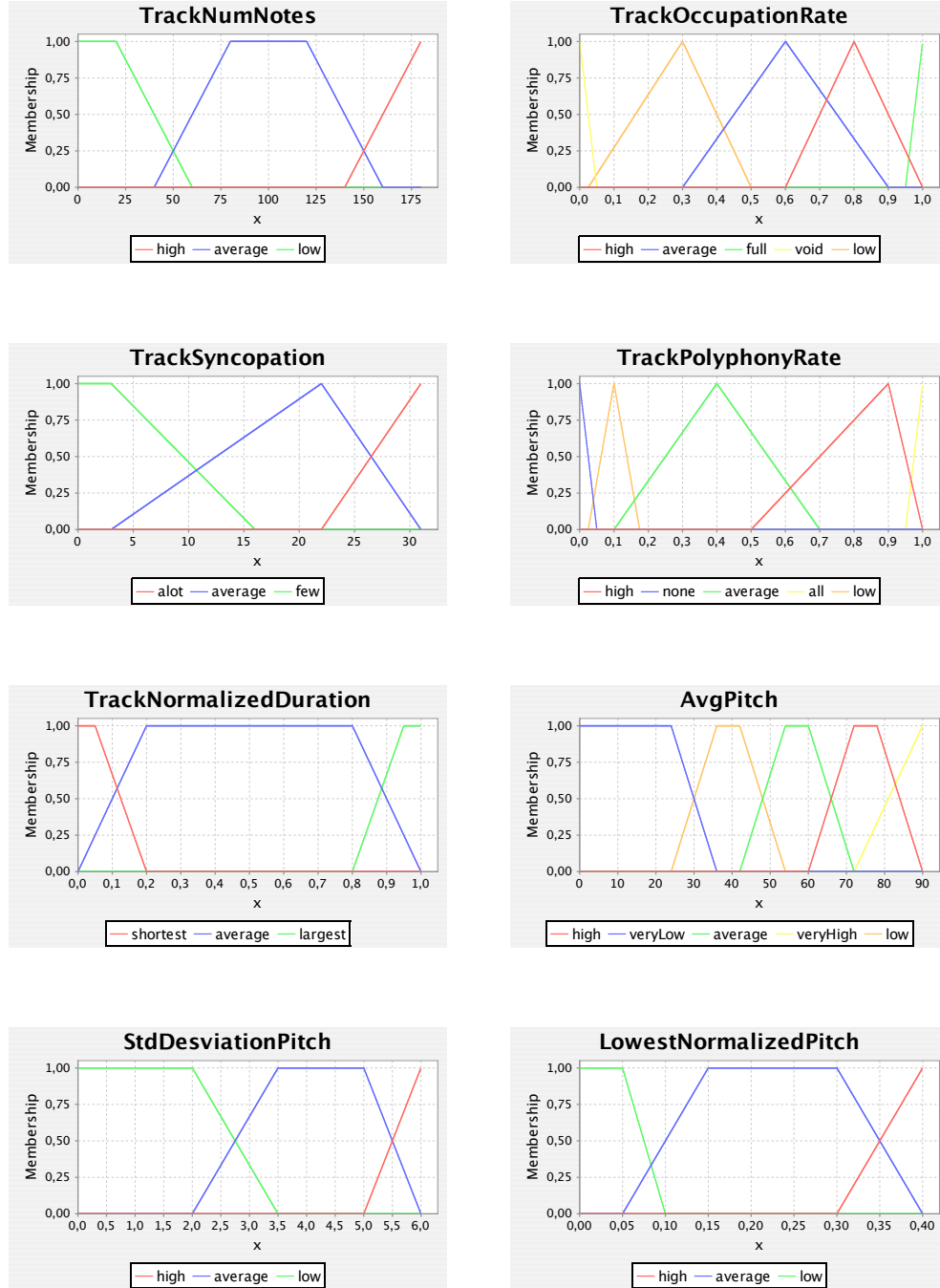


Figure B.1: Predefined fuzzy sets for attributes in the RIPPER-ALL200 fuzzy rule system (see Table 3.31).

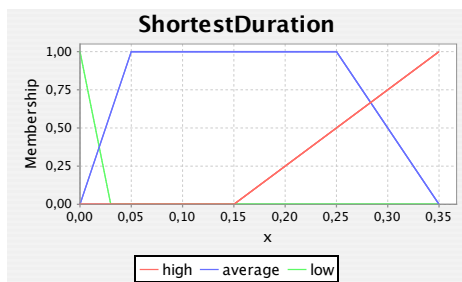
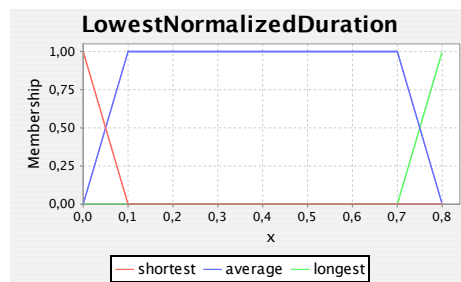
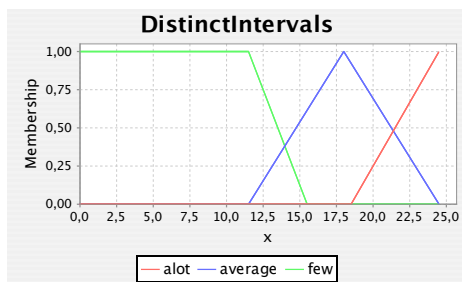
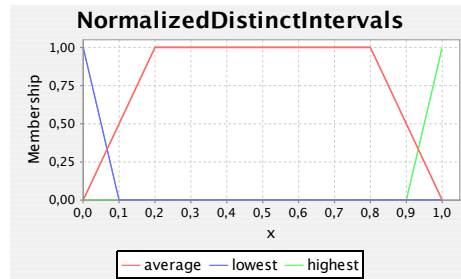
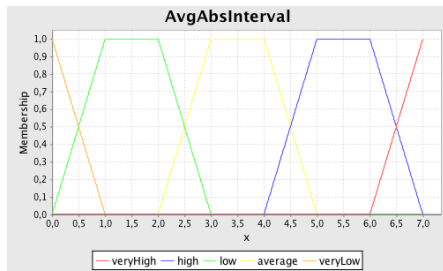
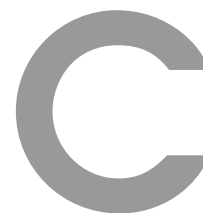


Figure B.1: (cont.) Predefined fuzzy sets for attributes in the RIPPER-ALL200 fuzzy rule system (see Table 3.31)



Resumen en castellano

C.1 Introducción

La recuperación de información musical, o *music information retrieval* (*MIR*), en inglés, es un campo de investigación dedicado a la extracción de información relevante a partir del contenido de obras musicales digitales (Orio, 2006; Typke et al., 2005). En cuanto a la aplicación de técnicas de reconocimiento de formas en este campo, se pueden encontrar dos enfoques principales: la recuperación de información musical a partir del sonido (*audio information retrieval*) (Foote, 1999; Tzanetakis and Cook, 2000b), y la recuperación de información musical desde representaciones simbólicas (principalmente, partituras digitales o representaciones equivalentes) (Downie, 2003).

En la recuperación de información musical desde fuentes sonoras, se procesa la señal digital de audio directamente, donde la información musical no está representada explícitamente, sino que hay que inferirla. Normalmente se utilizan como fuente ficheros en formato WAV o MP3. Por otro lado, cuando se usan representaciones simbólicas, se procesan representaciones que contienen símbolos con un significado musical explícito: notas con altura, duración, intensidad, etc. Nos referimos al tipo de información que podemos encontrar escrita, por ejemplo, en los pentagramas de una típica partitura de música occidental, por lo que llamaremos *partitura digital* a este tipo de representaciones. Los formatos digitales que contienen este tipo de información pueden ser archivos de texto (MusicXML, kern, abc (Good, 2001; Selfridge-Field, 1997)) o archivos binarios como los ficheros MIDI (Selfridge-Field, 1997), entre otros.

Entre algunos de los problemas que actualmente reciben más atención por parte de la comunidad investigadora se encuentran

- La clasificación de obras musicales por género, artista, o cualquier otro conjunto de etiquetas que se pueda asociar a un conjunto de obras:

APPENDIX C. RESUMEN EN CASTELLANO

el objetivo es predecir el género musical de una obra, su compositor, intérprete, pertenencia geográfica, etc.

- Identificación de la melodía: identificar la línea melódica en un fichero musical multi-parte. Se denomina así a aquellos formatos de representación simbólica de la música donde esta se encuentra dividida en secciones (distintos pentagramas en una partitura) o pistas (en el caso de ficheros MIDI).

La melodía y el género musical son los dos conceptos con los que tratamos en este trabajo. Son algunos de los conceptos más imprecisos y ambiguos, a la hora de definirlos, que se pueden encontrar en la teoría musical. A menudo, su definición y ámbito no están claramente establecidos, al menos no de una forma que sea completamente computable. El presente trabajo trata sobre la aplicación de técnicas de reconocimiento de formas a la selección de melodía y al reconocimiento de géneros musicales a partir de música codificada simbólicamente, utilizando descripciones estadísticas del material musical, con el objetivo de capturar la parte computable de los concepto de melodía y género. Ambos problemas van a ser abordados sin utilizar información relacionada con el timbre. De esta forma, se pretende obtener, como objetivo secundario, algunas evidencias sobre la viabilidad de resolver este tipo de problemas usando únicamente información presente en lo que hemos denominado la partitura digital, sin servirnos de ninguna información ajena a ella (instrumentación, metadatos, etc.)¹.

Así pues, en este trabajo nos concentramos en qué es lo que se está tocando (qué notas), y no en cómo suena (cómo se interpreta). Se trata de un enfoque científico, más que ingenieril, donde evitar la información acerca del timbre nos permite estudiar las propiedades intrínsecas del contenido melódico, independientemente de qué instrumento, en última instancia, será utilizado para interpretar dichas notas. Esto es especialmente relevante cuando hablamos de formatos simbólicos como MIDI, donde cambiar de un instrumento a otro no altera el contenido musical en sí, sino sólo algunos parámetros de control. Frente a esos cambios, el contenido musical simbólico almacenado en este tipo de ficheros permanece inalterado (la partitura es la misma), aunque su conversión en sonido pueda variar substancialmente. Como metáfora de este objetivo, piénsese en una situación en que un oyente convenientemente entrenado intenta reconocer el género musical de una serie de obras interpretadas exclusivamente al piano.

¹..., excepto, obviamente, las etiquetas necesarias para aplicar métodos de aprendizaje supervisado.

Los ficheros MIDI son el material con el que se trabaja en el presente trabajo. Una propiedad clave de los llamados ficheros MIDI en *Formato 1* (el más ampliamente utilizado), es que es un formato *multi-pista*. La parte correspondiente a cada uno de los instrumentos que intervienen en la obra suele estar codificada en una pista separada. Si embargo, esto no siempre es así, ya que una misma pista puede contener partes para ser tocadas por instrumentos diferentes en diferentes momentos de la obra, o por el contrario, varias pistas podrían contener partes asignadas al mismo instrumento.

Resolver el problema de seleccionar la pista (o pistas) que contienen la melodía en un fichero MIDI es un paso previo al reconocimiento del género musical, ya que, en este trabajo, utilizaremos principalmente la melodía para identificar el género de una obra. Como el material de trabajo son ficheros MIDI, esto significa desarrollar un sistema capaz de distinguir qué pistas contienen la melodía, considerando al resto como acompañamiento. Una vez conseguido esto, la clasificación de géneros musicales se tratará de resolver extrayendo fragmentos de longitud fija de estas pistas de melodía (a las que llamaremos *ventanas*), de los cuales se intentará reconocer su género. Se investigará el comportamiento de un sistema de reconocimiento de géneros musicales, basado en técnicas de reconocimiento de formas, en función de la longitud de dichos fragmentos. Más adelante, mediante el uso de un transcriptor² automático, se extenderá dicho sistema al reconocimiento de género a partir de audio, donde no existe separación entre melodía y acompañamiento en la representación simbólica resultante.

C.1.1 Selección de partes melódicas

Uno de los problemas abordados en esta tesis es el de encontrar, de forma automática, las partes correspondientes a la melodía en un fichero multi-parte, usando propiedades estadísticas del contenido musical y técnicas de reconocimiento de patrones. Las soluciones a este problema son de aplicación, por ejemplo, en sistemas de consulta donde ésta es con frecuencia un fragmento de la melodía ([Uitdenbogerd and Zobel, 1999](#)). Conocer de antemano qué partes de la obra contienen secciones de la melodía puede reducir drásticamente el ámbito de búsqueda, que ya no tiene que ser la obra completa. También pueden servir en el ámbito de la indexación de grandes colecciones de obras musicales, mediante la generación automática de 'miniaturas' o *thumbnails*, y huellas digitales.

