

MULTI-LABEL FEATURE SELECTION WITH APPLICATION TO MUSICAL INSTRUMENT RECOGNITION

by
Trudie Sandrock

*Dissertation presented for the degree of Doctor of Philosophy in the
Faculty of Economic and Management Sciences at
Stellenbosch University*



Supervisor: Prof. S.J. Steel

December 2013

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 19 November 2013

Abstract

An area of data mining and statistics that is currently receiving considerable attention is the field of multi-label learning. Problems in this field are concerned with scenarios where each data case can be associated with a set of labels instead of only one. In this thesis, we review the field of multi-label learning and discuss the lack of suitable benchmark data available for evaluating multi-label algorithms. We propose a technique for simulating multi-label data, which allows good control over different data characteristics and which could be useful for conducting comparative studies in the multi-label field.

We also discuss the explosion in data in recent years, and highlight the need for some form of dimension reduction in order to alleviate some of the challenges presented by working with large datasets. Feature (or variable) selection is one way of achieving dimension reduction, and after a brief discussion of different feature selection techniques, we propose a new technique for feature selection in a multi-label context, based on the concept of independent probes. This technique is empirically evaluated by using simulated multi-label data and it is shown to achieve classification accuracy with a reduced set of features similar to that achieved with a full set of features.

The proposed technique for feature selection is then also applied to the field of music information retrieval (MIR), specifically the problem of musical instrument recognition. An overview of the field of MIR is given, with particular emphasis on the instrument recognition problem. The particular goal of (polyphonic) musical instrument recognition is to automatically identify the instruments playing simultaneously in an audio clip, which is not a simple task. We specifically consider the case of duets – in other words, where two instruments are playing simultaneously – and approach the problem as a multi-label classification one. In our empirical study, we illustrate the complexity of musical instrument data and again show that our proposed feature selection technique is effective in identifying relevant features and thereby reducing the complexity of the dataset without negatively impacting on performance.

Opsomming

‘n Area van dataontginning en statistiek wat tans baie aandag ontvang, is die veld van multi-etiket leerteorie. Probleme in hierdie veld beskou scenarios waar elke datageval met ‘n stel etikette geassosieer kan word, instelle van slegs een. In hierdie skripsie gee ons ‘n oorsig oor die veld van multi-etiket leerteorie en bespreek die gebrek aan geskikte standaard datastelle beskikbaar vir die evaluering van multi-etiket algoritmes. Ons stel ‘n tegniek vir die simulering van multi-etiket data voor, wat goeie kontrole oor verskillende data eienskappe bied en wat nuttig kan wees om vergelykende studies in die multi-etiket veld uit te voer. Ons bespreek ook die onlangse ontploffing in data, en beklemtoon die behoefte aan ‘n vorm van dimensie reduksie om sommige van die uitdagings wat deur sulke groot datastelle gestel word die hoof te bied. Veranderlike seleksie is een manier van dimensie reduksie, en na ‘n vlugtige bespreking van verskillende veranderlike seleksie tegnieke, stel ons ‘n nuwe tegniek vir veranderlike seleksie in ‘n multi-etiket konteks voor, gebaseer op die konsep van onafhanklike soek-veranderlikes. Hierdie tegniek word empiries ge-evalueer deur die gebruik van gesimuleerde multi-etiket data en daar word gewys dat dieselfde klassifikasie akkuraatheid behaal kan word met ‘n verminderde stel veranderlikes as met die volle stel veranderlikes.

Die voorgestelde tegniek vir veranderlike seleksie word ook toegepas in die veld van musiek dataontginning, spesifiek die probleem van die herkenning van musiekinstrumente. ‘n Oorsig van die musiek dataontginning veld word gegee, met spesifieke klem op die herkenning van musiekinstrumente. Die spesifieke doel van (polifoniese) musiekinstrument-herkenning is om instrumente te identifiseer wat saam in ‘n oudiosnit speel. Ons oorweeg spesifiek die geval van duette – met ander woorde, waar twee instrumente saam speel – en hanteer die probleem as ‘n multi-etiket klassifikasie een. In ons empiriese studie illustreer ons die kompleksiteit van musiekinstrumentdata en wys weereens dat ons voorgestelde veranderlike seleksie tegniek effektief daarin slaag om relevante veranderlikes te identifiseer en sodoende die kompleksiteit van die datastel te verminder sonder ‘n negatiewe impak op klassifikasie akkuraatheid.

Acknowledgements

The road to any doctoral study is lined with many supportive people along the way – this particular study perhaps even more so.

My studies would not have possible without the help of many wonderful people.

First I would like to thank my husband, Herman, for granting me the space and time to fulfil this ambition, and for many weekends spent as a single parent while I was busy working.

Also to my children, who patiently had to live with their mother’s divided attention from time to time, so that “mommy can work on her computer”.

A big thank you to all my friends and family who helped with babysitting during the course of my studies and also provided much-needed moral support, especially my parents and parents-in-law.

And last, but certainly not least, a heartfelt thank you to my supervisor, Prof. Sarel Steel. Without his guidance, support, inspiration, encouragement, patience, knowledge and passion for the subject, none of this would have been possible.

“Without music, life would be a mistake.”

- Friedrich Nietzsche

Contents

1.	CHAPTER 1: INTRODUCTION	
1.1	Statistics as a means of dealing with big data	1
1.2	Statistics as an interdisciplinary field	4
1.3	Lack of benchmark data	4
1.4	Overview of the thesis	5
2.	CHAPTER 2: MUSIC INFORMATION RETRIEVAL	
2.1	Introduction	9
2.2	Music and mathematics – art versus science	10
2.3	Music information retrieval	11
2.4	Musical sound	16
2.4.1	Musical versus non-musical sound	16
2.4.2	Amplitude and duration	18
2.4.3	Pitch	18
2.4.4	Timbre	19
2.5	Digital music	21
2.6	Audio feature extraction	23
2.6.1	Background	23
2.6.2	Theory of Fourier series	24
2.6.3	Discrete Fourier transforms	26
2.6.4	The short-time Fourier transform	27
2.6.5	Spectrograms	28
2.6.6	Other time-frequency representations	29
2.6.7	Extracting features	30
2.6.8	The MPEG7 standard	30
2.7	Commonly used features	31

2.7.1	Temporal centroid	31
2.7.2	Spectral centroid	31
2.7.3	Spectral spread	33
2.7.4	Mel-Frequency Cepstral Coefficients (MFCCs)	34
2.7.5	Energy	35
2.7.6	Zero Crossing	35
2.7.7	Rolloff	36
2.7.8	Flux	37
2.7.9	Flatness coefficients	37
2.7.10	Projection coefficients	37
2.7.11	Harmonic peaks	38
2.7.12	Log Attack Time	38
2.8	Sub-fields of music information retrieval	39
2.9	Music classification	41
2.9.1	Classification of music by emotion	41
2.9.2	Classification of music by genre	44
2.10	Automatic music transcription	46
2.11	Query-by-example	49
2.12	Music synchronisation	51
2.13	Music structure analysis	53
2.14	Performance analysis	55
2.15	Other areas of MIR research	56
2.16	Summary	57

CHAPTER 3: INSTRUMENT RECOGNITION

3.1	Introduction	58
3.2	Timbre revisited	59
3.3	Goal of musical instrument recognition	61
3.4	Challenges in automatic instrument recognition	62
3.5	Instrument recognition: scope and approaches	67
3.5.1	Signal complexity	67
3.5.2	Instrument types	68

3.5.3	Feature extraction	70
3.5.4	Choice of data	71
3.5.5	Taxonomy	72
3.6	Classification methods	76
3.6.1	Commonly used classifiers	76
3.6.2	Support vector machines	76
3.6.3	k-Nearest Neighbours	80
3.6.4	Gaussian mixture models	81
3.6.5	Decision trees	83
3.6.6	Other classifiers	84
3.6.7	Boosting	85
3.6.8	Multi-label methods	85
3.7	Previous work	87
3.8	Related aspects	92
3.8.1	Commonly used features	92
3.8.2	Feature selection in an instrument recognition context	93
3.8.3	Some related applications	96
3.9	Summary	96

CHAPTER 4: MULTI-LABEL LEARNING

4.1	Introduction	98
4.2	Formal definition and notation	99
4.3	Categorisation of multi-label methods	100
4.4	Problem transformation methods	101
4.4.1	Binary relevance	101
4.4.2	Classifier chains	104
4.4.3	Calibrated label ranking	104
4.4.4	Label powerset	106
4.5	Algorithm adaptation methods	108
4.5.1	Multi-label kNN	108
4.5.2	Multi-label C4.5	109
4.5.3	Predictive clustering trees	110

4.5.4	Other algorithm adaptation methods	110
4.6	Ensemble methods	111
4.6.1	Random k-labelsets	111
4.6.2	Ensembles of classifier chains and pruned sets	112
4.6.3	Random forests	113
4.7	Multi-label evaluation measures	113
4.7.1	Overview	113
4.7.2	Example-based measures	114
4.7.3	Label-based measures	116
4.7.4	Rankings-based measures	117
4.8	Other statistics	118
4.9	Multi-label software	119
4.10	Benchmark datasets	120
4.11	Summary	121

CHAPTER 5: MULTI-LABEL FEATURE SELECTION

5.1	Introduction	122
5.2	Aim and benefits of feature selection	123
5.3	Measuring the efficacy of feature selection	129
5.4	General approaches to feature selection	130
5.4.1	Exhaustive subset search	131
5.4.2	Filter approach	131
5.4.3	Wrapper approach	133
5.4.4	Embedded approach	134
5.4.5	Other approaches	134
5.5	Multi-label feature selection	135
5.5.1	Overview of multi-label feature selection	135
5.5.2	Problem transformation approaches	135
5.5.3	“True” multi-label approaches	137
5.6	Multi-label feature selection based on probe variables	139
5.6.1	Probe variables	139
5.6.2	Multi-label feature selection using independent probes	140

5.7	Summary	144
------------	----------------	------------

CHAPTER 6: GENERATING MULTI-LABEL DATA

6.1	Introduction	145
6.2	Previous approaches to simulating multi-label data	147
6.3	A simple approach to simulating multi-label data	150
6.4	Summary	157

CHAPTER 7: RESULTS OF SIMULATION STUDY

7.1	Introduction	158
7.2	Experimental design	159
7.2.1	Study parameters	159
7.2.2	Methodology	161
7.2.3	Hyperparameters of the SVM	163
7.3	Results	164
7.3.1	Scope	164
7.3.2	Size of training data	164
7.3.3	Number of features	165
7.3.4	Ratio between size of training data and number of features	166
7.3.5	Number of labels	166
7.3.6	Label correlations	167
7.3.7	Correlations between features	169
7.3.8	Overall efficiency of feature selection	170
7.3.9	Number of features selected	172
7.3.10	General remarks	175
7.4	Summary	176