²Sistema capaz de transcribir una señal de audio musical en una partitura.

APPENDIX C. RESUMEN EN CASTELLANO

¿Qué es la melodía?

Existen multitud de definiciones de este concepto en el marco de la teoría musical. La mayoría de ellas coinciden en identificar algunos de los rasgos de la melodía, como su carácter secuencial o monofónico, su papel como referencia dentro de la obra, su relación con la 'letra', su dependencia cultural, etc. Se propone aquí una definición que intenta aunar algunos de estos rasgos:

La *melodía* es un término musical que a menudo se refiere a aquel componente fundamental de una obra musical que captura la mayor parte de la atención del oyente, y a la cual el resto de componentes se subordinan.

Esta definición se enmarca dentro de un contexto general de obras musicales donde hay al menos dos componentes: la melodía y el acompañamiento. En ([Selfridge-Field, 1998](#)) se distinguen diversos tipos de melodías:

- *compuesta* melodías donde hay una única línea melódica a la que pertenecen algunas de las notas, mientras el resto tienden a servir de acompañamiento, siendo el caso más frecuente el caso de música para cuerda sin acompañamiento.
- *auto-acompañada* melodías donde algunas de las notas pertenecen tanto a la idea temática como al soporte armónico (o rítmico).
- *inmersa* melodías confinadas a voces interiores.
- *errante* melodías en las cuales el tema migra de una parte a otra.
- *distribuída* melodías en las cuales las notas que la definen están divididas entre diferentes partes, de manera que el prototipo melódico no puede ser aislado en una sola parte.

C.1.2 Reconocimiento del género musical

Esta tarea puede definirse como el problema de asignar automáticamente uno o más géneros a una obra o fragmento de obra musical. La hipótesis de trabajo es que las obras musicales que pertenecen al mismo género deben compartir ciertos rasgos comunes que, una vez identificados, podrían permitir clasificar correctamente nuevas obras por géneros. El problema ha sido abordado en la literatura principalmente mediante la aplicación de técnicas de aprendizaje supervisado, evitando así la necesidad de formular

una definición computable de qué es un género musical. No obstante, se presenta aquí una breve discusión del concepto.

El género musical es una cualidad de la música que la mayoría de los oyentes pueden percibir de forma intuitiva. Es, probablemente, el descriptor musical más popular, ya que es utilizado con frecuencia para describir, categorizar o comparar colecciones, obras o autores. No existe una definición formal de género musical. Es más, a menudo los términos *género* y *estilo* son a menudo tratados como sinónimos. En (Fabbri, 1999), el autor define ambos términos con el objeto de establecer una distinción entre ellos³:

- El género es *“un tipo de música, tal y como es aceptado por una comunidad por cualquier razón o propósito o criterio, es decir, un conjunto de eventos musicales cuyo curso es gobernado por reglas (de cualquier tipo) aceptadas por una comunidad.”*
- El estilo es *“una disposición recurrente de rasgos en eventos musicales que es típica de un individuo (compositor, intérprete), un grupo de músicos, un género, un lugar o un periodo de tiempo.”*

Estas definiciones sugieren que género es un concepto más amplio que el de estilo. En aquel existe la concurrencia de una audiencia que de algún modo acuerda aglutinar bajo el mismo término una colección de obras musicales. Las fronteras entre diferentes géneros son imprecisas. La existencia de *corpora* de referencia, es decir, la definición por extensión de los géneros, parece la única forma de delimitar de forma aproximada dichas fronteras. Aún teniendo en cuenta estas limitaciones, la investigación en este problema merece la pena, ya que proporciona un marco de trabajo donde diversos aspectos de la investigación en recuperación de información musical pueden ser probados y evaluados. Además, los géneros musicales siguen siendo, hoy día una forma efectiva de describir y catalogar a los músicos y su obra.

Las aplicaciones más inmediatas de los resultados de esta investigación son la clasificación, indexación y búsqueda basada en contenido en bibliotecas de música digital, donde se puede encontrar música digitalizada (MP3), secuenciada (MIDI) o representada de forma estructurada (XML). En general, cualquier organización de objetos musicales basada en etiquetas es campo de aplicación para este tipo de sistemas de reconocimiento.

³(trad. libre del autor)

C.2 Selección de partes melódicas

Como se ha indicado en la introducción, la selección de partes melódicas, según el enfoque adoptado en esta tesis, consiste en decidir qué pistas contienen la melodía en un fichero MIDI multi-pista. Como objetivo secundario se plantea el obtener modelos de pistas de melodía de forma objetiva, mediante representaciones legibles. Para conseguirlo, se aplica una metodología clásica de reconocimiento de formas a los datos de entrada, integrando un mínimo de conocimiento *a priori*, cuyo uso se limita a definir el universo de discurso.

Las técnicas de aprendizaje utilizadas son los árboles de decisión, algoritmos de inferencia de reglas y lógica difusa. La representación del contenido musical se realiza mediante descriptores estadísticos de bajo nivel, como medias, modas, desviaciones estándar, etc., aplicadas a diferentes aspectos melódicos, rítmicos y armónicos de las pistas MIDI.

Los modelos obtenidos mediante entrenamiento supervisado asignan una probabilidad de ser melodía a cada pista candidata. Estos modelos son posteriormente procesados para obtener, a partir de ellos, representaciones comprensibles que caracterizan melodías. Las pistas vacías (que no contienen eventos de nota) y las pistas asignadas al canal de percusión⁴ son descartadas como candidatas. El resto de pistas son descritas mediante vectores de valores estadísticos obtenidos a partir del contenido de la pista. Se ha utilizado para este problema un conjunto de descriptores estadísticos basados en características melódicas, rítmicas y armónicas de la secuencia de notas contenida en cada pista, así como algunas características relacionadas con la pista en sí, como su tasa de ocupación, duración, etc. Se utilizan no normalizados y normalizados. En total, se obtienen 34 descriptores estadísticos para cada pista. Cada pista utilizada en el entrenamiento está etiquetada con un valor booleano que indica si se trata o no de una pista de melodía.

Denominamos aquí *caracterización de pista de melodía* al problema de decidir si, dada una pista MIDI, esta contiene una melodía o no. Dado un sistema capaz de resolver este problema, llamaremos *selección de partes melódicas* al problema donde, dado un fichero MIDI, hay que identificar cuales de sus pistas contienen melodía con mayor probabilidad.

Como un fichero puede contener cero, una o más pistas de melodía, una decisión o propuesta del modelo se considera correcta si, al evaluarlo:

⁴El canal MIDI 10 es el canal reservado para percusión o kits de batería, según el estándar General Midi.

C.2. SELECCIÓN DE PARTES MELÓDICAS

1. Al menos una pista del fichero está etiquetada como *melodía* y la pista propuesta es una de ellas.
2. El fichero no contiene pistas de melodía y el modelo así lo propone.

En consecuencia, se pueden dar tres escenarios de error diferentes:

- **Error tipo 1:** Pista de melodía errónea. Existe efectivamente alguna pista de melodía, pero no es la predicha por el modelo.
- **Error tipo 2:** No hay pistas de melodía, pero el modelo a propuesto alguna.
- **Error tipo 3:** Existe alguna pista de melodía pero el modelo no propone ninguna.

En los *corpora* utilizados en los experimentos, los ficheros con una sola pista de melodía son mucho más frecuentes que aquellos con cero o más de una pista. Para entrenar y evaluar los modelos propuestos se han creado seis conjuntos de ficheros MIDI. Tres de ellos (llamados JZ200, CL200 y KR200) se han usado como conjuntos de datos de entrenamiento y evaluación. JZ200 contiene ficheros de jazz, CL200 está formado por ficheros de música clásica y KR200 es un conjunto de ficheros de música popular especialmente preparados para karaoke (contienen una parte para ser cantada). Cada uno de ellos consta de 200 ficheros. El resto de conjuntos de ficheros MIDI (llamados JAZ, CLA y KAR) se han usado exclusivamente para validar los modelos. Todos los corpora han sido etiquetados manualmente por músicos expertos. Aunque el proceso de etiquetado se reveló complejo, podemos resumirlo diciendo que, en general, si al menos el 50% de una pista era parte de lo que se percibía como melodía, entonces se etiquetaba como tal.

Se han llevado a cabo cuatro tipos de experimentos:

- Caracterización de pistas de melodía.
- Selección de partes melódicas.
- Especificidad de género en la selección de partes melódicas.
- Especificidad del conjunto de entrenamiento en la selección de partes melódicas (capacidad de generalización).

En el primero de ellos se han utilizado los conjuntos de datos CL200, JZ200 y KR200, usando un esquema de validación cruzada, usando diez

APPENDIX C. RESUMEN EN CASTELLANO

particiones del conjunto de datos para estimar la tasa de acierto del sistema. El clasificador utilizado ha sido *Random Forest*, una combinación de 10 árboles de decisión. Se ha obtenido un 99% de tasa de acierto media para CL200, un 98% para JZ200 y un 96% para KR200. Estos buenos resultados ponen de manifiesto la viabilidad del uso de descriptores estadísticos del contenido musical y árboles de decisión para la caracterización de pistas de melodía.

El análisis de los rasgos de las pistas mal clasificadas llevó a las siguientes conclusiones:

Falsos negativos La mayoría de pistas contenían alguna forma de polifonía; melodía en octavas, melodías a dos voces o partes corales homofónicas. Algunas pistas contenían secciones de acompañamiento.

Falsos positivos El tipo más común eran pistas que ‘emulaban’ a la auténtica pista de melodía, o pistas con secciones monofónicas similares a melodías, como respuestas a la melodía, voces de acompañamiento o de contrapunto.

En los experimentos de selección de melodía, se ha utilizado el mismo sistema y los mismos conjuntos de datos que en el experimento precedente, pero se ha usado un esquema de evaluación *leave-one-out* a nivel de fichero para estimar la tasa de acierto. Para seleccionar una pista de melodía por fichero, se establece el umbral de probabilidad mínimo, p_ϵ , para que una pista sea candidata, a $p_\epsilon = 0.25$. Se han obtenido tasas de acierto del 99% para CL200 y JZ200, y del 84.5% para KR200. en este último la mayoría de errores fueron de tipo 1 y 3.

Se ha investigado la sensibilidad del sistema al parámetro p_ϵ , llegando a la conclusión de que un valor cercano a cero (0.01) produce mejores resultados en general, por lo que se ha usado este valor en los experimentos que siguen.

En el tercer grupo de experimentos, se ha investigado la especificidad de los modelos respecto al género, entrenando el clasificador con muestras de un sólo género, usando los conjuntos de validación (CLA, JAZ, KAR) para comprobar cómo varia la tasa de acierto. Los resultados fueron peor de lo esperado cuando el conjunto de validación era del mismo género que el de entrenamiento. No obstante, en general, los resultados mejoran entrenando el clasificador con el mismo género del conjunto de validación. Esto se ha corroborado al entrenar el clasificador con varios géneros distintos al del conjunto de validación, excepto para música clásica, donde incluir muestras del género en el entrenamiento no supone una mejora significativa.

Por último, se ha realizado un grupo de experimentos entrenando el clasificador con muestras de todos los géneros. Los resultados sobre los

C.2. SELECCIÓN DE PARTES MELÓDICAS

conjuntos de validación muestran que, en general, los resultados mejoran cuando el conjunto de entrenamiento incluye muestras del género de validación, respecto a no incluirlas, en especial para el conjunto KAR. Se ha comprobado, además, que los errores de tipo 3 disminuyen en todos los casos, mientras que los errores de tipo 1 disminuyen para KAR y JAZ. La presencia del género de validación en el conjunto de entrenamiento no tiene un efecto apreciable en los errores de tipo 2. La tasa de acierto media combinada, teniendo en cuenta la cardinalidad de cada conjunto de validación, es del 85%.

C.2.1 Modelización de melodías mediante reglas

Una vez entrenado el clasificador RF, es posible obtener, a partir de los árboles de decisión que conforman el modelo, conjuntos de reglas que caractericen, en este caso, melodías. La metodología utilizada para ello se puede resumir en estos pasos:

Extracción de reglas a partir de ramas de los árboles de decisión.

Simplificación de reglas mediante la poda de antecedentes.

Selección de reglas a partir de un ranking de reglas.

En el primer paso sólo se extraen reglas de aquellas ramas que llevan a un nodo hoja *positivo*, es decir que caracteriza a una muestra como melodía. Las ramas que llevan a un nodo hoja negativo son ignoradas. De cada árbol de decisión se obtiene un conjunto de reglas diferente. Cuando un conjunto de reglas se aplica a una muestra, esta es aceptada si alguna regla del conjunto se cumple. A partir de un RF de K árboles se obtienen por tanto K conjuntos de reglas, que forman un sistema de reglas que asigna una probabilidad $1/r, r \leq K$ de ser melodía a una muestra cuando r conjuntos de reglas del sistema la aceptan.

En una segunda etapa se simplifican las reglas eliminando antecedentes (Quinlan, 1999), lo cual equivale a hacer la regla más general. Por último, de cada conjunto de reglas, se establece un ranking de reglas usando una función de puntuación basada en el número de muestras positivas (en este caso melodías) de un conjunto de validación que cada regla acepta. Se seleccionan, para cada conjunto de reglas, las N mejores del ranking correspondiente. Así, por ejemplo, si $K = 10$ y $N = 3$, tendremos diez conjuntos de reglas con tres reglas cada uno.

Se han realizado una serie de experimentos utilizando diversos conjuntos de entrenamiento para los RF, a los cuales se les ha aplicado la metodología de

APPENDIX C. RESUMEN EN CASTELLANO

obtención de sistema de reglas descrita, que se han aplicado a varios conjuntos de validación. En cuanto a la poda de antecedentes, por ejemplo, usando un conjunto de entrenamiento de unas 15 000 muestras se podaron el 12% del total de antecedentes (unos 25 000). Los resultados de clasificación mediante reglas se han comparado con los obtenidos directamente mediante RF. En cuanto a la categorización de pistas de melodía, los resultados obtenidos son comparables, y en algunos casos mejores, que los obtenidos con RF. Cabe destacar que algunos de los mejores resultados se han obtenido con valores de N pequeños. Especialmente destacables son los resultados con $N = 1$ donde, en algunos experimentos los sistemas de reglas exhibieron un comportamiento comparable a los RF. En cuanto a los experimentos de selección de partes melódicas, los sistemas de reglas produjeron resultados algo inferiores a los RF, excepto en un conjunto de validación de música popular, donde un sistema de reglas con $N = 2, K = 10$ dio resultados algo mejores que un RF, formado éste por diez árboles de decisión con cientos de nodos hoja cada uno.

Utilizando un algoritmo de aprendizaje de reglas como RIPPER (Cohen, 1995) directamente a partir de los datos (en lugar de extraerlas de un RF), se obtienen, en general, sistemas de reglas más compactos. Una de las reglas obtenidas de esta forma es la siguiente:

$$(TrackOccupationRate \geq 0.51) \wedge (TrackNumNotes \geq 253) \wedge (AvgPitch \geq 65.0) \\ \wedge (AvgAbsInterval \leq 3.64) \implies IsMelody = \text{true}$$

El problema con estas reglas es que, aunque se pueden leer perfectamente, su significado no es obvio. Por ejemplo, ¿es 253 un número de notas elevado? Además, siempre existe un problema con este tipo de reglas: si una tiene, por ejemplo 254 notas, pero cumple todos los demás antecedentes, no será considerada como melodía. En un intento de solventar ambos problemas a la vez, se ha utilizado lógica difusa para reescribir estas reglas en términos más comprensibles y, a la vez evitar descartar totalmente muestras candidatas que no cumplen 'por poco' alguna de las condiciones de las reglas. El procedimiento consiste, básicamente, en dos fases: primero, establecer una representación de los datos mediante lógica difusa. La entrada y salida del sistema debe convertirse en variables difusas, lo cual se consigue declarando una serie de términos lingüísticos sobre el dominio de cada atributo de entrada, y ajustando las funciones de pertenencia que definen dichos términos mediante un algoritmo genético. Una vez seleccionado la mejor representación difusa de los dominios de entrada, se aplica un procedimiento que permite convertir las reglas en reglas difusas. Por ejemplo, tras aplicar dicho procedimiento, la regla presentada arriba se convierte en

C.2. SELECCIÓN DE PARTES MELÓDICAS

$$\begin{aligned} & ((\text{AvgPitch IS } \textit{high}) \vee (\text{AvgPitch IS } \textit{veryHigh})) \\ & \wedge (\text{TrackOccupationRate IS NOT } \textit{void}) \\ & \wedge (\text{TrackOccupationRate IS NOT } \textit{low}) \\ & \wedge (\text{AvgAbsInterval IS NOT } \textit{fourth}) \\ & \wedge (\text{AvgAbsInterval IS NOT } \textit{high}) \\ & \wedge (\text{TrackNumNotes IS } \textit{high}) \implies \text{IsMelody IS true} \end{aligned}$$

Se ha investigado el comportamiento de estos sistemas de reglas difusas respecto a los sistemas de reglas originales. En general, los resultados obtenidos mediante reglas difusas en experimentos de categorización de pistas de melodía son peores que los obtenidos con las reglas originales. Así pues, siendo preferible utilizar sistemas de reglas no difusos, los sistemas difusos derivados de aquéllos pueden servir al propósito de hacer más comprensible el tipo de melodía que esas reglas modelan. Este tipo de caracterización de melodías mediante reglas difusas, obtenidas a partir de datos de entrenamiento, ha sido, hasta donde el autor conoce, el primero propuesto en el campo de la recuperación de información musical.

En esta tesis se ha presentado un sistema de selección de partes melódicas basado en un clasificador RF y descriptores estadísticos del contenido musical. Dado un fichero MIDI multi-pista, el sistema selecciona la pista de melodía más probable. La evaluación del sistema sobre conjuntos de datos de diferentes géneros ha puesto de manifiesto que utilizar únicamente datos de entrenamiento del mismo género da en general buenos resultados, siendo una solución más simple que entrenar el modelo con multitud de géneros. En el caso de la música clásica, sin embargo, donde las tasas de acierto son menos elevadas, los resultados no varían significativamente con la presencia de muestras de entrenamiento del mismo género. De hecho, para este género, probablemente sea más adecuado utilizar otro enfoque, como por ejemplo técnicas de extracción de melodía ([Isikhan and Ozcan, 2008](#); [Uitdenbogerd and Zobel, 1998](#)).

Se ha propuesto y discutido un método para obtener sistemas de reglas reducidos para caracterizar pistas de melodía a partir de modelos RF. Estos sistemas obtienen tasas de acierto comparables a las de los RF en los que se basan. Con el objetivo de hacer estas reglas más comprensibles, se ha aplicado un procedimiento de conversión en reglas difusas basado en algoritmos genéticos. Las tasas de acierto usando estas reglas difusas son peores que las de las reglas no difusas, pero los sistemas difusos pueden servir al propósito de hacer más comprensible el tipo de melodía que esas reglas modelan.

C.3 Reconocimiento del género musical

Como se ha mencionado en la introducción, la metodología aplicada al problema de reconocimiento del género musical se basa en dos técnicas:

Ventana deslizante La extracción de información a partir de una pista se realiza usando una *ventana deslizante*. Esta se mueve a lo largo de la secuencia musical, tomando muestras de su contenido de una determinada longitud y a intervalos determinados. Ambos parámetros, longitud de la ventana e intervalo de muestreo se expresan en tiempos o compases.

Descripción estadística global A partir del contenido de cada muestra se calcula un conjunto de descriptores estadísticos. de esta forma, una secuencia musical es descrita por uno o más vectores de descriptores numéricos. Se ha definido un conjunto de 28 descriptores estadísticos. Entre otros, se calcula el rango, la media y la desviación estándar de ciertas características musicales como alturas, duraciones, intervalos, notas no diatónicas, etc. sin tener en cuenta información relacionada con el timbre.

Dado este contexto, se aplican técnicas supervisadas de reconocimiento de patrones para la clasificación de fragmentos musicales por géneros. El conjunto de obras utilizado está compuesto por melodías de jazz y música clásica, codificadas como ficheros MIDI. Se ha realizado un estudio exhaustivo de diferentes algoritmos de clasificación, conjuntos de descriptores y valores de longitud y desplazamiento de ventana. El propósito de esta investigación es comprobar que la metodología propuesta es válida para la clasificación automática por géneros de obras musicales representadas simbólicamente.

Con el objetivo de disponer de resultados con los cuales poder comparar la metodología, se ha realizado un estudio sobre la capacidad del ser humano de diferenciar géneros musicales a partir de melodías monofónicas y en ausencia de timbre. Se encuestó a 149 individuos, a los que se les presentaron 40 fragmentos de melodías de jazz y música clásica, divididos en tres niveles de dificultad. Los fragmentos eran interpretados por un ordenador utilizando un timbre neutro. Se obtuvo una tasa de error del 16%, que se ha tomado como referencia para otros resultados. Se detectó también una correlación negativa del error respecto a la edad y el nivel de estudios generales. Estos resultados sugieren que existe información implícita acerca del género en las secuencias de notas que conforman una melodía, al menos cuando nos referimos a diferenciar la música clásica y el jazz.

C.3.1 Resultados de clasificación de géneros

El espacio de los parámetros de ventana sobre los que se ha realizado la investigación está limitado a ventanas de un tamaño máximo de 100 compases. Fijado el tamaño de ventana ω , para cada valor de desplazamiento posible δ (hasta el tamaño de la ventana) se extrae un conjunto de datos formado por segmentos de pista extraídos de las pistas de melodía de las obras. El objetivo es comprobar como, evolucionan los resultados de clasificación en función de la combinación de ambos parámetros, $\langle \omega, \delta \rangle$. Para ello, la tasa de acierto de clasificación sobre cada conjunto de datos se ha estimado mediante validación cruzada estratificada de 10 particiones. Las particiones se han realizado a nivel de fichero MIDI, para evitar que fragmentos de la misma melodía puedan aparecer a la vez en el conjunto de entrenamiento y el de test.

El primer estudio realizado ha consistido en analizar la potencia discriminatoria de cada descriptor por separado. Con algunos de ellos se han obtenido tasas de acierto superiores al 80% para algún valor ω . Descriptores como el rango de alturas o intervalos presentan un comportamiento más estable en función de ω que el resto. En general los descriptores basados en altura de nota parecen obtener mejores tasas de acierto que aquellos basados en la duración de las notas.

Se ha realizado un ranking de descriptores en función de los resultados obtenidos usándolos por separado, asumiendo que existe independencia estadística entre ellos. El ranking se basa en una medida estadística de la separabilidad entre clases que proporciona cada descriptor. Este ranking a permitido definir cuatro conjuntos reducidos de descriptores, de los cuales se ha estudiado su comportamiento en el espacio ω, δ .

Se han estudiado principalmente dos clasificadores: Bayesiano cuadrático y k -vecinos. Cada experimento se denota $(\omega, \delta, \mu, \gamma)$, donde μ representa un conjunto de descriptores y γ el clasificador utilizado. Para cada conjunto de datos $\langle \omega, \delta \rangle$ se han realizado 12 experimentos de clasificación mediante validación cruzada. La mejor tasa de acierto en término medio, 90%, se obtuvo con un clasificador Bayesiano y un conjunto de 12 descriptores. En el caso del clasificador Bayesiano, todos los conjuntos de descriptores reducidos (6, 10 y 12 descriptores) dieron mejores resultados que utilizar el conjunto completo (28 descriptores). La mejor tasa de acierto para el clasificador Bayesiano fue del 96%, con 10 descriptores y sobre el conjunto de datos $\langle 98, 1 \rangle$. Los peores resultados fueron obtenidos al usar valores pequeños de ω , probablemente debido a la escasez de eventos musicales en fragmentos de pocos compases, que provocan que los descriptores estadísticos sean poco fiables.

APPENDIX C. RESUMEN EN CASTELLANO

Para el clasificador k -vecinos se optó por un valor $k = 1$, tras comprobar que las tasas de acierto para distintos valores de k en conjuntos de datos $\langle \omega, \delta \rangle$ seleccionados aleatoriamente no variaban significativamente. Las tasas de acierto para todos los modelos con $\omega \leq 35$ son comparables. Para valores mayores de ω , los modelos construidos con todos los descriptores produce mejores resultados que utilizar conjuntos reducidos. La mejor tasa de acierto para el k -vecinos, 96%, se obtuvo para el conjunto de datos $\langle 95, 13 \rangle$, con todos los descriptores. La tasa de acierto media del k -vecinos fue del 89% con todos los descriptores, un par de puntos por encima que la de los modelos basados en conjuntos reducidos de descriptores.

En resumen, el clasificador Bayes basado en un conjunto de 12 descriptores y el clasificador k -vecinos con $k = 1$ y usando todos los descriptores, se comportaron de forma similar, obteniendo tasas de acierto medio mejores que el resto de combinaciones.