CHAPTER 8: APPLICATION TO MUSIC DATA

8.1	Introduction	177
8.2	ISMIS contest data	179
8.3	Definition of data	180
8.3.1	Training data	180
8.3.2	Test data	184
8.3.3	Features	185
8.4	Data characteristics	186
8.4.1	Single instrument, single pitch	187
8.4.2	Mixture pairs with single instruments	207
8.4.3	Dimension reduction	211
8.5	Empirical results	216
8.5.1	Methodology	216
8.5.2	Overall accuracy	217
8.5.3	Hyperparameter choice	219
8.5.4	Feature selection	224
8.5.5	Choice of classifier	231
8.5.6	Feature importance	232
8.6	Summary	239

CHAPTER 9: CONCLUSIONS

9.1	Summary	240
9.2	Directions for further research	242
9.2.1	Feature selection	242
9.2.2	Simulating multi-label data	243
9.2.3	Musical instrument recognition	243

REFERENCES	244
-------------------	------------

APPENDIX A: R PROGRAMS	281	
A.1	Simulation study – main program	281
A.2	Simulation study – data generation	286
A.3	Simulation study – feature selection	288
A.4	Multi-label evaluation measures	290
A.5	Instrument recognition – main program	292
A.6	Instrument recognition – program for data sampling	303
A.7	Instrument recognition – program for feature selection	305
APPENDIX B: DETAILED RESULTS FROM SIMULATION STUDY	307	
B.1	Results of initial simulation runs – Determining a value for SVM hyperparameter C	308
B.2	Detailed results of simulation runs for different parameter configurations – NO feature selection	309
B.3	Detailed results of simulation runs for different parameter configurations – WITH feature selection	313
B.4	Detailed results of simulation runs for different parameter configurations – Number of relevant and irrelevant features selected	317

CHAPTER 1

Introduction

1.1 Statistics as a means of dealing with big data

Statistics can informally be defined as the study of data. One of the earliest developments in the field of statistics was the introduction of the method of least squares by Legendre in the early 1800s. This was followed by developments in probability theory and by the early 20th century, major advances were being made in the fields of multivariate analysis and experimental design. However, many of the theories being developed were not widely known outside the field of theoretical statistics, simply because the computational power to perform complex calculations was not available. A major shift occurred however in the 1970s, when advances in computer technology completely changed the computational capabilities of statisticians, and therewith heralded a whole new era of statistical analysis.

A well-known result in computer science is Moore's Law, which states that the number of transistors on integrated circuits approximately doubles every two years; in other words, the amount of computing power that can be purchased for the same amount of money doubles approximately every two years. While this explains the

increase in computing power experienced over the past few decades, Kryder's Law (Walter, 2005) is often used to illustrate and predict an even greater increase in the storage capacity of computer hard drives. As an example of the massive amount of storage that is easily available, we cite the fact that for less than \$600, a disk drive can be purchased which has the capacity to store all of the world's music (Manyika *et al.*, 2011). The enormous increase in storage capacity, together with the increase in computing power, have largely contributed to the explosion of data that has taken place in recent years. Coupled with developments in multimedia devices such as digital cameras and digital audio players, not to mention the emergence of the internet era, the amount of data generated on a yearly basis has grown to such an extent that in 2007 the world for the first time produced more data than could fit in all of the world's storage and in 2011, twice as much data was produced as can be stored (Baraniuk, 2011). In a 2012 report by the International Data Corporation (IDC), it is predicted that the digital universe will grow by a factor of 300 from 2005 to 2020, from 130 exabytes in 2005 to 40 000 exabytes (or 40 trillion gigabytes) in 2020 (Gantz and Reinsel, 2012).

In business and industry, one of the latest buzz phrases is *big data*. There is no formal definition of what constitutes big data, but it is generally accepted to refer to datasets "whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze" (Manyika *et al.*, 2011). Examples of big data can be found in most industries. For example, the Compact Muon Solenoid (CMS) detector of the Large Hadron Collider at CERN will produce raw measurement data at a rate of 320 terabits per second, which is far beyond the capabilities of current processing and storage systems (Baraniuk, 2011). In its first few weeks of work, the Sloan Digital Sky Survey telescope in New Mexico collected more data than had previously been collected in the entire history of astronomy. A successor, due to come online in Chile in 2016, will collect the same quantity of data every five days (Cukier, 2010). The retail giant Walmart handles more than one million customer transactions on an hourly basis, feeding databases of more than 2.5 petabytes (Cukier, 2010). In 2010, the social networking website Facebook hosted 40 billion photos (Cukier, 2010); today that figure must be substantially higher. All these examples point to one thing: the amount of data in the world is increasing exponentially.

The term *data deluge* has been used to describe this abundance of data. The task of making sense of these vast quantities of data falls in part to statistics and statisticians. Google's chief economist, Hal Varian, has called statistics the sexy job of this decade (Lohr, 2009). Manyika *et al.* (2011) calculate that the United States alone faces a shortage of 140 000 to 190 000 people with deep analytical skills – that is, people who can operate as data scientists.

The crucial need for novel ways of analysing and interpreting big data is therefore clearly apparent. We can therefore expect a wave of innovation driven by big data, and hopefully pioneered by statisticians and other data scientists.

One way in which big datasets can be reduced in terms of complexity, is through feature selection. Many datasets today have hundreds if not thousands of features (or variables) and some way is needed of eliminating noise by filtering out unnecessary information; this is where feature selection comes into play. In this thesis, we will specifically consider the problem of feature selection in a multi-label classification context. In a standard binary classification problem, each example in a dataset is associated with one of two possible labels, while in a multi-class classification problem each example is associated with one label from a possible set of more than two labels. Multi-label classification problems – which are becoming more and more prevalent in an era of digital media – is concerned with scenarios where each example (or data case) can be associated with a set of possible labels instead of just one.

Despite the importance of feature selection to reduce data complexity and the increasing prevalence of multi-label problems, little has been published regarding multi-label feature selection. In this thesis – although we will not work with “big data” as such – we will propose a new technique for performing feature selection in a multi-label context and therefore contribute in a small way to addressing the many challenges inherent in working with big data.

1.2 Statistics as an interdisciplinary field

Statistics is in essence an interdisciplinary field. It is most likely one of the very few fields of study which is essential to possibly every other field of study. Whether your interests stretch to music, astronomy or cricket, statistics can be applied in an analytical way to enhance the body of knowledge in that field (as examples, see Beran, 2004; Feigelson and Babu, 2012 and Kimber and Hansford, 1993). According to the well-known statistician L.J. Savage: “Statistics is basically parasitic: it lives on the work of others. This is not a slight on the subject for it is now recognized that many hosts die but for the parasites they entertain. Some animals could not digest their food. So it is with many fields of human endeavours, they may not die but they would certainly be a lot weaker without statistics.” (Rao, 1997).

This thesis serves as an example of one such collaboration. While the fields of music and mathematical science may be thought of as worlds apart by many, statistical techniques and concepts fit in fairly naturally with the analysis and interpretation of musical data. In this thesis we will specifically address the problem of musical instrument recognition, and use statistical techniques – specifically, multi-label learning as well as our proposed new method for multi-label feature selection – to contribute to the field of music information retrieval (MIR).

1.3 Lack of benchmark data

Benchmark datasets play an important role. Without widely-used benchmark data, it is difficult to objectively compare techniques and / or algorithms, and it is also difficult to evaluate the success of new techniques. This means that it is difficult for researchers to build on previous work of other researchers, so progress is hampered. Although there has been an explosion in the amount of data available worldwide (as discussed in Section 1.1), there are still areas of study where a lack of easily and freely accessible benchmark datasets is hampering the progress being made in these fields. This study stands at the crossroads of two such fields: multi-label learning and instrument recognition.

Since multi-label learning is a fairly new field, the number of available benchmark datasets is still fairly limited. In addition, the few benchmark datasets that are available tend to be limited in terms of certain data characteristics. From a purely theoretical point of view, proposed new multi-label techniques could be evaluated by using simulated multi-label data, but little work has been done with regards to simulating multi-label data – presumably because it is not a straightforward problem. One of the contributions of this thesis therefore is the proposal of a new technique for simulating multi-label data which allows for explicit control over many data characteristics, and which can be very useful for generating multi-label datasets which can be used to evaluate and compare multi-label techniques. In this thesis we will limit our focus to the evaluation of a multi-label feature selection technique, but datasets generated using the proposed new method could be used to objectively compare multi-label classification techniques as well.

The field of musical instrument recognition also suffers from a lack of available benchmark datasets, which is hampering progress in the development of new techniques to address this problem. The creation of suitable benchmark datasets for musical instrument recognition problems is not a statistical task (nor an easy one) and falls outside of the scope of this thesis. In the practical application discussed in Chapter 8 we will therefore use a dataset that has previously been used in a data mining competition.

1.4 Overview of the thesis

We will start with a comprehensive overview of music information retrieval (MIR) in Chapter 2. We will discuss some of the links between mathematics and music, and point out that the field of MIR bridges some of the perceived gaps between mathematics and music. We will formally define MIR and present a short history of the origins of the field, after which we will move on to an overview of the early work in music and statistics. The next part of Chapter 2 is devoted to an overview of the physical concepts of musical sound, and will explain the different elements of musical sound with a specific focus on timbre, which is that element of sound which is responsible for different instruments producing different sound characteristics. We

will briefly explain the way information is captured in digital audio recordings, and then proceed to a detailed explanation of how features are extracted from digital audio. In this regard we will first provide background by discussing the theory of Fourier series and the short-time Fourier Transform (STFT), which is often used as a basis for audio feature extraction. We will then provide definitions and descriptions of some of the most commonly used audio features, including those that will be used in this study. After all of these preliminaries, we will close Chapter 2 with a discussion of some of the main sub-fields of MIR with a specific focus on the classification of music by emotion, the classification of music by genre, automatic music transcription, query-by-example, music synchronisation, music structure analysis and performance analysis.

In Chapter 3, the focus will be on musical instrument recognition, another sub-field of MIR and the main field of application of this thesis. We will briefly revisit the concept of musical timbre and then formally define the goal of musical instrument recognition. This will be followed by a discussion of the challenges inherent in automatic musical instrument recognition problems. Before progressing with a discussion of some previous work in the field, we will define the scope of instrument recognition problems, outline some common approaches to the problem and briefly discuss some of the commonly used classifiers in the field – in this regard, we will touch on support vector machines (SVMs), k-Nearest Neighbours (kNN), Gaussian mixture models (GMMs) and decision trees. We will also mention boosting, and discuss previous multi-label approaches to the instrument recognition problem. In the next section we will discuss some of the relevant previous work in the field. We finish Chapter 3 with a look at some aspects related to instrument recognition, with a specific focus on feature selection in an instrument recognition context.

Chapter 4 defines the multi-label classification problem, and presents a categorisation of different multi-label classification methods into problem transformation methods, algorithm adaptation methods as well as ensemble methods. Each of these categories will then be examined in more detail, with a discussion of the different algorithms in each category. Of specific interest is the binary relevance (BR) problem transformation method, since this is the multi-label method that will be implemented in the remainder of this thesis. Multi-label methods require different evaluation

measures than single-label methods, so after a discussion of the different multi-label algorithms, we will present an overview of the different multi-label evaluation measures. We will also discuss the concepts of label cardinality and label density, which are often used to describe multi-label datasets. We will conclude Chapter 4 with a brief look at some multi-label software as well as some benchmark multi-label datasets.