C.3.2 Combinaciones de clasificadores

Para tratar de mejorar las tasas de acierto obtenidas con clasificadores individuales, la investigación se ha centrado en el uso de combinaciones de clasificadores. En concreto se han evaluado combinaciones de clasificadores basados en distintas reglas de combinación de decisiones, de las cuales tres son aportadas en este trabajo. Los experimentos realizados sobre 19 conjuntos de datos de referencia del repositorio UCI⁵ demuestran la precisión y robustez de dichas combinaciones, y en particular de las reglas propuestas, respecto al uso de clasificadores individuales. Otros experimentos realizados sobre clasificación de géneros musicales, utilizando descriptores estadísticos, corroboran que se trata de una buena elección también en este contexto.

C.3.3 Fusión de dominios simbólico y sonoro

La representación de secuencias musicales simbólicas en ausencia de timbre mediante descriptores estadísticos, propuesta en esta tesis, se ha usado en combinación con descriptores de audio en problemas de reconocimiento de género a partir de audio. Los descriptores simbólicos se han incorporado por medio de un sistema de transcripción que, a partir de un fichero de audio, genera un fichero MIDI. Este sistema no incorpora a la transcripción información de timbre explícita, por lo que el uso de dicha representación estadística es especialmente adecuada en este contexto. Aunque el sistema de transcripción está diseñado para tratar con señales sonoras polifónicas con un

⁵<http://archive.ics.uci.edu/ml/>

C.3. RECONOCIMIENTO DEL GÉNERO MUSICAL

único timbre, se asume que “una transcripción monotímica, inexacta pero sistemática, de una señal de audio multitímica contiene cierta información relacionada con el género musical”.

Se han evaluado dos estrategias de combinación de descriptores: fusión temprana y fusión tardía. La primera, que consiste en entrenar un clasificador con una combinación de conjuntos de descriptores simbólicos y de audio, ha mostrado mejoras consistentes, aunque no significativas, sobre todos los corpora evaluados, con respecto a clasificar utilizando conjuntos de descriptores por separado, o utilizar únicamente conjuntos de descriptores de audio. La comparativa con los resultados obtenidos por otros autores sobre los mismos corpora también es favorable al enfoque aquí presentado.

Algunos resultados, sin embargo, mostraron que la estrategia de fusión temprana era muy sensible a qué conjuntos de descriptores en particular se estaban utilizando. Se postula así como conveniente el uso de una estrategia de fusión tardía, con el objeto de tratar de mejorar los resultados de una forma más robusta. En este tipo de sistemas, y en concreto en aquellos basados en estrategias de consenso, un conjunto de clasificadores es entrenado de forma individual. Dado un nuevo objeto, cada clasificador toma una decisión sobre él, tras lo cual se aplica una estrategia para combinar (consensuar) las decisiones de todos los clasificadores en una sola.

En este trabajo se propone un sistema de fusión tardía denominado “conjunto cartesiano de modelos”, el cual es capaz de combinar diferentes conjuntos de descriptores con diferentes tipos de clasificadores. Este sistema se ha aplicado al problema de la clasificación de géneros musicales combinando descriptores simbólicos y de audio. Se han realizado una serie de experimentos sobre conjuntos de ficheros de audio de referencia, donde se ha mostrado que el rendimiento del sistema, en términos de tasa de acierto, es superior, en general, a los resultados obtenidos mediante fusión temprana. La estrategia QBWWV de combinación de decisiones, una de las propuestas en esta tesis, se ha revelado como la mejor opción, en general, en comparación con otras reglas de combinación. Los resultados obtenidos en los experimentos de clasificación mediante fusión tardía refuerzan la hipótesis de que las combinaciones de clasificadores pueden liberar, en parte, al usuario de la siempre difícil elección del conjunto de descriptores y el clasificador apropiado para un problema determinado.

El conjunto cartesiano de modelos permite además incorporar algoritmos de selección de descriptores. Se ha investigado el uso de este tipo de algoritmos, obteniendo tiempos de entrenamiento y test un orden de magnitud inferiores a la alternativa de usar todos los descriptores. El tamaño de los conjuntos de datos se ha reducido hasta menos del 4% de su tamaño original, en término medio. Esto se ha conseguido manteniendo la tasa de

acierto a un nivel razonable al menos en dos de los cinco conjuntos de datos utilizados. Estos experimentos ha mostrado que se debe buscar una solución de compromiso entre la ganancia en tiempos de entrenamiento y/o test y la pérdida de la potencia discriminatoria del conjunto cartesiano de modelos.

C.4 Conclusiones y desarrollo futuro

En esta tesis se presentan varios sistemas de clasificación para obras musicales que trabajan con contenido musical simbólico y evitan el uso de información relacionada con el timbre. Estos sistemas han demostrado que es posible obtener información sobre el carácter melódico de una parte musical, o sobre su género, basándose únicamente en información estadística extraída de la secuencia de notas que la forman.

En concreto, se presenta un método para identificar automáticamente la melodía en una partitura digital multi-pista. El método ha sido aplicado a ficheros MIDI. El contenido musical se describe mediante vectores de descriptores estadísticos. Mediante aprendizaje supervisado se entrenan modelos capaces de reconocer el carácter melódico de una pista. El sistema se ha probado con música clásica, jazz y música popular (de karaoke), con buenos resultados. Estos han evidenciado que, dada una obra de un género determinado, el modelo construido debe haber sido expuesto a muestras de ese mismo género para obtener mejores resultados a la hora de seleccionar la pista de melodía, debido a la especificidad de las pistas de melodía y acompañamiento en cada género.

A partir de los modelos capaces de identificar pistas de melodía se han obtenido representaciones comprensibles, inferidas automáticamente a partir de los datos, en forma de reglas que caracterizan a dichas pistas. Estas representaciones tienen indudable interés musicológico, ya que son representaciones obtenidas de forma completamente objetiva.

En un futuro próximo se pretende integrar en el método una serie de descriptores basados en entropía, como los propuestos por ([Madsen and Widmer, 2007b](#); [Margulis et al., 2008](#)), además de un proceso de selección de descriptores. Se abordará también la anotación de nuevos conjuntos de obras de diferentes géneros. En este proceso se pretende anotar también otros tipos de pistas, de manera que el método de identificación de melodía puede ser fácilmente adaptado a la identificación de otro tipo de pistas, como pistas de bajo, percusión, etc. En particular el método está actualmente siendo adaptado a la identificación de pistas de bajo. Además se está investigando la combinación de identificación de melodía y de bajo, de manera que la

C.4. CONCLUSIONES Y DESARROLLO FUTURO

probabilidad de una pista de ser melodía, condicione la probabilidad de otras pistas de ser de bajo, y viceversa.

Otro de los problemas abordados en esta tesis es la clasificación automática de géneros musicales, que ha sido explorado desde diferentes ángulos. En primer lugar, se ha considerado el reconocimiento de género de melodías codificadas simbólicamente. Para ello, se ha propuesto una metodología de investigación basada en la definición de un espacio de estudio determinado por la longitud de los fragmentos de secuencia que forman los conjuntos de datos. Para una longitud de fragmento determinada se aplican las fases de extracción y selección de descriptores y clasificación. Se han evaluado varios tipos de clasificadores y conjuntos de descriptores, sobre un problema de dos clases (jazz y música clásica). Se ha presentado también un estudio sobre la capacidad humana de distinguir melodías de estos dos géneros musicales, jazz y música clásica, en ausencia de timbre, que ha servido como referencia para los resultados experimentales.

Los resultados obtenidos corroboran la hipótesis de que las melodías representadas simbólicamente, y excluyendo información sobre el timbre, contienen información sobre el género, ya que los sistemas entrenados con información estadística extraída de fragmentos de dichas melodías son capaces de discriminar entre música clásica y jazz en más del 90% de los casos, en término medio, llegando en algunos casos al 95% de tasa de acierto. Los mejores resultados se han obtenido con fragmentos de larga duración (a partir de 30 compases, aproximadamente). Los descriptores estadísticos que más han contribuido a estos resultados son los basados en las alturas de las notas, y en concreto las desviaciones típicas, frente a las medias o los rangos.

El problema se ha abordado también mediante el uso de combinaciones de clasificadores por votación. Se han aportado tres nuevas reglas de voto ponderado para estos sistemas, con los cuales se han obtenido los mejores resultados de clasificación para la mayoría de experimentos de evaluación realizados sobre diferentes conjuntos de datos (incluyendo problemas de clasificación de géneros musicales).

El esquema de representación de información musical simbólica mediante descriptores estadísticos propuesto en esta tesis se ha combinado con descriptores de audio para resolver el problema de la clasificación de géneros utilizando muestras de audio. Esto se ha logrado introduciendo un sistema de transcripción musical capaz de convertir ficheros de audio en ficheros MIDI. Los descriptores simbólicos y de audio se han combinado mediante dos estrategias: concatenación de descriptores en un solo conjunto de características (fusión temprana) y combinación de clasificadores entrenados en diferentes espacios de características (fusión tardía). Para esta última estrategia se ha propuesto un sistema denominado “conjunto cartesiano de

APPENDIX C. RESUMEN EN CASTELLANO

modelos”, que combina a la vez distintos espacios de descriptores y distintos tipos de clasificadores. Este sistema incluye una etapa de selección de características y de modelos, una vez entrenados.

En ambos enfoques, fusión temprana y fusión tardía, se han obtenido mejores resultados combinando descriptores simbólicos y de audio que utilizando únicamente descriptores de audio. Además la combinación de clasificadores mediante el conjunto cartesiano de modelos a proporcionado mejores resultados que la concatenación de descriptores.

El método basado en clasificar fragmentos musicales descritos estadísticamente se puede extender de varias maneras. Centrándonos en la clasificación de géneros musicales, en primer lugar, las decisiones tomadas para cada fragmento perteneciente a la misma pista, se pueden combinar en una sola que se asigna a la pista completa. Además, estas decisiones se podrían tomar secuencialmente, de manera que la decisión tomada para un fragmento, podría afectar a la decisión tomada para el siguiente fragmento de la misma pista. En segundo lugar, como un fichero MIDI esta formado por una serie de pistas, si estas están pre-etiquetadas, por ejemplo, usando un sistema de identificación de pistas MIDI (melodía, bajo,...) como el propuesto en esta tesis, esta información se puede combinar con las decisiones tomadas para cada pista para decidir el género de la obra musical completa. Esto tiene la ventaja, frente a otros sistemas similares, de que no sólo se obtiene una decisión para la obra completa, sino que se dispone de información pormenorizada sobre el carácter/género/cualidad/etc. de cada pista y de distintos fragmentos o secciones dentro de cada pista.

Bibliography

- D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991. (Cited on page 39).
- Peter Ahrendt. *Music Genre Classification Systems - A Computational Approach*. PhD thesis, Technical University of Denmark, Kongens Lyngby, Denmark, February 2006. URL http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=4438. (Cited on pages 16 and 20).
- A. Arzt, G. Widmer, and S. Dixon. Automatic page turning for musicians via real-time machine listening. In *Proceeding of the 2008 conference on ECAI*, pages 241–245, 2008. (Cited on page 3).
- J.-J. Aucouturier and P. Pachet. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004. (Cited on page 207).
- J.J. Aucouturier and P. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1):83–93, 2003. (Cited on pages 17 and 18).
- E. Backer and P. Kranenburg. On musical stylometry—a pattern recognition approach. *Pattern Recognition Letters*, 26(3):299–309, 2005. (Cited on page 21).
- D. Bainbridge and T. Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001. (Cited on page 3).
- R. Basili, A. Serafini, and A. Stellato. Classification of musical genre: a machine learning approach. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004. (Cited on page 22).
- P. Bellini, I. Bruno, and P. Nesi. Assessing optical music recognition tools. *Computer Music Journal*, 31(1):68–93, 2007. (Cited on page 3).
- Pierfrancesco Bellini, Ivan Bruno, and Paolo Nesi. Optical music recognition: Architecture and algorithms. In Kia Ng and Paolo Nesi, editors, *Interactive Multimedia Music Technologies*, pages 80–110. IGI Global, 2008. (Cited on page 2).
- E. Berdahl, H.C. Steiner, and C. Oldham. Practical hardware and algorithms for creating haptic musical instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME-2008), Genova, Italy*, 2008. (Cited on page 3).