We will present a brief overview of feature selection in Chapter 5. We will describe the aims and benefits of feature selection, and will also briefly present some ways of measuring the efficacy of feature selection. We will then present an overview of approaches to single-label feature selection as a general introduction to the problem. We will then move on to an overview of multi-label feature selection – a field about which relatively little has been published as yet. In this regard we will first present some approaches to the problem which have been proposed in the literature. Finally we will introduce a new multi-label feature selection method based on the concept of independent probe variables. This constitutes one of the major contributions of this thesis, as it provides a novel way of implementing feature selection in a multi-label context in a way which is easy to implement.

Another major contribution of this thesis is presented in Chapter 6. The importance of benchmark datasets was outlined in Section 1.3, but the available multi-label benchmark datasets tend to be fairly limited in terms of certain data characteristics. Since multi-label learning is a young field, relatively little has as yet been done regarding the simulation of multi-label data, which is a fairly complex problem. In Chapter 6 we will first outline some previous approaches to the simulation of multi-label data, and highlight their shortcomings. We will then present our proposal for simulating multi-label data, which is a fairly simple approach but which allows for a good measure of control over certain data characteristics.

Chapter 7 contains the results of our empirical simulation study. We will present our experimental design and methodology, and then analyse the results by looking at the impact of different data characteristics as well as the efficacy of our proposed feature selection method. We will highlight some interesting – if counter-intuitive – results

from the data simulation process, and will also demonstrate that the proposed feature selection method is very effective.

The results of the empirical instrument recognition study can be found in Chapter 8. We will discuss the origin of the datasets used, and then define and describe the datasets in detail. We will specifically present some characteristics of the data which highlight the complexity of instrument recognition problems. We will then proceed with a discussion of the methodology used in the empirical study, followed by a detailed discussion of results. In particular, we will demonstrate the efficacy of our proposed feature selection method and will also use our proposed selection method to derive a measure of feature importance which can provide interesting direction for further instrument recognition studies, especially when considered at an instrument level.

We will close in Chapter 9 with some conclusions and directions for further research.

CHAPTER 2

Music Information Retrieval

“May not Music be described as the Mathematic of Sense, Mathematic as the Music of reason? The soul of each the same! Thus the musician feels Mathematic, the mathematician thinks Music, - Music the dream, Mathematic the working life, - each to receive its consummation from the other.”

James Joseph Sylvester, 19th century English mathematician

“Mathematics and music, the most sharply contrasted fields of intellectual activity which can be found, and yet related, supporting each other, as if to show forth the secret connection which ties together all the activities of our mind...”

Hermann von Helmholtz, 19th century German physicist

2.1 Introduction

As the quotes above illustrate, many people would not consider music and mathematics to be closely related at all, while many others find them to be cut from the same cloth. The purpose of this chapter is not to discuss the relative merits of these opposing views, but instead to show how music and mathematics come together in the relatively new field of music information retrieval (MIR).

In Section 2.2 we will start with an extremely brief discussion of the relationship between music and mathematics through the ages, and then introduce the field of MIR in Section 2.3. We will also pay particular attention to some of the pioneering works combining music and statistics. In Section 2.4, the concept of musical sound and its various attributes are formalised, with a short overview of digital music given in Section 2.5. In Section 2.6 we discuss audio feature extraction – the process of extracting information that is meaningful for analysis purposes from music data. Some commonly used features in MIR are then discussed in Section 2.7. Several sub-

fields of MIR are introduced in Section 2.8 and in the remainder of the chapter some of these are discussed in more detail.

2.2 Music and mathematics – art versus science

“From ancient Greek times, music has been seen as a mathematical art.” So claim Flood and Wilson in the opening sentence of the preface to the book *Music and Mathematics* (Fauvel *et al.*, 2003).

One of the earliest realisations of the link between music and mathematics is manifested in the legend of Pythagoras and the blacksmith. According to the legend, one day Pythagoras was walking past the blacksmith’s shop and heard the noises of the hammers striking against the anvils. He noticed that occasionally, some of the sounds seemed to be in harmony and on further investigation found that the weights of the hammers were in whole-number ratios to each other (in other words, in proportions 2:1, 3:2, 4:3 and so on) if the sound they produced was harmonious. Pythagoras repeated this experiment at home using differing lengths of strings and subsequently realised that consonant sounds and simple number ratios were correlated.¹ Although the story of the blacksmith is probably largely mythical – indeed, most modern scholars now consider it to be an ancient Middle Eastern folk tale (James, 1993) – these early experiments with strings and numerical ratios laid the foundations for thousands of years of Western music (Isacoff, 2002).

For almost 2000 years from the time of Pythagoras, the close relationship between mathematics and music was assumed as a given. Indeed, in the Middle Ages music was considered to be so closely interlinked with mathematics that they were studied together in what was referred to as the *quadrivium* – basically a division of mathematics into arithmetic, geometry, music and astronomy. Scientists (in the modern day sense of the word) such as Galileo Galilei, Johannes Kepler and Isaac Newton all contributed to research in the field of music theory. Considering some of

¹ These ratios form the basis of the design of instruments such as the piano; however, for many hundreds of years problems relating to tuning according to this insight of Pythagoras attracted the attention of some of the greatest minds of the time such as Galilei and Newton. See Bibby (2003) for an overview of tuning and the (long!) road to equal temperament, or Isacoff (2002) for a more detailed exposition.

the contrapuntal compositions from musicians such as J.S. Bach, they could possibly be called mathematicians in their own right – Bach’s *Goldberg Variations* is a prime example of a composition with a very strong mathematical foundation (Kellner, 1981). However, a clear separation started appearing between mathematics and music around the time of the Industrial Revolution and its counterpart in the arts, the Romantic period, and this separation is discussed – and lamented – at length in James (1993). Around about this time, the focus of science moved from the theoretical to the practical and music went from being regarded as a science to being seen as entertainment only (James, 1993).

These days, many people would probably consider music and mathematics to be on opposite sides of the spectrum. Few people today would see music as science or a “mathematical art” (as Flood and Wilson call it), as indeed few would probably consider mathematics to be an art. Instead, mathematics is regarded as science – complex and intimidating to everyone but a select few. Music, on the other hand, is generally considered an art, a field that appeals to our emotions and can be enjoyed by anyone. Over the past few decades however, the field of music information retrieval (MIR) appears to have bridged at least some of the modern-day gap between mathematics and music.

2.3 Music information retrieval

Music information retrieval is primarily concerned with the reduction of music to a workable data format and then extracting meaningful information from the data. Tzanetakis *et al.* (2002) define MIR as “the process of indexing and searching music collections”. Other terms often used to refer to more or less the same area of study are *music data mining*, *computational musicology*, *machine listening*, *musical audio mining*, *(computational) auditory scene analysis* as well as numerous other terms.

MIR is a relatively young field: having emerged around the 1960s and started maturing in the late 1990s (Wiering, 2007), it really started gaining momentum around the turn of the millennium with the establishment of ISMIR (International Society for Music Information Retrieval). The first annual ISMIR conference was

held in 2000 in Plymouth, Massachusetts, USA, where 35 papers were presented by 63 different authors. By 2012, the ISMIR conference in Oporto, Portugal had increased in size to 101 papers by 264 authors.

Major changes in the way music is distributed and stored, due to new digital technologies, have also enhanced the importance of the MIR field.

MIR is in essence an interdisciplinary field, spanning fields such as music, mathematics, statistics, computer science, engineering, psychology and quite a few others. As Li *et al.* (2011) lament in the preface of *Music Data Mining*: “Learning about music data mining is challenging as it is an interdisciplinary field that requires familiarity with several research areas and the relevant literature is scattered in a variety of publication venues.”

Some of the music-related journals in which MIR publications can be found are:

- Journal of Mathematics and Music
- Journal of New Music Research
- IEEE Transactions on Audio, Speech and Language Processing
- Computer Music Journal
- Computing in Musicology
- Perspectives of New Music

However, because of the multi-disciplinary nature of the field relevant papers are also often published in journals of fields such as statistics, mathematics, engineering and computer science.

Statistics is a field well-suited to dealing with the type of research problems encountered in music information retrieval. Music audio – once reduced to quantifiable data – translates to very big and complex datasets, something that the field of statistics is specifically well-equipped to deal with. Prior to the advent of fast computer processing speeds over the past couple of decades, extracting the relevant data from audio was an almost impossible task. Similarly, before the development of machine learning techniques, there was no easy way of making sense of vast music datasets. Consequently, relatively few applications of statistical methods to music

exist before the turn of the millennium. According to Nettheim (1997), early applications of statistics to Western classical music appeared in the 1930s, while in the 1950s and 1960s information theory was applied to music (albeit not particularly successfully). The development of computer databases of music in the 1980s facilitated a greater amount of statistical work in the field of music.

A good overview of statistical applications in music prior to the advent of machine learning techniques is given by Nettheim (1997). This author also mentions the difficulty of finding publications about statistical applications in music, since they are scattered among a wide variety of sources. He does, however, provide a very good overview of work that has been done in the field up to that point (1997) by researchers in a variety of disciplines ranging from psychology to musicology and many others. A running theme throughout his paper relates to errors made in the correct application and interpretation of statistics by non-statisticians (for example, use of a normal distribution when a Poisson distribution would have been more appropriate, misunderstanding of the nature of chi-square tests and wrong assumptions made regarding correlation). Some of the most interesting applications referred to in his paper are:

- A 1983 study by C.G. Marillier, in which the tonal progressions in Haydn symphonies are analysed and presented graphically, leading to interesting conclusions that would not have been possible without computer assistance.
- A study by Voss and Clarke (1978) claiming that music is well modelled by a $\frac{1}{f}$ process; although this claim was endorsed in two further studies by different authors (Gardner, 1978; Mandelbrot, 1982), Nettheim challenged this claim in one of his earlier papers (Nettheim, 1992).
- Work by the composer Barlow (1980), in which he attempts to parameterise many of the relevant features – such as rhythm, harmony and pitch – of his composition.

Some other statistical techniques used by authors in the studies referred to by Nettheim (1997) are factor analysis, cluster analysis and Markov chains; it seems, however, that the majority of earlier work in the field was limited to the use of descriptive statistics.

One of the seminal early works regarding the use of statistics in musicology, is a book by Jan Beran (2004). Beran is a statistics professor, but also a composer and pianist, which means that the book gives very good insight into both statistics and music (although the level of detail and complexity is somewhat slanted to the statistical and mathematical).

Beran (2004) starts with some general background about the mathematical foundations of music, and then devotes attention to several statistical techniques chapter-by-chapter. In each chapter (and therefore for each statistical technique discussed), he gives a short motivation for why the technique is suitable for use on musical data. He then details the basic principles of the technique, followed by examples of specific applications in music. Some of the techniques discussed, together with examples of applications in music are:

- *Time series analysis.* Since music is by its very nature a sequence of time-ordered events, time series analysis can be important for analysing musical data. Some of the applications described are the analysis and modelling of musical instruments and pitch perception.
- *Markov chains and hidden Markov models.* Musical events can often be categorised into a finite number of categories occurring in a time-sequence, leading to the question of whether the category transitions could be characterised by probabilities. Markov chains and hidden Markov models are a natural way of considering such processes. Applications such as the classification of folk songs by hidden Markov models and reconstructing scores from acoustic signals are presented.
- *Principal component analysis (PCA).* Musical observations often consist of vectors. For instance, in performance analysis, in the observation of different performances an observation can consist of a vector of tempo measurements at separate score onset times. To detect similarities and differences between different performances, principal component analysis can be used to find the most informative projections.
- *Discriminant analysis.* A typical application of discriminant analysis in music is assigning anonymous compositions to a specific time period, or even to a composer. It has also been used to investigate purity of intonation of singing.