BIBLIOGRAPHY

- T. Bertin-Mahieux, D. Eck, M.I. Mandel, S. Bressler, B. Shinn-Cunningham, D.P.W. Ellis, and R.J. Weiss. Automatic tagging of audio: The state-of-the-art. *Machine Audition: Principles, Algorithms and Systems*. IGI Publishing, 2010. (Cited on page 3).
- S. G. Blackburn. *Content Based Retrieval and Navigation of Music Using Melodic Pitch Contours*. PhD thesis, Department of Electronics and Computer Science, University of Southampton, UK, 2000. URL <http://citeseer.nj.nec.com/blackburn00content.html>. (Cited on page 146).
- C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. URL <http://archive.ics.uci.edu/ml/index.html>. (Cited on page 189).
- A. Blum. Empirical support for winnow and weighted-majority based algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26(1):5–23, 1997. (Cited on page 52).
- Marina Bosi and Richard E. Goldberg. *Introduction to digital audio coding and standards*. (The Kluwer International Series in Engineering and Computer Science). Springer, NY, 2003. (Cited on page 1).
- Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001. (Cited on page 47).
- M.J. Bruderer and M.F. McKinney. Perceptual evaluation of models for music segmentation. In *Proceedings of the 4th Conference on Interdisciplinary Musicology*, page 32, 2008. (Cited on page 232).
- J.J. Burred and A. Lerch. A Hierarchical Approach To Automatic Musical Genre Classification. In *Proc. Of the 6 th Int. Conf. on Digital Audio Effects (DAFx)*, pages 8–11, 2003. (Cited on page 16).
- G. Buzzanca. A supervised learning approach to musical style recognition. In *Music and Artificial Intelligence. Additional Proceedings of the Second International Conference, ICMAI 2002*, 2002. (Cited on page 167).
- Donald Byrd and Megan Schindele. Prospects for improving OMR with multiple recognizers. In *ISMIR 2006, 7th International Conference on Music Information Retrieval*, pages 41–46, 2006. (Cited on page 3).
- P. Cano, E. Gómez, F. Gouyon, P. Herrera, M. Koppenberger, B. Ong, X. Serra, S. Streich, and N. Wack. ISMIR 2004 audio description contest. Technical Report MTG-TR-2006-02, Pompeu Fabra University, 2006. (Cited on pages 73, 75, and 199).

- O. A. S. Carpinteiro. A self-organizing map model for analysis of musical time series. In A. de Padua Braga and T. B. Ludermir, editors, *Proceedings 5th Brazilian Symposium on Neural Networks*, pages 140–145. IEEE Comput. Soc, 1998. (Cited on page 21).
- Zehra Cataltepe, Yusuf Yaslan, and Abdullah Sonmez. Music genre classification using midi and audio features. *EURASIP J. Appl. Signal Process.*, 2007(1): 150–150, 2007. ISSN 1110-8657. (Cited on pages 22 and 159).
- Oscar Celma. Foafing the music: Bridging the semantic gap in music recommendation. In *Proceedings of the 5th International Semantic Web Conference (ISWC2006)*, pages 927–934, 2006. (Cited on page 2).
- W. Chai and B. Vercoe. Folk music classification using hidden Markov models. In *Proc. of the Int. Conf. on Artificial Intelligence*, Las Vegas, USA, 2001. (Cited on page 167).
- Pablo Cingolani. jFuzzyLogic: Open source fuzzy logic (java), August 2008. URL <http://jfuzzylogic.sourceforge.net>. (Cited on page 130).
- William W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. In *International Joint Conference on Artificial Intelligence*, pages 988–994, 1993. (Cited on page 48).
- William W. Cohen. Fast effective rule induction. *Machine Learning: Proceedings of the Twelfth International Conference*, 1995. (Cited on pages 48 and 258).
- Darrell Conklin. Melody classification using patterns. In Christina Anagnostopoulou Rafael Ramirez, Darrell Conklin, editor, *Proc. of the 2nd Int. Workshop on Machine Learning and Music*, pages 31–35, Bled, Slovenia, September 2009. (Cited on pages 3 and 23).
- Arshia Cont. On the creative use of score following and its impact on research. In *Sound and Music Computing*, Padova, Italy, July 2011. URL <http://articles.ircam.fr/textes/Cont11a/>. (Cited on page 3).
- P.R. Cook. *Music, cognition, and computerized sound: an introduction to psychoacoustics*. The MIT Press, 2001. (Cited on page 4).
- D. Cope. *Virtual music: computer synthesis of musical style*. MIT press, 2001. (Cited on page 8).
- David Cope. *Experiments in Musical Intelligence.*, volume 2. Cambridge University Press, New York, NY, USA, 1996. (Cited on pages 11 and 16).

BIBLIOGRAPHY

- Oscar Cordón and Francisco Herrera. A general study on genetic fuzzy systems. In Jenny Smith, editor, *Genetic Algorithms in Engineering and Computer Science*, chapter 3, pages 33–57. John Wiley & Sons, 1995. (Cited on pages [65](#), [81](#), and [123](#)).
- O. Cornelis, R. De Caluwe, G. De Tré, A. Hallez, M. Leman, T. Matthé, D. Moelants, and J. Gansemans. Digitisation of the ethnomusicological sound archive of the royal museum for central africa (belgium). *International Association of Sound and Audiovisual Archives Journal*, 26:35–43, 2005. (Cited on pages [73](#), [77](#), and [199](#)).
- Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley, 1991. (Cited on page [195](#)).
- Alastair J. D. Craft, Geraint A. Wiggins, and Tim Crawford. How many beans make five? the consensus problem in music-genre classification and a new evaluation method for single-genre categorisation systems. In *Proceedings of the Int. Conf. on Music Information retrieval, ISMIR 2007*, pages 73–76, Vienna, Austria, 2007. (Cited on page [159](#)).
- P. Cruz-Alcázar, E. Vidal, and J. C. Pérez-Cortes. Musical style identification using grammatical inference: The encoding problem. In Alberto Sanfeliu and José Ruiz-Shulcloper, editors, *Proceedings of CIARP 2003*, pages 375–382, La Habana, Cuba, 2003. (Cited on pages [3](#), [159](#), and [167](#)).
- P.P. Cruz-Alcázar and E. Vidal. Two grammatical inference applications in music processing. *Applied Artificial Intelligence*, 22(1):53–76, 2008. (Cited on pages [8](#), [16](#), and [23](#)).
- Padraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In *Machine Learning: ECML 2000, 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 109–116. 2000. (Cited on page [52](#)).
- R. Dannenberg, B. Thom, and D. Watson. A machine learning approach to musical style recognition. In *Proceedings of the International Computer Music Conference (ICMC'97)*, pages 344–347, 1997. (Cited on pages [21](#) and [167](#)).
- W.B. De Haas, M. Rohrmeier, R.C. Veltkamp, and F. Wiering. Modeling harmonic similarity using a generative grammar of tonal harmony. In *Proceedings of the Tenth International Conference on Music Information Retrieval (ISMIR'09)*, pages 549–554. Citeseer, 2009. (Cited on page [3](#)).
- C. DeCoro, Z. Barutcuoglu, and R. Fiebrink. Bayesian aggregation for hierarchical genre classification. In *Proceedings of the International Conference on Music Information Retrieval*, pages 77–80, 2007. (Cited on page [22](#)).

- A. Dessen, A. Cont, and G. Lemaitre. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *Proc. 11th International Society for Music Information Retrieval Conference (ISMIR'2010)*, 2010. (Cited on page 2).
- T. Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems*, pages 1–15. 2000. (Cited on page 52).
- S. Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, 2007. (Cited on page 4).
- S. Dixon, F. Gouyon, and G. Widmer. Towards characterization of music via rhythmic patterns. In *Proceedings of 5th International Conference on Music Information Retrieval*, Barcelona, Spain, 2004. (Cited on page 207).
- P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of simple bayesian classifier. *Machine Learning*, 29:103–130, 1997. (Cited on pages 39 and 195).
- Markus Dopler, Markus Schedl, Tim Pohle, and Peter Knees. Accessing Music Collections via Representative Cluster Prototypes in a Hierarchical Organization Scheme. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, PA, USA, September 14–18 2008. (Cited on page 146).
- Shyamala Doraisamy. *Polyphonic Music Retrieval: The N-gram approach*. PhD thesis, Imperial College London, London, UK, 2004. (Cited on page 145).
- Shyamala Doraisamy and Stefan Rüger. Robust polyphonic music retrieval with n-grams. *Journal of Intelligent Information Systems*, 21(1):53–70, 2003. ISSN 0925-9902. (Cited on page 194).
- J. Stephen Downie. *Music information retrieval*, chapter 7, pages 295–340. Annual Review of Information Science and Technology. Information Today Books, 2003. (Cited on pages 1 and 249).
- S. Dubnov and G. Assayag. *Mathematics and Music*, chapter 9, pages 147–158. Springer, 2002. (Cited on page 167).
- J. Duchene and S. Leclercq. An optimal transformation for discriminant and principal component analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10(6):978–983, Nov 1988. ISSN 0162-8828. doi: 10.1109/34.9121. (Cited on page 22).
- R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2000. (Cited on pages 37, 39, 45, and 63).

BIBLIOGRAPHY

- D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. *Advances in neural information processing systems*, 20:385–392, 2007. (Cited on page 2).
- Jana Eggink and Guy J. Brown. Extracting melody lines from complex audio. In *ISMIR*, 2004. (Cited on page 14).
- D.P.W. Ellis, B. Whitman, T. Jehan, and P. Lamere. The echo nest musical fingerprint. In *Proc. 11th International Society for Music Information Retrieval Conference (ISMIR'2010)*, 2010. (Cited on page 4).
- D. Espí, P. J. Ponce de León, C. Pérez-Sancho, D. Rizo, J. M. Iñesta, F. Moreno-Seco, and A. Pertusa. A cooperative approach to style-oriented music composition. In *Proc. of the Int. Workshop on Artificial Intelligence and Music, MUSIC-AI*, pages 25–36, Hyderabad, India, 2007. (Cited on pages 8 and 234).
- F. Fabbri. Browsing music spaces: Categories and the musical mind. In *Proceedings of the IASPM Conference*, 1999. (Cited on pages 18 and 253).
- Arthur Flexer, Fabien Gouyon, Simon Dixon, and Gerhard Widmer. Probabilistic Combination of Features for Music Classification. *Proc. ISMIR*, 2006. (Cited on page 207).
- Jonathan Foote. An overview of audio information retrieval. *Multimedia Syst.*, 7(1):2–10, 1999. ISSN 0942-4962. doi: <http://dx.doi.org/10.1007/s005300050106>. (Cited on pages 1 and 249).
- J. A. Freeman and D. M. Skapura. *Neural Networks. Algorithms, Applications, and Programming Techniques*. Addison-Wesley, 1991. (Cited on pages 28 and 30).
- Anders Friberg and Sven Ahlbäck. Recognition of the melody in a polyphonic symbolic score using perceptual knowledge. In *Proceedings of the 4th Conference on Interdisciplinary Musicology*, Thessaloniki, Greece, 2008. (Cited on page 79).
- David Gerhard. *Computationally Measurable Differences Between Speech and Song*. PhD thesis, Simon Fraser University, CA, USA, April 2003. URL <http://www2.cs.uregina.ca/~gerhard/publications/dbg-final-thesis.pdf>. (Cited on page 2).
- Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: Musical information retrieval in an audio database. In *ACM Multimedia*, pages 231–236, 1995. (Cited on pages 9 and 14).
- Édouard Gilbert and Darrell Conklin. A probabilistic context-free grammar for melodic reduction. In *Proceedings of the MUSIC-AI 2007: International Workshop on Artificial Intelligence and Music*, pages 83–94, Hyderabad, India, January 2007. (Cited on page 3).

- E. Gomez, A. Klapuri, and B.Meudic. Melody description and extraction in the context of music content processing. *Journal of New Music Research (JNMR)*, 32(1):23–40, 2003. (Cited on pages 2, 11, and 15).
- Emilia Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, University Pompeu Fabra, Barcelona, Spain, July 2006. URL <http://www.iaa.upf.es/~egomez/thesis/>. (Cited on page 2).
- Emilia Gómez and Perfecto Herrera. Estimating the tonality of polyphonic audio files: cognitive versus machine learning modelling strategies. In *Proceedings of Fifth International Conference on Music Information Retrieval*, Barcelona, Spain, 2004. (Cited on page 4).
- Eva Gómez-Ballester, Luisa Micó, and Jose Oncina. Some approaches to improve tree-based nearest neighbour search algorithms. *Pattern Recognition*, 39(2): 171–179, February 2006. (Cited on page 39).
- Michael Good and Geri Actor. Using musicxml for file interchange. *Web Delivering of Music, International Conference on*, 0:153, 2003. doi: <http://doi.ieeecomputersociety.org/10.1109/WDM.2003.1233890>. (Cited on page 17).
- Michael Good. Musicxml: An internet-friendly format for sheet music. In *XML 2001 Conference Proceedings*, Orlando, FL, December 2001. (Cited on pages 1 and 249).
- Masakata Goto. Development of the rwc music database. In *Proceedings of the 18th International Congress on Acoustics*, volume I, pages 553–556, 2004. (Cited on pages 24 and 74).
- Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical and jazz music databases. In *Proceedings of 3rd International Symposium on Music Information Retrieval*, 2002. (Cited on page 74).
- Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Music genre database and musical instrument sound database. In *Proceedings of the 4th International Conference on Music Information Retrieval*, 2003. (Cited on page 74).
- A. Grecu, T. Lidy, and A. Rauber. Enhancing audio classification with template features and postprocessing existing audio descriptors. Technical report, Music Information Retrieval Evaluation eXchange (MIREX 2009), October 2009. (Cited on page 215).

BIBLIOGRAPHY

- Andrei Greçu. *Musical Instrument Sound Separation: Extracting Instruments from Musical Performances - Theory and Algorithms*. VDM Verlag Dr. Müller, Saarbrücken, Germany, 2008. ISBN 9783836459457. (Cited on pages 2 and 204).
- M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009. (Cited on pages 80 and 242).
- M. Hamanaka, K. Hirata, and S. Tojo. Implementing “a generative theory of tonal music”. *Journal of New Music Research*, 35(4):249–277, 2006. (Cited on page 3).
- G. R. Hancock and A. J. Klockars. The quest for alpha; developments in multiple comparison procedures in the quarter century since games (1971). *Review of Educational Research*, 66(3):269–306, 1996. (Cited on page 186).
- A. S. Hedayat, N. J. A. Sloane, and John Stufken. Orthogonal arrays: Theory and applications. URL <http://www2.research.att.com/~njas/doc/OA.html>. (Cited on page 65).
- A.S. Hedayat, Neil J. A. Sloane, and John Stufken. *Orthogonal Arrays: Theory and Applications*. Springer, first edition, 1999. (Cited on page 65).
- W. Hewlett and E. Selfridge-Field. *Melodic Similarity: Concepts, Procedures, and Applications.*, volume 11 of *Computing in Musicology*. Cambridge: MIT Press, 1998. (Cited on page 12).
- R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90, 1993. (Cited on page 49).
- A. Holzapfel, Y. Stylianou, A.C. Gedik, and B. Bozkurt. Three dimensions of pitched instrument onset detection. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(6):1517–1527, aug. 2010. ISSN 1558-7916. doi: 10.1109/TASL.2009.2036298. (Cited on page 2).
- M. Hontanilla, C. Pérez-Sancho, and J.M. Iñesta. Composer recognition using language models. In *Proc. of Signal Processing, Pattern Recognition, and Applications (SPPRA 2011)*, pages 76–83, Innsbruck, Austria, February 2011. ACTA Press. ISBN 978-0-88986-865-6. (Cited on page 21).
- Holger H. Hoos, Keith A. Hamel, Kai Renz, and Jürgen Kilian. The GUIDO notation format: A novel approach for adequately representing score-level music. In *Proceedings of the International Computer Music Conference, International Computer Music Conference*, pages 451–454, 1998. URL <http://www.cs.ubc.ca/~hoos/Publ/icmc98b.pdf>. author’s preprint. (Cited on page 17).

- X. Hu, J.S. Downie, C. Laurier, M. Bay, and A.F. Ehmann. The 2007 mirex audio mood classification task: Lessons learned. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 462–467. Citeseer, 2008. (Cited on page 21).
- David Huron. Music information processing using the humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002. ISSN 0148-9267. doi: <http://dx.doi.org/10.1162/014892602760137158>. (Cited on page 17).
- Plácido R. Illescas, David Rizo, and José M. Iñesta. Harmonic, melodic, and functional automatic analysis. In *Proceedings of the 2007 International Computer Music Conference*, volume I, pages 165–168, 2007. (Cited on page 4).
- International Electrotechnical Commission (IEC). IEC 1131 - Programmable controllers. Part 7 - Fuzzy Control Programming, January 1997. (Cited on page 241).
- C. Isikhan and G. Ozcan. A survey of melody extraction techniques for music information retrieval. In *Proceedings of the 4th Conference on Interdisciplinary Musicology*, Thessaloniki, Greece, 2008. (Cited on pages 2, 101, and 259).
- ISO/IEC. Information technology – multimedia content description interface – part 10: Schema definition. *ISO/IEC 15938-10:2005*, April 2005. (Cited on page 15).
- I. Karydis, A. Nanopoulos, A. Papadopoulos, E. Cambouropoulos, and Y. Manolopoulos. Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data. In *Proceedings 4th Sound and Music Computing Conference (SMC'2007)*, Lefkada, 2007. URL <http://delab.csd.auth.gr/papers/SMC07knpcm.pdf>. (Cited on page 14).
- Ioannis Karydis. Symbolic music genre classification based on note pitch and duration. *Lecture notes in computer science*, 4152:329–338, 2006. (Cited on page 22).
- Youngmoo E. Kim, Wei Chai, Ricardo Garcia, and Barry Vercoe. Analysis of a contour-based representation for melody. In *ISMIR*, 2000. (Cited on page 11).
- A. Kirke and E.R. Miranda. A survey of computer systems for expressive music performance. *ACM Computing Surveys (CSUR)*, 42(1):1–41, 2009. (Cited on page 3).

BIBLIOGRAPHY

- P.B. Kirlin and P.E. Utgoff. A framework for automated schenkerian analysis. In *ISMIR 2008: proceedings of the 9th International Conference of Music Information Retrieval*, page 363. Lulu. com, 2008. (Cited on page 3).
- J. Kittler, M. Hatef, Robert P.W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998. (Cited on page 217).
- A. Klapuri and M. Davy. *Signal processing methods for music transcription*. Springer-Verlag New York Inc, 2006. (Cited on page 4).
- Peter Knees, Elias Pampalk, and Gerhard Widmer. Artist classification with web-based data. In *Proceedings of the 5th International ISMIR 2004 Conference*, Barcelona, Spain, October 2004. (Cited on pages 2 and 3).
- T. Kohonen. Self-organizing maps. *Proc. IEEE*, 78(9):1464–1480, 1990. (Cited on page 30).
- T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. SOM_PAK, the self-organizing map program package, v:3.1. SOM_PAK 3.1, apr. 1995. URL http://www.cis.hut.fi/research/som_pak. (Cited on pages 37 and 241).
- L. I. Kuncheva. That elusive diversity in classifier ensembles. In *Proc. 1st Iberian Conf. on Pattern Recognition and Image Analysis (IbPRIA'03)*, volume 2652 of *Lecture Notes in Computer Science*, pages 1126–1138. 2003. (Cited on page 52).
- Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004. ISBN 0471210781. URL <http://www.amazon.com/Combining-Pattern-Classifiers-Methods-Algorithms/dp/0471210781>. (Cited on pages 54 and 55).
- P. Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008a. (Cited on pages 2 and 3).
- Paul Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, June 2008b. (Cited on page 16).
- C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller. Part I, II. *IEEE Transactions on systems, man and cybernetics*, 20(2):404–432, 1990. ISSN 0018-9472. (Cited on pages 60, 61, and 124).
- Kyogu Lee. *A System for Chord Transcription, Key Extraction, and Cadence Recognition from Audio Using Hidden Markov Models*. PhD thesis, Stanford University, CA, USA, May 2007. (Cited on page 2).

- M. Leman. Systematic musicology at the crossroads of modern music research. *Systematic and comparative musicology: concepts, methods, findings*, page 89, 2008. (Cited on page 3).
- K. Lemström and J. Tarhio. Searching monophonic patterns within polyphonic sources. In *Proceedings of the RIAO Conference, volume 2*, pages 1261–1278, 2000. URL citeseer.ist.psu.edu/lemstrom00searching.html. (Cited on page 14).
- F. Lerdahl, R. Jackendoff, and R.S. Jackendoff. *A generative theory of tonal music*. The MIT Press, 1996. (Cited on page 3).
- D.J. Levitin. Memory for musical attributes. *Foundations of cognitive psychology: Core readings*, pages 295–310, 2002. (Cited on page 11).
- M. Levy and M. Sandler. Music information retrieval using social tags and audio. *Multimedia, IEEE Transactions on*, 11(3):383–395, 2009. (Cited on pages 2, 3, and 16).
- M. Li and R. Sleep. Improving melody classification by discriminant feature extraction and fusion. In *5th International Conference on Music Information Retrieval (ISMIR)*, 2004. (Cited on page 22).
- Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 282–289, New York, NY, USA, 2003. ACM. ISBN 1-58113-646-3. doi: <http://doi.acm.org/10.1145/860435.860487>. URL <http://doi.acm.org/10.1145/860435.860487>. (Cited on page 207).
- T. Lidy, A. Rauber, A. Pertusa, and J.M. Iñesta. Improving genre classification by combination of audio and symbolic descriptors using a transcription system. In *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 61–66, Vienna, Austria, 2007. (Cited on pages 199, 206, 207, and 208).
- T. Lidy, A. Rauber, A. Pertusa, P. J. Ponce de León, and J. M. Iñesta. Audio music classification using a combination of spectral, timbral, rhythmic, temporal and symbolic features. Music Information Retrieval Evaluation eXchange. Philadelphia, Pennsylvania, USA, September 2008. (Cited on pages 199, 209, and 210).
- T. Lidy, A. Grecu, A. Rauber, A. Pertusa, P. J. Ponce de León, and J. M. Iñesta. A multi-feature-set multi-classifier ensemble approach for audio music classification. Technical report, Music Information Retrieval Evaluation eXchange (MIREX 2009), October 2009. (Cited on pages 199, 204, and 213).

BIBLIOGRAPHY

- Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, pages 34–41, London, UK, September 11–15 2005. ISBN 0-9551179-0-9. (Cited on pages 203, 206, 207, and 208).
- Thomas Lidy and Andreas Rauber. *Machine Learning Techniques for Multimedia*, chapter Classification and Clustering of Music for Novel Music Access Applications, pages 249–285. Number 11 in Cognitive Technologies. Springer, February 2008. (Cited on pages 2 and 146).
- Thomas Lidy, Rudolf Mayer, Andy Rauber, Pedro J. Ponce de León, Antonio Pertusa, and José M. Iñesta. A cartesian ensemble of feature subspace classifiers for music categorization. In J. Stephen Downie and Remco C. Veltkamp, editors, *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 279–284, Utrecht, Netherlands, August 2010a. International Society for Music Information Retrieval. ISBN 978-90-393-53813. (Cited on pages 16, 159, and 199).
- Thomas Lidy, Carlos N. Silla, Jr., Olmo Cornelis, Fabien Gouyon, Andreas Rauber, Celso A. A. Kaestner, and Alessandro L. Koerich. On the suitability of state-of-the-art music information retrieval methods for analyzing, categorizing and accessing non-western and ethnic music collections. *Signal Process.*, 90:1032–1048, April 2010b. ISSN 0165-1684. doi: <http://dx.doi.org/10.1016/j.sigpro.2009.09.014>. URL <http://dx.doi.org/10.1016/j.sigpro.2009.09.014>. (Cited on pages 16, 77, 166, 203, and 216).
- S. Lippens, J. Martens, M. Leman, B. Baets, H. Meyer, and G. Tzanetakis. A comparison of human and automatic musical genre classification. In *Proceedings of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP 2004*, volume 4, pages 233–236, 2004. (Cited on pages 17, 18, and 159).
- Y.L. Lo, C.H. Lee, and C.H. Wang. Scalable multi-feature index structure for music databases. *Information Sciences*, 179(15):2662–2675, 2009. (Cited on page 4).
- Søren Tjagvad Madsen and Gerhard Widmer. Towards a computational model of melody identification in polyphonic music. In *20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 459–464, 2007a. (Cited on page 15).
- S.T. Madsen and G. Widmer. A complexity-based approach to melody track identification in midi files. In *Proceedings of the International Workshop on Artificial Intelligence and Music (MUSIC-AI 2007) held at the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India*, 2007b. (Cited on pages 14, 79, 84, 230, and 264).