- *Cluster analysis.* Some of the applications discussed in Beran (2004) are an investigation of the distribution of notes, with cluster analysis showing a clear separation between early (pre-Bach) music from the rest, and performance analysis according to tempo curves, showing apparent individual styles for the pianists Alfred Cortot and Vladimir Horowitz.
- *Multidimensional scaling (MDS).* Beran (2004) describes two applications: using frequencies of intervals and interval sequences to differentiate between musical time periods, and the use of MDS to study perceptual differences in music (for example, differences between expert and novice music listeners, or perceptual effects of timbre and pitch).

Other chapters in Beran's book are devoted to exploratory data mining in musical spaces, global measures of structure and randomness, hierarchical methods and circular statistics. A comprehensive list of references is also provided.

A 2007 book by David Temperley entitled "*Music and Probability*" focuses on music perception and cognition from a probabilistic perspective. The focus in this book is on the perception of key and the perception of meter, and Temperley (2007) models this using a Bayesian approach.

These early works in music and statistics all contributed in one way or another to the development of the research area of music information retrieval, several sub-fields of which will be discussed in more detail later in this chapter. However, as briefly mentioned before, one of the chief complexities of mining musical data is extracting meaningful information from raw audio signals. This is done via a process called audio feature extraction, and this needs to be explained before MIR sub-fields can be discussed in more detail. The concept of musical sound and attributes will be discussed first, leading into a discussion of audio feature extraction.

2.4 Musical sound

2.4.1 Musical versus non-musical sound

The definition of music as “organised sound” is generally attributed to the French composer Edgard Varèse. At its most basic level, music consists of periodic sounds that start and stop at different moments in time, and can be stored as a recording in either analogue or digital format. Quite substantial transformation is necessary to get musical data into a form suitable for traditional statistical algorithms, even in the case where music already exists in digital format. The first step in extracting information from audio is the feature extraction step, and this is described in Section 2.6. However, some basic concepts of musical sound and tones need to be reviewed first, as these will greatly aid understanding of the features obtained from audio data.

Sound is created when air molecules are set into motion by some kind of vibration. These vibrating air molecules are channelled through the auditory canal to the eardrums, which then vibrate in response and set off a complex series of events in the ear and brain to enable a human to “hear” sound.

In the case of the voice, airflow from the lungs causes the vocal cords to vibrate (see Benade (1990) for a detailed account of this process); musical instruments create vibrations in different ways, depending on the type of instrument. In a string instrument such as the violin or cello, strings are set into vibration by a bow being drawn across them, or by being plucked by the player’s fingers. These strings pass over a bridge at the top end of the instrument, and the vibrations of the strings across the bridge in turn set off vibrations in the body of the instrument from which audible sound then radiates. Woodwinds, such as the flute, have a column of air inside a tube which is then set in motion by the player blowing across the edge of a hole in the side of the instrument. In some other woodwind instruments such as the clarinet, the air is set in motion by blowing into a reed set into the end of the tube. In brass instruments (of which the trumpet is a well-known example), sound is produced by the vibrations of the player’s lips against a mouthpiece connected to the instrument which then set off vibrations in the air column inside the instrument.

Whatever the source of the vibration, the resulting changes in air pressure can be represented as a continuous signal over time.

While all sounds are created by air molecules vibrating, not all sounds are musical. Musical tones have a regular, repeating vibration, distinguishing them from non-musical sounds. The waveform of a door slamming would look very different from that of a guitar string being plucked, as Figure 2.1 shows. In the case of the guitar string, the continuous, regular repeated vibrations are obvious (graphs from <http://www.howmusicworks.org>):

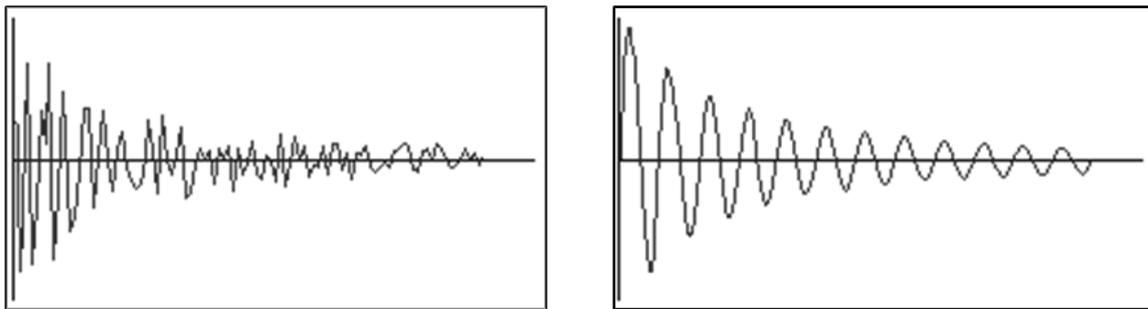


Figure 2.1: Waveforms of a door slamming (left) and a plucked guitar string (right)

Although there can be some form of regularity in non-musical sounds as well, the vibrations are not regular enough for the ear to pick up on and they will therefore not be perceived as musical.

Musical tones or sound waves consist of four main elements:

- Amplitude
- Duration
- Pitch
- Timbre

Each of these will now be discussed in more detail.

2.4.2 Amplitude and duration

Amplitude corresponds to the size of the vibration, and is perceived by the human ear as loudness. Larger vibrations (with a higher amplitude) result in a louder sound.

Duration refers to the length of time for which a tone sounds.

2.4.3 Pitch

The frequency of the sound vibration is generally referred to as the *pitch*, and this is perceived by the ear as how high or low a tone sounds; higher tones have more vibrations per second. Frequencies in music are measured in Hertz (Hz), and it refers to the number of cycles per second in the sound wave. In Western music, pitch is now standardised, with 440Hz corresponding to the A above middle C and is referred to as modern concert pitch.²

A pure tone sounding at a single frequency corresponds to a sine wave, which is the general solution to the second-order differential equation for simple harmonic motion. In other words, any object that is subject to a returning force proportional to its displacement from a given location (such as a string) vibrates as a sine wave. In the case of the human ear, this is also a close approximation of the equation of motion of a particular point on the basilar membrane in the ear, and therefore governs the human perception of sound (Benson, 2008).

Mathematically, the differential equation

$$\frac{d^2y}{dt^2} = -\kappa y$$

² Although this is the ISO standard, some orchestras (notably the Chicago Symphony Orchestra and the New York Philharmonic) use 442Hz while the Berlin and Vienna Philharmonic orchestras use 443Hz (Lerch, 2006). The difference is hard for the human ear to discern, but it does have an effect on timbre.

has the solution

$$y = A \cos \sqrt{\kappa} t + B \sin \sqrt{\kappa} t$$

or

$$y = c \sin(\sqrt{\kappa} t + \phi)$$

This means that a sound wave with frequency ν Hz, peak amplitude c and phase ϕ corresponds to a sine wave of the form

$$c \sin(2\pi\nu t + \phi),$$

or in the case of the modern concert pitch A of 440 Hz

$$c \sin(880\pi t + \phi)$$

shown in Figure 2.2 below with a peak amplitude of 0.7 and phase 0:

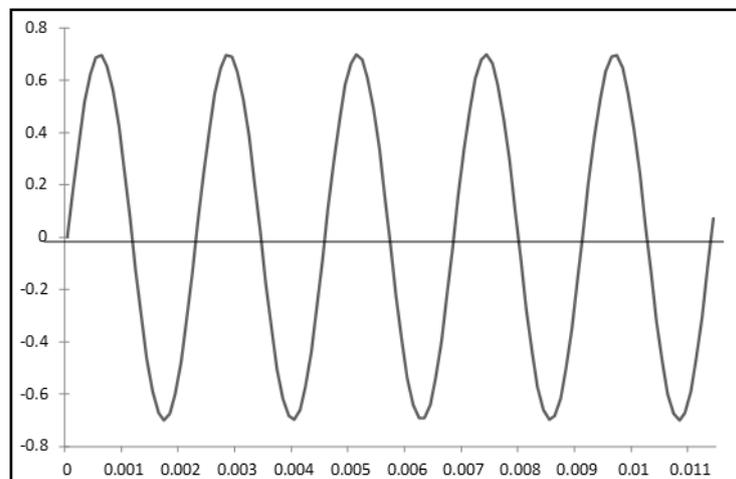


Figure 2.2: A sound wave for concert pitch A, with pitch = 440 Hz, phase = 0 and amplitude = 0.7

2.4.4 Timbre

Timbre is the most difficult aspect of a sound to define in a scientific way. The official definition of timbre by the American Standards Association is “that attribute

of sensation in terms of which a listener can judge that two sounds having the same loudness and pitch are dissimilar” (American Standards Association, 1960). In other words, timbre is defined by what it is not rather than by what it is.

Simply put, timbre is what causes the clarinet to sound different to the flute or the violin even though it is playing the same pitch. It also accounts for the difference in sound when a violin string is plucked rather than bowed.

A sine wave such as the one portrayed in Figure 2.2 above, is the wave of a pure tone at a single frequency. However, the vibrations caused by musical instruments do not occur at a single frequency. Instead, a sound generated by an instrument produces many different vibrations simultaneously. The lowest of these frequencies is called the fundamental frequency, or F_0 , and is equivalent to the pitch of the tone. The other frequencies are usually (but not always) integer multiples of the fundamental frequency, and are called overtones or harmonics. A tone with a fundamental frequency of 200Hz could therefore also have harmonics sounding at 400Hz, 600Hz, 800Hz, 1,000Hz, and so on.

The terms “overtone” and “harmonic” are usually used synonymously. However, the numbering is different. The first harmonic corresponds to the fundamental frequency (F_0), with subsequent frequencies numbered 2, 3, etc. The first overtone is considered to be the first frequency above the fundamental frequency. Consequently, the second overtone will be the same as the third harmonic. Certain instruments (for example percussive instruments) have overtones that are not integer multiples of the fundamental frequency, resulting in sounds with no clear sense of pitch. These overtones are called inharmonic overtones or partials.

Harmonics account for the colour of the tone; that is, the timbre. Different musical instruments have different amplitudes for the different harmonics, and no instrument can produce all of the harmonics (the clarinet, for instance, only has odd harmonics). Each instrument therefore has its own harmonic profile – almost like a fingerprint. The harmonic profile of the clarinet will therefore be distinctly different from that of the flute. In addition, differing designs (even if only slightly) in similar instruments

will also result in different harmonic profiles; so, for example, a Stradivarius violin will have a different “fingerprint” than a modern-day Yamaha violin.