- Masoud Makrehchi, Otman A. Basir, and Mohamed Kamel. Generation of fuzzy membership function using information theory measures and genetic algorithm. In Taner Bilgiç, Bernard De Baets, and Okyay Kaynak, editors, *Fuzzy Sets and Systems - IFSA 2003*, volume 2715 of *Lecture Notes in Computer Science*, pages 603–610. Springer, 2003. (Cited on pages [124](#), [138](#), and [230](#)).
- Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning, ICML '97*, pages 211–218, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3. URL <http://portal.acm.org/citation.cfm?id=645526.757762>. (Cited on page [56](#)).
- E.H. Margulis, A.P. Beatty, and C.U.L. Connotea. Musical Style, Psychoaesthetics, and Prospects for Entropy as an Analytic Tool. *Computer Music Journal*, 32(4), 2008. (Cited on pages [21](#), [23](#), [230](#), and [264](#)).
- Alan Marsden. Modelling the perception of musical voices: a case study in rule-based systems. In *Computer Representations and Models in Music*, pages 239–263. Academic Press, 1992. (Cited on page [14](#)).
- R. Mayer, R. Neumayer, and A. Rauber. Combination of audio and lyrics features for genre classification in digital audio collections. In *Proceeding of the 16th ACM international conference on Multimedia*, pages 159–168. ACM, 2008a. (Cited on page [16](#)).
- R. Mayer, R. Neumayer, and A. Rauber. Rhyme and style features for musical genre classification by song lyrics. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08), Philadelphia, PA, USA*, 2008b. (Cited on pages [2](#) and [16](#)).
- Rudolf Mayer, Andreas Rauber, Pedro. J. Ponce de León, Carlos Pérez-Sancho, and José M. Iñesta. Feature selection in a cartesian ensemble of feature subspace classifiers for music categorisation. In *Proceedings of the 3rd International Workshop on Machine Learning and Music*, ACM Multimedia Conference, Firenze, Italy, October 2010. (in press). (Cited on page [199](#)).
- M. Baroni. *Proposal for a Grammar of Melody: The Bach Chorales*. Les Presses de l'Université de Montréal, 1978. (Cited on page [11](#)).
- Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48, 1998. (Cited on page [195](#)).
- D. McEnnis, C. McKay, and I. Fujinaga. jAudio: A feature extraction library. In *International Conference on Music Information Retrieval*. Citeseer, 2005. (Cited on page [4](#)).

BIBLIOGRAPHY

- C. McKay and I. Fujinaga. Combining features extracted from audio, symbolic and cultural sources. In *International Conf. on Music Information Retrieval*, 2008. (Cited on page 159).
- C. McKay, J.A. Burgoyne, J. Hockman, J.B.L. Smith, G. Vigiensoni, and I. Fujinaga. Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features. In *at the Int. Society for Music Information Retrieval Conference.*, Utrecht, Netherlands, 2010. (Cited on pages 16 and 24).
- Cory McKay. *Automatic music classification with jMIR*. PhD thesis, Music Technology Area, Department of Music Research, Schulich School of Music, McGill University, Montreal, January 2010. (Cited on pages 3, 17, 146, and 168).
- Cory McKay and Ichiro Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Int. Conf. on Music Information Retrieval, ISMIR 2004*, pages 525–530, 2004. (Cited on pages 22, 159, and 168).
- Cory McKay and Ichiro Fujinaga. Automatic music classification and the importance of instrument identification. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM05)*, Montreal, Canada, March 2005. (Cited on pages 7 and 168).
- Cory McKay and Ichiro Fujinaga. jSymbolic: A feature extractor for midi files. In *Proceedings of the International Computer Music Conference*, pages 302–305, 2006a. (Cited on page 168).
- Cory McKay and Ichiro Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved? In *International Conference on Music Information Retrieval*, pages 101–106, 2006b. (Cited on pages 18 and 168).
- Cory McKay and Ichiro Fujinaga. Style-independent computer-assisted exploratory analysis of large music collections. *Journal of Interdisciplinary Music Studies*, 1(1):63–85, 2007a. (Cited on page 168).
- Cory McKay and Ichiro Fujinaga. jWebMiner: A web-based feature extractor. In *Proceedings of the International Conference on Music Information Retrieval*, pages 113–114, 2007b. (Cited on pages 2 and 168).
- K. McLeod. Genres, sub-genres, sub-sub-genres, etc.: Sub-genre naming in electronic/dance music. *Journal of Popular Music Studies*, 13:59–76., 2001. (Cited on page 20).
- Klaus Meffert et al. JGAP - Java Genetic Algorithms and Genetic Programming Package, 2008. URL <http://jgap.sf.net/>. (Cited on page 130).

- K.A. Meintanis and F. Shipman. Creating and evaluating multi-phrase music summaries. In *Proc. of ISMIR*, 2008. (Cited on page 4).
- R. Miotto, N. Montecchio, and N. Orio. Content-based cover song identification in music digital libraries. In *Digital Libraries*, pages 195–204. Springer, 2010. (Cited on page 3).
- MIREX. Music Information Retrieval Evaluation Exchange. URL http://www.music-ir.org/mirex/wiki/MIREX_HOME. (Cited on page 4).
- MIREX. Music Information Retrieval Evaluation Exchange, 2007a. URL http://www.music-ir.org/mirex/wiki/2007:Main_Page. (Cited on page 73).
- MIREX. Audio genre classification, 2007b. URL http://www.music-ir.org/mirex/wiki/2007:Audio_Genre_Classification. (Cited on page 199).
- MMA. *Standard MIDI-File Format Spec. 1.1*. MIDI Manufacturers Association, 1990. (Cited on page 17).
- F. Moreno-Seco, J. M. Iñesta, P. J. Ponce de León, and L. Micó. Comparison of classifier fusion methods for classification in pattern recognition tasks. *Lecture Notes in Computer Science*, 4109:705–713, 2006. (Cited on page 189).
- Francisco Moreno-Seco, Luisa Micó, and Jose Oncina. A fast approximately k-nearest-neighbour search algorithm for classification tasks. *Lecture Notes in Computer Science*, 1876:823–831, 2000. (Cited on page 39).
- Francisco Moreno-Seco, Luisa Micó, and Jose Oncina. A modification of the laesa algorithm for approximated k-nn classification. *Pattern Recognition Letters*, 22:1145–1151, 2003. (Cited on page 39).
- D. Müllensiefen and M. Pendzich. Court decisions on music plagiarism and the predictive value of similarity algorithms. *Musicae Scientiae*, 13(1 suppl):257, 2009. (Cited on page 3).
- E. Narmour. *The analysis and cognition of basic melodic structures: The implication-realization model*. University of Chicago Press, 1990a. (Cited on page 3).
- E. Narmour. *The Analysis and Cognition of Basic Melodic Structures*. University Of Chicago Press, 1990b. (Cited on page 11).
- K. Ng, T.V. Pham, B. Ong, and A. Mikroyannidis. Preservation of interactive multimedia performances. *International Journal of Metadata, Semantics and Ontologies*, 3(3):183–196, 2008. (Cited on page 3).

BIBLIOGRAPHY

- D. Opitz and J. Shavlik. Generating accurate and diverse members of a neural-network ensemble. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 535–541, 1996. (Cited on page 52).
- N. Orio, L. Snidaro, and S. Canazza. Semi-automatic metadata extraction from shellac and vinyl discs. *Proceedings of AXMEDIS*, pages 38–45, 2008. (Cited on page 3).
- Nicola Orio. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, 1(1):1–90, 2006. (Cited on pages 1 and 249).
- Nicola Orio and David Rizo. MusiCLEF - evaluation and benchmarking of music search engines. In *Conference on Multilingual and Multimodal Information Access Evaluation*, 2011. (Cited on page 4).
- G. Ozcan, C. Isikhan, and A. Alpkocak. Melody extraction on midi music files. In *Seventh IEEE International Symposium on Multimedia*, page 8, 2005. (Cited on page 86).
- F. Pachet, G. Westermann, and D. Laigre. Musical datamining for EMD. In *Proceedings of the Wedelmusic Conference*, 2001. (Cited on page 166).
- E. Pampalk, S. Dixon, and G. Widmer. Exploring music collections by browsing different views. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR'03)*, pages 201–208, Baltimore, USA, 2003. (Cited on page 2).
- Elias Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Vienna, Austria, March 2006. URL <http://www.ofai.at/~elias.pampalk/publications/pampalk06thesis.pdf>. (Cited on pages 146 and 166).
- Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Comput. Music J.*, 26:27–49, July 2002. ISSN 0148-9267. doi: 10.1162/014892602760137167. URL <http://portal.acm.org/citation.cfm?id=1112823.1112831>. (Cited on pages 2 and 201).
- Derek Partridge and Niall Griffith. Multiple classifier systems: Software engineered, automatically modular leading to a taxonomic overview. *Pattern Analysis and Applications*, 5:180–188, 2002. (Cited on page 52).
- J. Paulus and A. Klapuri. Music structure analysis using a probabilistic fitness measure and an integrated musicological model. In *ISMIR 2008: proceedings of the 9th International Conference of Music Information Retrieval*, page 369. Lulu. com, 2008. (Cited on page 3).

- C. Pérez-Sancho. *Stochastic Language Models for Music Information Retrieval*. PhD thesis, Alicante, Spain, July 2009a. (Cited on page 145).
- C. Pérez-Sancho, J. M. Iñesta, and J. Calera-Rubio. Style recognition through statistical event models. In *Proceedings of the Sound and Music Computing Conference, SMC '04*, 2004. (Cited on pages 187 and 194).
- C. Pérez-Sancho, J. M. Iñesta, and J. Calera-Rubio. Style recognition through statistical event models. *Journal of New Music Research*, 34(4):331–340, December 2005. (Cited on page 159).
- C. Pérez-Sancho, D. Rizo, S. Kersten, and R. Ramírez. Genre classification of music by tonal harmony. In *Proc. Int. Workshop on Machine Learning and Music, MML 2008*, pages 21–22, Helsinki, Finland, 2008. (Cited on page 23).
- C. Perez-Sancho, D. Rizo, and J. M. Iñesta. Genre classification using chords and stochastic language models. *Connection Science*, 21(2 & 3):145–159, May 2009. ISSN 0954-0091. (Cited on pages 23, 167, and 199).
- Carlos Pérez-Sancho. *Stochastic Language Models for Music Information Retrieval*. PhD thesis, University of Alicante, June 2009b. (Cited on pages 73, 167, and 194).
- Carlos Pérez-Sancho, David Rizo, José M. Iñesta, Pedro J. Ponce de León, S. Kersten, and R. Ramirez. Genre classification of music by tonal harmony. *Intelligent Data Analysis*, 14(5):533–545, September 2010. ISSN 1088-467X. (Cited on page 159).
- A. Pertusa. *Computationally efficient methods for polyphonic music transcription*. PhD thesis, University of Alicante, Department of Software and Computing Systems, 2010. (Cited on pages 2, 4, 199, and 204).
- A. Pertusa and J. M. Iñesta. Multiple fundamental frequency estimation using Gaussian smoothness. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Las Vegas, USA, 2008a. ISBN 1-4244-1484-9. (Cited on page 204).
- A. Pertusa and J. M. Iñesta. Multiple fundamental frequency estimation using Gaussian smoothness and short context. In MIREX, multiple f_0 estimation and tracking contest, 2008b. (Cited on page 204).
- A. Pertusa, A. Klapuri, and J. M. Iñesta. Recognition of note onsets in digital music using semitone bands. *LNCS. Proc. of Progress in Pattern Recognition, Image Analysis and Applications. CIARP 2005*, 3773:869–879, November 2005. (Cited on page 203).

BIBLIOGRAPHY

- Jeremy Pickens. A survey of feature selection techniques for music information retrieval. Technical report, Center for Intelligent Information Retrieval, Departament of Computer Science, University of Massachussetts, 2001. (Cited on pages [82](#) and [146](#)).
- John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999. (Cited on page [42](#)).
- P. J. Ponce de León and J. M. Iñesta. Feature-driven recognition of music styles. In *1st Iberian Conference on Pattern Recognition and Image Analysis. LNCS, 2652*, pages 773–781, 2003. (Cited on page [169](#)).
- P. J. Ponce de León, J. M. Iñesta, and C. Pérez-Sancho. *Classifier ensembles for genre recognition*, chapter 3, pages 41–53. Pattern Recognition: Progress, Directions and Applications. Centre de Visió per Computador. Universitat Autònoma de Barcelona, 2006. ISBN 84-933652-6-2. (Cited on page [53](#)).
- Pedro J. Ponce de León, José M. Iñesta, and Carlos Pérez-Sancho. A shallow description framework for musical style recognition. *Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence*, 3138, 2004. (Cited on page [82](#)).
- Pedro J. Ponce de León, David Rizo, and José M. Iñesta. Towards a human-friendly melody characterization by automatically induced rules. In Simon Dixon, David Brainbridge, and Rainer Typke, editors, *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 437–440, Vienna, September 2007. Austrian Computer Society. (Cited on page [143](#)).
- Pedro J. Ponce de León, David Rizo, and Rafael Ramirez. Melody characterization by a fuzzy rule system. In *Proccedings of the Music and Machine Learning Workshop*, July 2008. (Cited on page [143](#)).
- J. Ross Quinlan. Simplifying decision trees. *Int. J. of Man-Machine Studies*, 27 (3):221–248, 1987. (Cited on page [48](#)).
- J. Ross Quinlan. Simplifying decision trees. *Int. J. Human-Computer Studies*, 51 (2):497–510, 1999. (Cited on pages [106](#) and [257](#)).
- A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. In P. Constantopoulos and I. Solvberg, editors, *5th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2001)*, pages 402–414, Darmstadt, Sep 2001. Springer. (Cited on page [203](#)).
- A. Rauber, E. Pampalk, and D. Merkl. Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity. In *Proc. ISMIR*, pages 71–80, 2002. (Cited on page [203](#)).