The theory of Fourier series shows that sound waves can be decomposed into the sum of different sine waves, all with different amplitudes. Since different instruments have overtones with different amplitudes, the sum of these sine waves will result in a different waveform for each instrument. The following graphs are oscillograph traces of these waveforms for flute, clarinet and guitar, all playing the same pitch (trace lasting for only one hundredth of a second), and showing a clearly different pattern (graphs from Taylor, 2003).

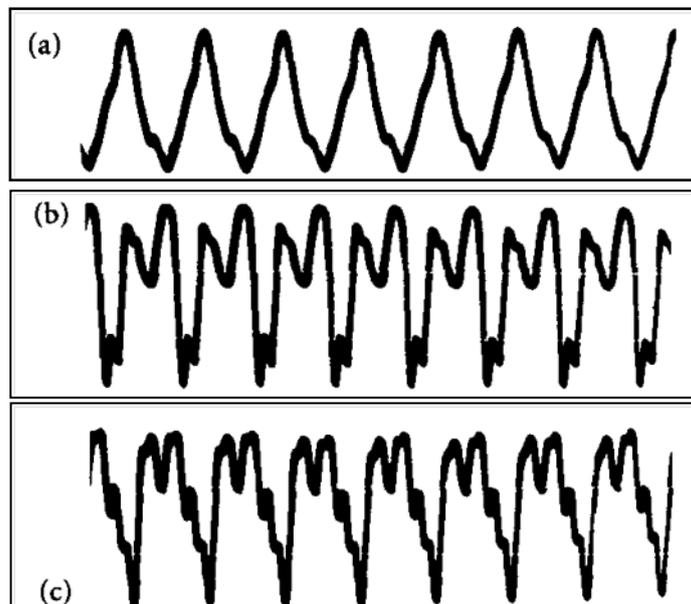


Figure 2.3: Waveforms of flute (a), clarinet (b) and guitar (c), all playing the same pitch.

2.5 Digital music

It is clear from the above that a vast amount of information is contained within a single sound wave. This information can be captured in the form of an analogue or digital recording.

In analogue music recordings (such as vinyl records or cassette tapes), variations in air pressure are converted into an electrical analogue signal and the variations of the

electrical signal are then converted to variations in a physical recording medium such as a vinyl record or cassette tape.

These days, the vast majority of music is recorded in a digital format such as compact disc (CD) (uncompressed data) or file formats such as .WAV (uncompressed) or MP3 (compressed). The simplest way of converting an analogue signal to a digital signal, is to sample the signal a large number of times a second, with a binary number representing the height of the waveform at each sampling point. CD's are based on a sampling rate of 44.1 kHz, translating to 44,100 samples per second of audio, equally spaced in time. At each sampling point, a 16 digit binary number represents the height of the waveform at that particular point (consequently, the dynamic range of a CD is referred to as 16 bits). MP3 files use lossy data compression which reduces the amount of data required to represent an audio recording, making it popular for file sharing over the Internet. An MP3 audio file created using a 128 kbit/s setting will result in a file the size of which is just $\frac{1}{11}$ th of that of an original CD quality file. Other popular formats for audio storage and compression are AAC (Advanced Audio Coding) and WMA (Windows Media Audio). The details of how these different file formats function and how they are obtained are not important for the purposes of this study.

Digital audio data therefore consists of sequences of amplitude values of the sound which are essentially unstructured and vast in number; for example, a 3-minute CD quality section of audio recorded in stereo and stored as uncompressed digital audio is represented by a sequence of almost 16 million binary numbers³. Data in such a format is not suitable for traditional data mining algorithms and we need to find a higher-level representation.

³ Calculated as 3 (minutes) x 60 (seconds) x 2 (stereo channels) x 44,100 (sampling rate) = 15,876,000

2.6 Audio feature extraction

2.6.1 Background

Audio feature extraction is the foundation of any type of music data mining, and can be defined as “the process of distilling huge amounts of raw audio data into much more compact representations that capture higher level information about the underlying musical content” (Tzanetakis, 2011). In other words, the goal is to compute a numerical representation of a segment of audio.

Extracting meaningful features from audio data is not a new area of research, and a lot of work has been done in areas such as speech processing and audio signal analysis. Many techniques used in speech signal processing have been successfully applied to music and there are a lot of useful synergies between the two fields. However, Müller *et al.* (2011) argue that a deep and thorough insight into the nature of music itself should always underlie signal processing (and thus feature extraction) in a musical audio context.

Since music signals are generally periodic and change over time, a representation that gives a separate notion of time and frequency is usually one of the first steps in audio feature extraction. Probably the most common audio representation used for audio feature extraction, is the short-time Fourier transform (STFT) (Müller *et al.*, 2011). This entails dividing the signal into small segments in time, and calculating the frequency content of each such segment. The STFT has its basis in the theory of Fourier series, which is the classic mathematical theory for describing musical tones. To understand the STFT, the general theory of Fourier series first needs to be reviewed. (The description below to a large extent follows Alm and Walker, 2002.)

2.6.2 Theory of Fourier series

Given a sound signal $f(t)$ with period Ω , its Fourier series defined on the interval $[0, \Omega]$ is:

$$f(t) = c_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{2\pi n t}{\Omega} + b_n \sin \frac{2\pi n t}{\Omega} \right) \quad (2.1)$$

with its Fourier coefficients defined by

$$c_0 = \frac{1}{\Omega} \int_0^{\Omega} f(t) dt$$

$$a_n = \frac{2}{\Omega} \int_0^{\Omega} f(t) \cos \frac{2\pi n t}{\Omega} dt, \quad n = 1, 2, 3, \dots$$

$$b_n = \frac{2}{\Omega} \int_0^{\Omega} f(t) \sin \frac{2\pi n t}{\Omega} dt, \quad n = 1, 2, 3, \dots$$

The constant c_0 represents a constant background air pressure level; each additional term in the Fourier series in (2.1) has a frequency of $\frac{n}{\Omega}$, so that we get a superposition of waves which are integer multiples of a fundamental frequency $\frac{1}{\Omega}$.

The Fourier series in (2.1) can be rewritten using complex exponentials:

$$f(t) = c_0 + \sum_{n=1}^{\infty} (c_n e^{i2\pi n t/\Omega} + c_{-n} e^{-i2\pi n t/\Omega}) \quad (2.2)$$

with the Fourier coefficients given by

$$c_n = \frac{1}{\Omega} \int_0^{\Omega} f(t) e^{-i2\pi n t/\Omega} dt, \quad n = 0, \pm 1, \pm 2, \dots \quad (2.3)$$

Parseval's equality (Alm and Walker, 2002), a well-known result in the theory of Fourier series, states that

$$\frac{1}{\Omega} \int_0^{\Omega} |f(t)|^2 dt = |c_0|^2 + \sum_{n=1}^{\infty} \{|c_n|^2 + |c_{-n}|^2\} \quad (2.4)$$

or, since $|c_n|^2 = |c_{-n}|^2$,

$$\frac{1}{\Omega} \int_0^{\Omega} |f(t)|^2 dt = |c_0|^2 + \sum_{n=1}^{\infty} 2|c_n|^2.$$

If we define the energy of a function $g(t)$ over $[0, \Omega]$ as

$$\int_0^{\Omega} |g(t)|^2 dt$$

then $\Omega |c_n|^2$ is the energy of the complex exponential $c_n e^{i2\pi nt/\Omega}$.

So by Parseval's equality (Equation 2.4) we can show that the energy of the sound signal f is equal to the sum of the energies of the complex exponentials in its Fourier series, and the Fourier series spectrum $\{2|c_n|^2\}_{n \geq 1}$ therefore completely captures the energies in the frequencies of the audio signal. (The term $|c_0|^2$ is the energy of the constant background and is inaudible, so can be ignored.)

To illustrate this graphically, Figure 2.4 shows the oscillograph trace of a piano tone (with a frequency of 329.628Hz) together with the computer calculated Fourier spectrum of this tone (graphs from Alm and Walker, 2002).

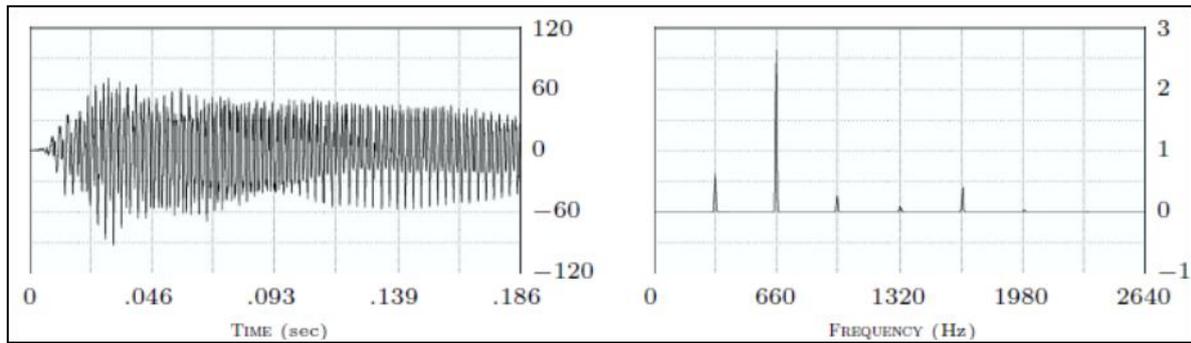


Figure 2.4: Piano tone (left) with its Fourier spectrum (right)

The spectrum clearly shows the fundamental frequency of $\cong 330\text{Hz}$, with harmonics sounding at integer multiples of the fundamental. The different amplitudes for the different harmonics are part of what constitutes the timbre of the sound.

2.6.3 Discrete Fourier transforms

To calculate Fourier spectra, approximations to the Fourier coefficients are generally used. These approximations are called discrete Fourier transforms (DFT).

For N a large positive integer, let

$$t_k = \frac{k\Omega}{N} \quad \text{for } k = 0, 1, 2, \dots, N-1$$

and

$$\Delta t = \frac{\Omega}{N}$$

Then the n^{th} Fourier coefficient c_n (as defined in Equation 2.3) is approximated by

$$c_n \approx \frac{1}{\Omega} \sum_{k=0}^{N-1} f(t_k) e^{-i2\pi n t_k / \Omega} \Delta t$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} f(t_k) e^{-i2\pi nk/N}$$

which is the DFT of the finite sequence of numbers $\{f(t_k)\}$.

When calculating Fourier spectra, the DFT approximations for the Fourier coefficients are often used. It is possible to calculate the DFT of an audio clip in its entirety, but although this would give an indication of how the energy of the signal is distributed among different frequencies, it would give no information about when frequencies start and stop. For example, Figure 2.5 shows the graph of a recording of a piano playing four successive tones, together with its calculated Fourier spectrum. Unlike Figure 2.4, where there was one single tone, in this instance it is fairly difficult to determine fundamental frequencies and harmonics, since there is a mixture of spectra from individual tones (graphs from Alm and Walker, 2002).

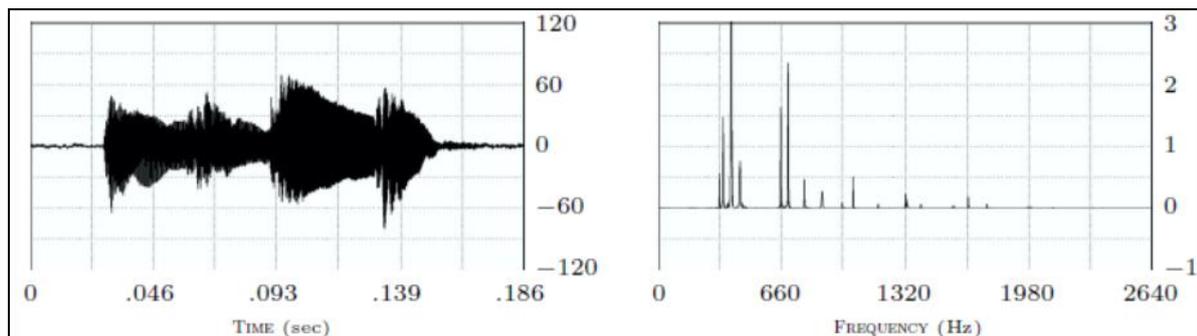


Figure 2.5: Piano passage of 4 tones (left) with its Fourier spectrum (right)

To address this shortcoming, windowing is applied to the sound signal $f(t)$ prior to calculating the DFT, and this process – which is referred to as the short-time Fourier transform (STFT) – produces Fourier coefficients which are localised in time.

2.6.4 The short-time Fourier transform

To calculate the STFT, the sound signal $f(t)$ is multiplied by a sequence of windows $\{w(t - \tau_m)\}$ with $m = 1, 2, \dots, M$, where M is the number of windows. In other words, instead of calculating the DFT of the sound signal $f(t)$, the DFT of the sequence

$$\{f(t)w(t - \tau_m)\}_{m=1,\dots,M}$$

is calculated instead. The STFT is therefore a DFT which is adapted to deal with local sections of a signal as it changes over time, and for this reason the STFT is also sometimes referred to as the windowed Fourier transform.

The choice of window is important, since windowing “smears” the spectrum so that each component in the Fourier series includes some energy from nearby components. Some popular windows are the rectangular, Hann, Hamming, Gaussian and Blackman windows, and windows are usually allowed to overlap. Window size is also important, since larger windows give a higher frequency resolution, but at a less accurate time resolution. This trade-off is very important in any type of time-frequency analysis.

Finding the STFT can be computationally expensive, but it can be computed at high speed by using the Fast Fourier transform (FFT), details of which can be found in Oppenheim (1970).

2.6.5 Spectrograms

Whereas the output of the DFT is called a spectrum, when the STFT is visualised in terms of its magnitude, it is referred to as the magnitude spectrum, or spectrogram.

Formally, a spectrogram is defined as the squared magnitude of the STFT. So if the STFT is given by $F(t, w)$, then the spectrogram $S(t, w)$ is calculated as

$$S(t, w) = |F(t, w)|^2$$

The resulting representation contains information about how the energy of a signal is distributed in both the time and frequency domains. The identity of a sound is mostly affected by the magnitude spectrum, and therefore in the majority of cases of audio feature extraction for analysing music, only the magnitude spectrum is considered (Tzanetakis, 2011).

In Figure 2.6, spectrograms for the piano and flute respectively are shown. Colours correspond to the magnitude, with red strong and blue weak. It is clear that the piano has more complex harmonics than the flute (graphs from Niwa *et al.*, 2006).

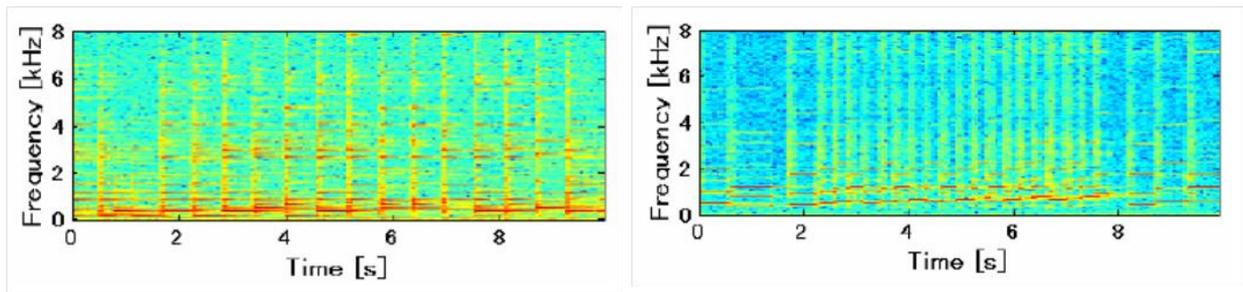


Figure 2.6: Spectrograms of piano (left) and flute (right)

However, a spectrogram will still contain some information which will not be important for analysis purposes, and the dimensionality will be very large, making it unsuitable for use with traditional data mining algorithms. A set of features is therefore usually calculated from the magnitude spectrum, giving some indication of the spectral shape, and these features are then used in all subsequent analyses. Some commonly used features will be defined and described in Section 2.7.

2.6.6 Other time-frequency representations

While the STFT is the most commonly used time-frequency representation, there are also many other techniques available to represent sound signals in this way, many of which are also based on the Fourier transform. Some of these techniques, such as wavelet analysis, the Mel filterbank and auditory models, are briefly described in Tzanetakis (2011).

2.6.7 Extracting features

Many researchers implement their own feature extraction algorithms as a preliminary step of their research. This allows customisation of features for the research question at hand. However, many audio features have become fairly standard and there are software programs and / or toolboxes available to calculate them. The table below (expanded from Tzanetakis, 2011) shows some of the freely available software for audio feature extraction:

Table 2.1: Software resources for feature extraction

Name	URL	Programming language / environment
Auditory Toolbox	tinyurl.com/3yomxwl	MATLAB
CLAM	clam-project.org	C++
D. Ellis Code	tinyurl.com/6cvtdz	MATLAB
HTK	htk.eng.cam.ac.uk	C++
jAudio	tinyurl.com/3ah8ox9	Java
Marsyas	marsyas.info	C++ / Python
MA Toolbox	www.pampalk.at/ma	MATLAB
MIR Toolbox	tinyurl.com/365oojm	MATLAB
Sphinx	cmusphinx.sourceforge.net	C++
VAMP Plugins	www.vamp-plugins.org	C++
Maaate	maaate.sourceforge.net	C++
FEAPI	feapi.sourceforge.net	C++
YAAFE	yaafe.sourceforge.net	C++ / Python

2.6.8 The MPEG7 standard

Based on research undertaken in the music information retrieval area, the ISO Motion Picture Experts Group (MPEG) proposed the MPEG-7 standard (Kim *et al*, 2005), which defines standardised descriptions for audiovisual data. Part of the MPEG-7 standard consists of a set of low-level audio descriptors in both the temporal and spectral domains. These descriptors can be extracted from audio automatically, and depict the variation of audio properties over time or frequency. MPEG-7 descriptors

are often used to analyse the similarity between different audio signals (Kim *et al.*, 2005). A major advantage of MPEG-7 features in terms of performance, is that the features can be computed directly from compressed audio data.

2.7 Commonly used features

In the following sections, some commonly used features will be defined and described. Not all features have formal, standardised definitions, and some could therefore be defined in more than one way. Wherever possible, the most generally accepted definition has been used; in instances where a formal, standardised definition exists (such as in the case of the MPEG-7 standard) this has been explicitly stated. These features will arise in our discussion of a practical dataset in Chapter 8.

2.7.1 Temporal centroid

The temporal centroid is the time instant where the energy of the sound is focused, and is calculated as the energy weighted mean of the sound duration (Jiang *et al.*, 2009b). Temporal centroid is formally defined in the MPEG-7 standard (Kim *et al.*, 2005).

2.7.2 Spectral centroid

Spectral centroid can be calculated in a number of different ways. It is generally defined as the centre of gravity of the magnitude spectrum of the STFT (Tzanetakis, 2002) and it gives a measure of the shape of the spectrum, with higher values corresponding to “brighter” sounds with more high frequencies.

The MPEG-7 standard includes three measures of spectral centroid: Audio Spectrum Centroid (referred to as Log Spectral Centroid by Jiang *et al.*, 2009b), Harmonic Spectral Centroid (referred to as Spectral Centroid by Jiang *et al.*, 2009b) as well as a basic Spectral Centroid measure not related to the harmonic structure of the signal.

The Audio Spectrum Centroid gives the centre of gravity of a log-frequency power spectrum, whereas the Harmonic Spectral Centroid is defined as the average of the amplitude-weighted mean of the harmonic peaks of the spectrum (Kim *et al.*, 2005).

Mathematically:

$$\text{Audio Spectrum Centroid} = \frac{\sum_{k'=0}^{(N_{FT}/2)-K_{low}} \log_2 \left(\frac{f'(k')}{1000} \right) P'(k')}{\sum_{k'=0}^{(N_{FT}/2)-K_{low}} P'(k')}$$

where $P'(k')$ and $f'(k')$ are the power coefficients and frequencies respectively of a modified power spectrum, obtained by summing all power coefficients below 62.5 Hz in the original power spectrum, and representing them by a single coefficient. K_{low} gives the index on the discrete frequency bin scale below which every power coefficient is summed; N_{FT} is the size of the DFT.

Furthermore:

$$\text{Harmonic Spectral Centroid} = \frac{1}{L} \sum_{l=0}^{L-1} LHSC_l$$

where L is the number of frames in the segment, and $LHSC_l$ represents the local harmonic spectral centroid at the l^{th} frame:

$$LHSC_l = \frac{\sum_{h=1}^{N_H} f_{h,l} A_{h,l}}{\sum_{h=1}^{N_H} A_{h,l}}$$

where $f_{h,l}$ is the frequency of the h^{th} harmonic peak estimated at the l^{th} frame, N_H is the number of harmonics taken into account, and $A_{h,l}$ is the corresponding amplitude.

2.7.3 Spectral spread

The MPEG-7 Audio Spectrum Spread (also called instantaneous bandwidth) is referred to by Jiang *et al.* (2009b) as Log Spectral Spread, while they refer to the MPEG-7 Harmonic Spectral Spread as Spectral Spread. Spectral Spread is an economical way of describing the shape of the power spectrum:

$$\text{Harmonic Spectral Spread} = \frac{1}{L} \sum_{l=0}^{L-1} LHSS_l$$

where L is the number of frames in the segment, and $LHSS_l$ represents the local harmonic spectral spread at the l^{th} frame:

$$LHSS_l = \frac{1}{LHSC_l} \sqrt{\frac{\sum_{h=1}^{N_H} [(f_{h,l} - LHSC_l)^2 A_{h,l}^2]}{\sum_{h=1}^{N_H} A_{h,l}^2}}$$

where $LHSC_l$, $f_{h,l}$, $A_{h,l}$ and N_H are as defined in Section 2.7.2 above. Furthermore,

$$\text{Audio Spectrum Spread} = \sqrt{\frac{\sum_{k'=0}^{(N_{FT}/2)-K_{low}} [\log_2 \left(\frac{f'(k')}{1000} \right) - ASC]^2 P'(k')}{\sum_{k'=0}^{(N_{FT}/2)-K_{low}} P'(k')}}^2$$

where ASC is the Audio Spectrum Centroid as defined in Section 2.7.2 and $P'(k')$, $f'(k')$, K_{low} and N_{FT} are all as defined in Section 2.7.2.