- A. Rebelo, G. Capela, and J.S. Cardoso. Optical recognition of music symbols. *International journal on document analysis and recognition*, 13(1):19–31, 2010. (Cited on page [2](#)).
- D. Rizo, J.M. Iñesta, and P.J. Ponce de León. Tree model of symbolic music for tonality guessing. In *Proc. of the IASTED Int. Conf. on Artificial Intelligence and Applications, AIA 2006*, pages 299–304, Innsbruck, Austria, 2006a. IASTED, Acta Press. ISBN 0-88986-404-7. (Cited on page [4](#)).
- David Rizo, Jose M Iñesta, and P. J. Ponce de León. Tree model of symbolic music for tonality guessing. In *Proc of the IASTED Int Conf on Artificial Intelligence and Applications AIA 2006*, pages 299–304. Acta Press, 2006b. (Cited on page [170](#)).
- David Rizo, Pedro J. Ponce de León, Carlos Pérez-Sancho, Antonio Pertusa, and José M. Iñesta. A pattern recognition approach for melody track selection in midi files. In Tindale A. Dannenberg R., Lemström K., editor, *Proc. of the 7th Int. Symp. on Music Information Retrieval ISMIR 2006*, pages 61–66, Victoria, Canada, 2006c. ISBN: 1-55058-349-2. (Cited on page [143](#)).
- David Rizo, Pedro J. Ponce de León, Antonio Pertusa, Carlos Pérez-Sancho, and José M. Iñesta. Melody track identification in music symbolic files. In *Proc. of the 19th Int. FLAIRS Conference*. AAAI Press, 2006d. ISBN 978-1-57735-261-7. (Cited on page [143](#)).
- A. Ruppín and H. Yeshurun. MIDI music genre classification by invariant features. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 397–399, Victoria, Canada, 2006. (Cited on page [22](#)).
- Stanley Sadie and George Grove. *The New Grove Dictionary of Music and Musicians*. Macmillan, 1980. (Cited on page [10](#)).
- John W. Sammon, Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, May 1969. (Cited on page [35](#)).
- N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *Signal Processing Magazine, IEEE*, 23(2):133–141, March 2006. ISSN 1053-5888. doi: 10.1109/MSP.2006.1598089. (Cited on pages [16](#) and [20](#)).
- Helmut. Schaffrath. *The Essen Folk Song Collection in the Humdrum Kern Format*. Menlo Park, CA: Center for Computer Assisted Research in the Humanities., 1995. (Cited on page [17](#)).
- Markus Schedl. *Automatically Extracting, Analyzing, and Visualizing Information on Music Artists from the World Wide Web*. PhD thesis, Johannes Kepler

BIBLIOGRAPHY

- University, Linz, Austria, July 2008. URL http://www.cp.jku.at/research/papers/schedl_phd_2008.pdf. (Cited on pages 2 and 3).
- E.D. Scheirer. Bregman's chimerae: Music perception as auditory scene analysis. In *Proc. International Conference on Music Perception and Cognition*. Citeseer, 1996. (Cited on page 4).
- Eleanor Selfridge-Field. *Beyond MIDI: The Handbook of Musical Codes*. MIT Press, 1997. (Cited on pages 1 and 249).
- Eleanor Selfridge-Field. *Conceptual and representational issues in melodic comparison*, volume 11 of *Computing in Musicology*, pages 3–64. Cambridge, Massachusetts: MIT Press, 1998. (Cited on pages 3, 10, 11, and 252).
- K. Seyerlehner, G. Widmer, and D. Schnitzer. From rhythm patterns to perceived tempo. In *Proc. Int. Symp. on Music Information Retrieval, Vienna, Austria*, pages 519–524. Citeseer, 2008. (Cited on page 4).
- Man-Kwan Shan and Fang-Fei Kuo. Music style mining and classification by melody. *IEICE TRANSACTIONS on INFORMATION and SYSTEMS*, 2003. (Cited on pages 21 and 86).
- C.N. Silla and A.A. Freitas. Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 3499–3504. IEEE, 2009. (Cited on page 16).
- C. N. Silla Jr., A. L. Koerich, and C. A. A. Kaestner. The latin music database. In *Proc. ISMIR*, Philadelphia, USA, 2008. (Cited on pages 4, 73, 76, and 199).
- R.K. Sinha, F.S. Machado, and C. Sellman. Digital rights management or discard restrictions on music? DRM, peer-to-peer piracy and the pricing of digital music. *Journal of Marketing*, 74(2):40–54, 2009. (Cited on page 4).
- Cees G. M. Snoek, Marcel Worring, and Arnold W. M. Smeulders. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 399–402, New York, NY, USA, 2005. ACM. ISBN 1-59593-044-2. doi: <http://doi.acm.org/10.1145/1101149.1101236>. URL <http://doi.acm.org/10.1145/1101149.1101236>. (Cited on pages 50 and 51).
- H. Soltau, T. Schultz, M. Westphal, and A. Waibel. Recognition of music types. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-1998)*, pages 1137–1140, 1998. (Cited on pages 159 and 166).

- M. Sordo, O. Celma, M. Blech, and E. Guaus. The Quest for Musical Genres: Do the Experts and the Wisdom of Crowds Agree? In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 255–260, 2008. (Cited on page 18).
- E. Stamatatos and E. Kavallieratou. Music Performer Verification Based on Learning Ensembles. *Lecture notes in computer science*, pages 122–131, 2004. (Cited on pages 3 and 21).
- E. Stamatatos and G. Widmer. Music performer recognition using an ensemble of simple classifiers. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI'2002)*, pages 335–339, Lyon, France, 2002. (Cited on pages 53 and 167).
- B. Stein, M. Koppel, and E. Stamatatos. Plagiarism analysis, authorship identification, and near-duplicate detection pan'07. In *ACM SIGIR Forum*, volume 41, pages 68–71. ACM, 2007. (Cited on pages 3 and 21).
- I.S.H. Suyoto and A.L. Uitdenbogerd. Effectiveness of note duration information for music retrieval. *Lecture notes in computer science*, 3453:265, 2005. (Cited on page 178).
- I.S.H. Suyoto and A.L. Uitdenbogerd. The Effect of Using Pitch and Duration for Symbolic Music Retrieval. In *Proceedings of the 13th Australasian Document Computing Symposium*, page 41, Hobart, Australia, 2008. (Cited on pages 15 and 178).
- Michael Tang, Chi Lap Yip, and Ben Kao. Selection of melody lines for music databases. In *Proceedings of Annual Int. Computer Software and Applications Conf. COMPSAC*, pages 243–248, 2000. (Cited on pages 14, 79, and 143).
- D. Temperley. Communicative pressure and the evolution of musical styles. *Music Perception*, 21(3):313–337, 2004a. (Cited on page 19).
- David Temperley. *The Cognition of Basic Musical Structures*. The MIT Press, 2004b. ISBN 0262701057. (Cited on page 12).
- B. Thom. Unsupervised learning and interactive Jazz/Blues improvisation. In *Proceedings of the AAAI2000*, pages 652–657, 2000. (Cited on pages 21, 167, and 171).
- E. Toch. *The shaping forces in music: an inquiry into the nature of harmony, melody, counterpoint, form*. Dover Publications, 1977. (Cited on page 10).
- Ernst Toch. *La melodía (translation of 'Melodielehre', 1923)*. SpanPress Universitaria, 1997. (Cited on page 10).

BIBLIOGRAPHY

- P. Toiviainen and T. Eerola. Method for comparative analysis of folk music based on musical feature extraction and neural networks. In *III International Conference on Cognitive Musicology*, pages 41–45, 2001. (Cited on pages 146, 167, and 171).
- J. Tsou. RISM Series A, pt. II: Music manuscripts after 1600 (online catalog)(review). *Notes*, 67(4):789–792, 2011. (Cited on page 17).
- D.R. Turnbull, L. Barrington, G. Lanckriet, and M. Yazdani. Combining audio content and social context for semantic music discovery. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 387–394. ACM, 2009. (Cited on page 2).
- R. Typke, F. Wiering, and R.C. Veltkamp. A survey of music information retrieval systems. In *Proc. of the 6th International Conference on Music Information Retrieval*, 2005. (Cited on pages 1 and 249).
- Rainer Typke. *Music Retrieval based on Melodic Similarity*. PhD thesis, Utrecht University, Netherlands, February 2007. URL <http://rainer.typke.org/publications/music-IR-melodic-similarity.pdf>. (Cited on page 3).
- G. Tzanetakis. *Manipulation, Analysis and Retrieval Systems for Audio Signals*. PhD thesis, Computer Science Department, Princeton University, 2002. (Cited on pages 73 and 199).
- G. Tzanetakis and P. Cook. Marsyas: A framework for audio analysis. *Organised sound*, 4(03):169–175, 2000a. (Cited on page 4).
- G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE Transactions on*, 10(5):293–302, Jul 2002a. ISSN 1063-6676. doi: 10.1109/TSA.2002.800560. (Cited on pages 16 and 77).
- G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10:205–210, 2002b. (Cited on page 159).
- G. Tzanetakis, A. Ermolinskyi, and P. Cook. Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research*, 32(2): 143–152, June 2003. (Cited on pages 21, 167, and 187).
- George Tzanetakis and Perry Cook. Audio information retrieval (AIR) tools. In *In Proc. Int. Symposium on Music Information Retrieval*, 2000b. (Cited on pages 1 and 249).

- Alexandra Uitdenbogerd and Justin Zobel. Melodic matching techniques for large music databases. In *Multimedia '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 57–66. ACM Press, 1999. (Cited on pages 9, 79, and 251).
- Alexandra L. Uitdenbogerd and Justin Zobel. Manipulation of music for melody matching. In *Multimedia '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 235–240. ACM Press, 1998. (Cited on pages 2, 14, 79, 86, 101, and 259).
- P. van Kranenburg and E. Backer. Musical style recognition - a quantitative approach. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*, pages 106–107, 2004. (Cited on page 167).
- Tuomas Virtanen. *Sound Source Separation in Monaural Music Signals*. PhD thesis, Tampere University of Technology, Finland, November 2006. URL http://www.cs.tut.fi/sgn/arg/music/tuomasv/virtanen_phd.pdf. (Cited on page 2).
- G.I. Webb. OPUS: An Efficient Admissible Algorithm for Unordered Search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995. (Cited on page 107).
- B. Whitman, G. Flake, and S. Lawrence. Artist detection in music with Minnowmatch. In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 559–568, 2001. (Cited on page 166).
- Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 3rd edition, January 2011. (Cited on page 242).
- Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 856–863. AAAI Press, 2003. (Cited on page 58).
- Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res.*, 5:1205–1224, 2004. ISSN 1532-4435. (Cited on page 59).
- Damian Zanette. Playing by numbers. *Nature*, 453(7198):988–989, 2008. (Cited on page 24).
- Jianjun Zhu, Xiangyang Xue, and Hong Lu. Musical genre classification by instrumental features. In *Int. Computer Music Conference, ICMC 2004*, pages 580–583, 2004. (Cited on page 159).

Reunido el Tribunal que suscribe en el día de la fecha acordó otorgar, por a
la Tesis Doctoral de Don/Dña. Pedro José Ponce de León Amador
la calificación de .

Alicante de de

El Secretario,

El Presidente,

**UNIVERSIDAD DE ALICANTE
CEDIP**

La presente Tesis de D. Pedro José Ponce de León Amador ha sido registrada con el nº
_____ del registro de entrada correspondiente.

Alicante ____ de _____ de _____

El Encargado del Registro,

La defensa de la tesis doctoral realizada por D/D^a Pedro José Ponce de León Amador se ha realizado en las siguientes lenguas: _____ y _____, lo que unido al cumplimiento del resto de requisitos establecidos en la Normativa propia de la UA le otorga la mención de “Doctor Europeo”.

Alicante, _____ de _____ de _____

EL SECRETARIO

EL PRESIDENTE