2.7.4 Mel-Frequency Cepstral Coefficients (MFCCs)

Mel-Frequency Cepstral Coefficients, or MFCCs, are perceptually motivated features based on the STFT. It describes the spectrum according to the human perception system in mel scale. These are commonly used features in the field of speech and speaker recognition, but are also widely used in music information retrieval.

The mel is a unit of pitch, and the mel scale is a scale of pitches perceived by listeners to be equal in distance from each other. (The human auditory system does not perceive pitch in a linear manner; below 1 kHz the mapping from Hz to mel scale is approximately linear, but logarithmic above.) A frequency f (in Hz) is converted to mel scale using the following formula:

$$\text{Pitch (mel scale)} = 2595 \log_{10} \left(1 + \frac{f}{700} \right)$$

The frequency (Hz) mel scale mapping is portrayed in Figure 2.7 (graph from Logan, 2000):

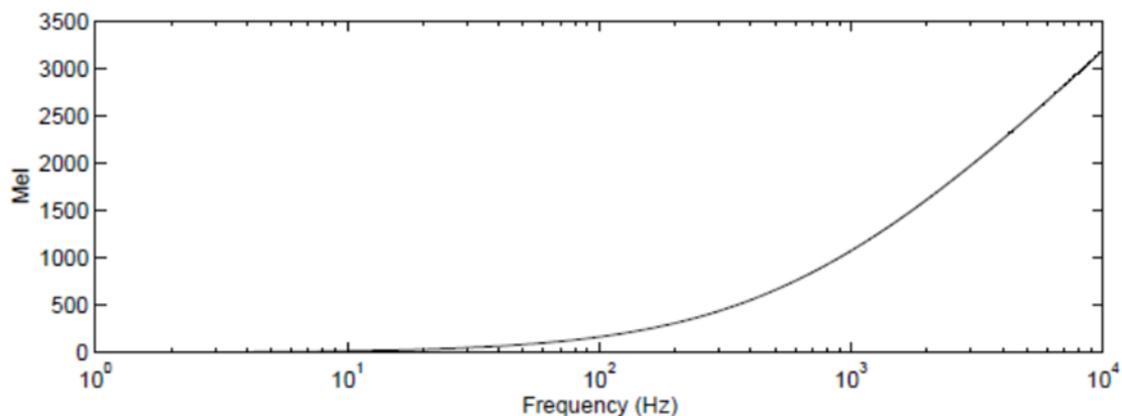


Figure 2.7: *The mel scale*

The MFCC representation is defined as the cepstrum of a windowed short-time signal. MFCCs are typically extracted as follows (Logan, 2000):

1. Convert a signal to frames (usually applying a windowing function, typically a Hamming window).
2. Calculate the discrete Fourier transform of each frame.

3. Calculate the logarithm of the amplitude spectrum (because perceived loudness of a signal has been shown to be approximately logarithmic) (Logan, 2000).
4. Group and smooth the spectral components according to the mel-frequency scale.
5. Apply a Discrete Cosine Transform to the resulting features to decorrelate them.

Typically, the first 13 MFCCs are retained in speech and music applications, although the number may vary between studies.

2.7.5 Energy

The spectral energy is defined as the sum over all values of the power spectrum:

$$Total\ Energy = \sum_{i=1}^N X_i^2$$

where X_i is the value of the magnitude of the Fourier transform at bin i and N is the total number of bins in the Fourier transform.

The average energy of the spectrum is sometimes used instead of the total energy (for example, Jiang *et al.* (2009b)).

2.7.6 Zero Crossing

The Zero Crossing Rate is a fairly common feature used for describing audio signals, musical or otherwise. It gives an indication of the noisiness of the signal, and is also a key feature in classifying percussive sounds. It is defined as the number of times the sampled signal changes sign in a given frame:

$$Zero\ Crossing\ Rate_t = \frac{1}{2} \sum_{n=1}^N |sign(x[n]) - sign(x[n-1])|$$

where

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

and $x[n]$ is the n^{th} sample signal at time t with N the frame size.

For clean signals (i.e. signals without noise), the Zero Crossing Rate is highly correlated with the spectral centroid (Tzanetakis, 2002).

2.7.7 Rolloff

Rolloff is another measure of spectral shape, and it shows how much of the energy of a signal is concentrated in the lower frequencies. It is often used to distinguish between voiced and unvoiced speech, but has also been extensively used in music information retrieval.

Rolloff is defined as the frequency R_t below which $C\%$ of the accumulated magnitudes of the spectrum lies. C is an empirical coefficient, but in most cases is taken to be 85. So, rolloff is computed from

$$\sum_{n=1}^{R_t} M_t[n] = \frac{C}{100} \sum_{n=1}^N M_t[n]$$

where $M_t[n]$ is the magnitude of the Fourier transform at time t and frequency bin⁴ n (Tzanetakis, 2002).

⁴ In this context, frequency bins are considered to be the discrete intervals in the Fourier transform.

2.7.8 Flux

Flux is defined as the difference between the magnitude of the amplitude spectrum points in a given frame and its successive frame, and it provides a measure of the amount of spectral change (Tzanetakis, 2002).

For $N_t[n]$ and $N_{t-1}[n]$ the normalised magnitudes of the Fourier transform at frames t and $t - 1$ respectively, Flux is defined by

$$Flux_t = \sum_{n=1}^N (N_t[n] - N_{t-1}[n])^2$$

2.7.9 Flatness coefficients

Flatness coefficients are defined in the MPEG-7 standard by the Audio Spectrum Flatness parameter, and reflect the flatness properties of the spectrum. For each frame a series of values is calculated, each value representing a measure of the deviation of the spectrum from a flat shape inside a predefined frequency band. It therefore gives a measure of how similar an audio signal is to white noise (flat spectral shape), or how harmonic it is.

2.7.10 Projection coefficients

Also derived from the MPEG-7 standard, these values represent a low-dimensional projection of a high-dimensional spectral space. They are derived from the singular value decomposition of the spectrum.

2.7.11 Harmonic Peaks

These are a sequence of spectrum coefficients of the local peaks of harmonics for a given frame.

2.7.12 Log Attack Time

Temporally, a sound signal can typically consist of four phases: attack, decay, sustain and release.

Attack is the time it takes for the sound to reach its initial maximum amplitude, *decay* is the time taken to reach the second level of amplitude (the sustain level), *sustain* is the amplitude level at which the sound is sustained after the decay phase (usually – but not necessarily – lower than the attack amplitude) and *release* is the time it takes for the amplitude to fall back to zero. Graphically these phases can be portrayed as follows (graph from Kim *et al.*, 2005):

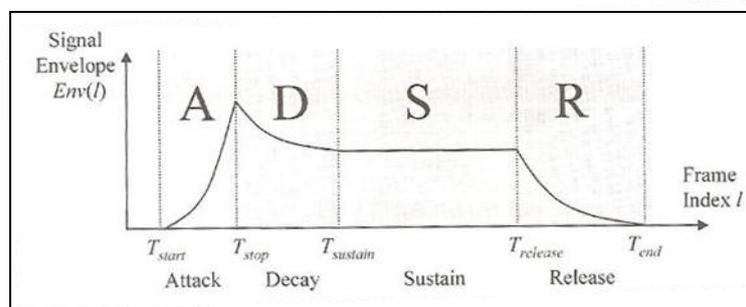


Figure 2.8: The Attack-Decay-Sustain-Release (ADSR) envelope of a sound

A sound does not have to have all of these phases; for instance, the organ does not have a decay phase. The ADSR envelope of a sound can be very useful in distinguishing between musical instruments since different instruments have different envelope characteristics. The piano, for instance, is characterised by a very sharp attack phase, whereas a wind instrument such as the flute will have a more gradual attack. Experiments performed in the 1950s by composer Pierre Schaeffer also showed that the attack phase of a sound is crucial for enabling humans to differentiate between different instruments (Levitin, 2006). Some form of feature representation

of this ADSR envelope could therefore be very useful in the instrument recognition problem.

The Log Attack Time (MPEG-7 standard) is defined as the decimal base logarithm of the duration from the time when the signal starts to the time when it reaches either its maximum value, or its sustained part, i.e.

$$\text{Log Attack Time} = \log_{10}(T_{\text{stop}} - T_{\text{start}})$$

There can be some difficulty in determining where the attack portion of a sound ends, and where the steady state begins. A suggestion for a simple way of estimating T_{start} and T_{stop} is given by Kim *et al.* (2005) as:

- Estimate T_{start} as the time at which the sound signal envelope exceeds 2% of its maximum value
- Estimate T_{stop} as the time at which the sound signal envelope reaches its maximum value

2.8 Sub-fields of music information retrieval

Music information retrieval can be text-based or content-based. Text-based retrieval relies on manually generated annotations such as composer name, opus number and lyrics. In this instance, retrieval can be handled in conventional ways as it would be for any other non-musical problem. Content-based retrieval, however, uses the raw musical data as input. Text-based retrieval has the advantage of being able to rely on simple data and of being able to utilise standard existing retrieval techniques. However, the reliance on manual annotations is a significant drawback, especially in the light of the growing corpus of digitally available music. Content-based retrieval means that classification and other tasks can be automated, something which is especially beneficial not only in terms of handling large volumes of data, but also in the way that new additions to databases (i.e. new pieces of music) can immediately be annotated or classified. The major complicating factor for content-based retrieval however, is the complexity of musical data. A novel way of combining text-based

retrieval with simple content-based retrieval has been proposed by Levy and Sandler (2009). For the remainder of this chapter, we will focus on content-based music information retrieval only.

Although MIR is a vibrant research field, there are still many unsolved problems, mostly because music data is so complex.

Some of the main fields of research in MIR are:

- Instrument recognition (will be discussed in detail in Chapter 3)
- Classification of music by mood / emotion (Section 2.9.1)
- Classification of music by genre (Section 2.9.2)
- Automatic music transcription (Section 2.10)
- Query-by-example (Section 2.11)
- Score following, audio alignment and music synchronisation (Section 2.12)
- Music structure analysis (Section 2.13)
- Performance analysis (Section 2.14)

In the rest of this chapter, some of these fields of research will be discussed in more detail.

Descriptions of MIR techniques in this chapter largely follow Müller (2011), Fu *et al.* (2011), Yang and Chen (2012) and Benetos *et al.* (2012), since they provide good overviews of the state-of-the-art of some of the main MIR tasks. Müller (2011) does not cover any music classification tasks, but focuses in-depth on new developments in the fields of query-by-example (music retrieval), music synchronisation, structure analysis and performance analysis. He discusses methods used in the most recent research in each of these fields, and also points out challenges faced by researchers in each field. Fu *et al.* (2011) focus on music classification and provide a review of state-of-the-art techniques for music classification. They discuss genre classification, mood classification, artist identification, instrument recognition and pay particular attention to the different features and classifiers used in the different fields. They also discuss open issues that require further investigation. Benetos *et al.* (2012) provide an overview of work done in the field of automatic music transcription (AMT) and discuss challenges that present themselves in the field. They then go on to suggest

directions for future research which may help to overcome some or all of these challenges. Yang and Chen (2012) focus on the classification of music by emotion, and provide a comprehensive review of the methods that have been proposed in this field. They also discuss the challenges inherent in music emotion classification and suggest some directions for further research. These papers provide a good starting point for further reading about music information retrieval; they also provide comprehensive references.

2.9 Music classification

Music classification encompasses a wide range of applications, but four of the most prevalent are music classification by mood / emotion, music genre classification, performer identification and instrument recognition. Since the last of these is one of the main topics of this dissertation, it will be discussed in more detail in Chapter 3. Performer identification is closely related to the field of performance analysis, which will be discussed in detail in Section 2.14. In this section we will therefore focus on classification by mood and genre.

A fairly comprehensive overview of classification in music was done by Weihs *et al.* (2007) but given how young and dynamic the field of music classification is, this can be considered a fairly old review and it does not discuss any of the latest state-of-the-art developments. We will therefore largely follow a newer survey paper by Fu *et al.* (2011).

2.9.1 Classification of music by emotion

Emotion is undeniably part of music; in a 1959 work entitled *The Language of Music* the author and musician Deryck Cooke goes as far as claiming that music is a language of emotional expression (Davies, 1994). A few studies have also confirmed that emotion plays a big role in how people search for music; for instance, in an early user survey in the MIR field, Lee and Downie (2004) found that 28% of respondents said they would search or browse music according to their mood or emotional state.

A study by Lamere (2008) also shows that mood is third only to genre and locale in terms of the type of tags assigned on the Last.fm⁵ website. The important role of mood / emotion⁶ in music retrieval is therefore quite clear. Furthermore, some studies have also suggested the possibility of using mood or emotion as a form of music recommendation (Yang and Chen, 2012); for instance, Dornbush *et al.* (2005) propose a mobile digital music player which can detect a user's emotions and play suitable and relevant music to match the user's mood.

There is a big crossover between the technical study of emotion classification in music and psychology. Research in music classification by emotion is enhanced by considering relevant psychological studies; however, the classification task is also complicated by the complexity of human emotions. Perhaps the greatest difficulty lies in the fact that emotions are subjective – there is no “ground truth” in terms of which emotion could or should be associated with a particular piece of music. Furthermore, emotions are not consistent, even for a single listener. Evaluating classification accuracy of any emotion classification algorithm is therefore greatly hampered by the subjective nature of human emotions. A discussion of the psychological aspects of music classification by emotion is beyond the scope of this thesis, but a good source for further reading on the emotional aspects of music is Gabrielsson and Juslin (2003).

In a typical approach to a music emotion classification problem we have a number of emotions (classes) and the aim is to train a classifier to classify a new piece of music into one (or more) of these classes. A number of machine learning techniques such as support vector machines (SVM), decision trees, neural networks and Gaussian mixture models (GMM) have been used in the context of music emotion classification, but the support vector machine seems to give superior results in many cases (Yang and Chen, 2012). Although music emotion classification is often approached as a single-label classification problem [see for example Laurier and Herrera, 2007 (SVM); Lu *et al.*, 2006 (GMM); Peeters, 2008 (GMM); Yang *et al.*, 2006 (fuzzy classifiers); Yang *et al.*, 2008 (multi-modal approach)], quite a number of researchers have taken the view that it should rather be approached as a multi-label

⁵ <http://www.last.fm> (accessed 5 November 2013)

⁶ In music information retrieval, the terms ‘mood’ and ‘emotion’ are often used interchangeably. However, in psychology there is a clear distinction made between the two: emotion generally refers to a short experience in response to something, while mood is a longer-term experience not necessarily as a response to something (Sloboda and Juslin, 2001).

problem, since a piece of music might evoke more than one emotion (whether simultaneously or at different places in the piece of music) (for example Li and Ogihara, 2003; Wiczorkowska *et al.*, 2006; Trohidis *et al.*, 2011).

In the context of music emotion classification, the above approach (that is, classifying a piece of music into one or more categories of emotions) is called the *categorical* approach; another way to tackle these types of problems is the *dimensional* approach, in which emotions are plotted on a (typically two-dimensional) system of axes. Details of this approach can be found in Yang and Chen (2012) and Trohidis *et al.* (2011).

One aspect that is especially important in music emotion classification, is that of feature extraction. While it is always important to extract and select suitable features for a model, in music emotion classification it is doubly so, because of the inherent relationship between some aspects of music and emotion. For instance, the tempo of a piece of music – that is, whether it is a fast or a slow piece of music – will have a profound effect on the emotions induced by the music. Gabriellson and Lindström (2001) give a good overview of the influence of different factors in musical structure (such as tempo variations, loudness and articulation) on perceived emotional expression.

A major challenge in music emotion classification is the lack of consensus about which emotion model should be used when classifying music. There exists a multitude of emotion models; some of those regularly encountered are those of Hevner (Hevner, 1935), Thayer (Thayer, 1989), Farnsworth (Farnsworth, 1954), Russell (Russell, 1980) and Tellegen-Watson-Clark (Tellegen *et al.*, 1999), but many others exist and are also used in the MIR field. These models also mostly have different numbers of emotion categories or dimensions, so there is also not consensus on the number of classes that should be used. In addition (as briefly implied before), there is not even consensus about whether emotions should in fact be viewed as categories (classes) or as dimensions (points on a continuum). The comparative study by Yang and Chen (2012) found that the number of emotion classes ranged from 3 to 18, or 2- to 3-dimensional emotion spaces across the 26 studies they considered.

Another complicating factor referred to earlier is the subjective nature of emotions: there is no “correct” answer as to which emotion is associated with a particular piece of music. In order to label training data in this context, music is annotated manually in a number of possible ways, such as labelling by experts or untrained subjects, or even web-based annotation from music websites such as Last.fm. This is a labour-intensive process and the result obtained is still not the “ground truth”.

A consequence of the above-mentioned complications is that there is no benchmark dataset that is widely used for music emotion classification, as different researchers tend to collect and use their own data based on the emotion model of their choice. A major hurdle in the field of music emotion classification is therefore that it is extremely difficult to compare results from different studies.

An example of a commercial application of music classification by emotion, is Moodagent⁷ – available as a smartphone app, stand-alone desktop version or through music streaming services such as Spotify. Moodagent creates a unique profile for individual music tracks through a combination of digital processing, audio analysis, music science and artificial intelligence; users can then create playlists based on mood.

2.9.2 Classification of music by genre

Musical genre classification is the most widely studied area in MIR, according to Fu *et al.* (2011). This may be at least partly due to the fact that genre is a natural way to search for music – a study by Lamere (2008) has shown that genre is the most commonly used tag category on the social music website Last.fm (68% of tags were genre tags).

The aim of musical genre classification is to allocate a genre category to a piece of music. As with other music classification problems, this is accomplished by first extracting the relevant features from the raw audio data and then training a classifier on the derived dataset.

⁷ <http://www.moodagent.com> (website accessed 10 February 2013)

One of the seminal papers in the field of musical genre classification was that of Tzanetakis and Cook (2002). According to them, genre characteristics are typically related to the instrumentation, rhythm and harmonic content of the music so this should be kept in mind when deciding which features to extract. In terms of features used, MFCCs appear to be the most important features used in genre classification (Fu *et al.*, 2011).

Fu *et al.* (2011) give a very good summary of different studies that have been undertaken in the field of musical genre classification, and compare features and classifiers used and also report on accuracy rates. Some of the classifiers compared in their paper are k-nearest neighbours, support vector machines, Gaussian mixture models and boosted trees. Support vector machines seem to fare very well in the context of musical genre classification: "... [the SVM] has been used predominantly in music classification and consistently outperformed other standard classifiers like k-NN and GMM" (Fu *et al.*, 2011). Multi-label learning methods are also increasingly being employed for musical genre classification (Lukashevich *et al.*, 2009; Wang *et al.*, 2009), and Sanden and Zhang (2011) propose an interesting ensemble technique for approaching the problem. Novel techniques that have also been used in this context are locality preserving non-negative tensor factorisation (Panagakis *et al.*, 2009) and convolutional deep belief networks (Lee *et al.*, 2009).

An advantage in the field of musical genre classification (compared to, for instance, music emotion classification) is the availability of public benchmark datasets. One of the most widely used datasets is one provided by Tzanetakis and Cook in their 2002 study, but others are also available, for example the Dortmund dataset (Homburg *et al.*, 2005) and the CAL-500 dataset (Turnbull *et al.*, 2008).

As is often the case in music classification, a complication in musical genre classification is the subjective nature of the classes. Genre labels are created by humans, and there are no strict definitions and boundaries (Tzanetakis and Cook, 2002). Furthermore, there is no general consensus as to which genre taxonomy is the correct one to use. A study by Pachet and Cazaly (2000) compared the genre taxonomies of three Internet music retailers and found that there was very little agreement between the taxonomies. New genres are also regularly being added to the

corpus of music, and the definitions of existing ones change over time (McKay and Fujinaga, 2006). As Lamere (2008) states: “An ongoing problem for music librarians and editors is how to represent the music genre taxonomy”. Some researchers have therefore gone as far as suggesting that the genre classification problem be abandoned in favour of more general research into music similarity (McKay and Fujinaga, 2006). Despite these misgivings, research into musical genre classification still seems to go from strength to strength. While the papers by Tzanetakis and Cook (2002) and Fu *et al.* (2011) provide good starting points for the interested reader, a comprehensive source of further references is Sturm (2012), where almost 500 references are provided.

Although many commercial music recommendation systems such as Spotify and Pandora can recommend music by genre, most of these are based on text-based retrieval methods and therefore rely on manually allocated textual tags rather than content-based genre classifications.

2.10 Automatic music transcription

Automatic music transcription (AMT) is one of the main non-classification fields of study under the broader umbrella of MIR. The broad aim of AMT is to produce a musical score, or some form of musical notation, from a musical audio recording. Benetos *et al.* (2012) published a short overview of the current state of research in the AMT field in the ISMIR2012 proceedings. The importance of AMT is mirrored by the opening sentence of their publication: “Automatic music transcription is considered by many to be the Holy Grail in the field of music signal analysis.”

In written symbolic form (as found on musical scores), a musical note consists of pitch, onset and duration. In order to transcribe an audio note event, these three components need to be estimated. The estimation of pitch is related to what is probably the most important subtask of AMT, namely multiple-F0 estimation. Note onset and duration also need to be estimated and additional factors such as instrumentation need to be considered as well.

Multiple-F0 estimation is a complex task, the aim of which is to determine the fundamental frequencies of every note (many of which will be played concurrently) in a polyphonic piece of music. In the monophonic case, pitch estimation is significantly simpler and the problem is generally considered solved; however, in the polyphonic case, no system yet exists which is able to automatically transcribe the pitch of polyphonic music without any restrictions regarding instrument type, number of instruments, et cetera (Benetos *et al.*, 2012). An idea of the nature of the complexity of multiple-F0 estimation can be formed by considering the following two graphs; the first (Figure 2.9) shows the spectrum of a single instrument sound, whilst the second (Figure 2.10) shows the spectrum for a mixture of four (harmonic) sounds (graphs from Klapuri, 2004):

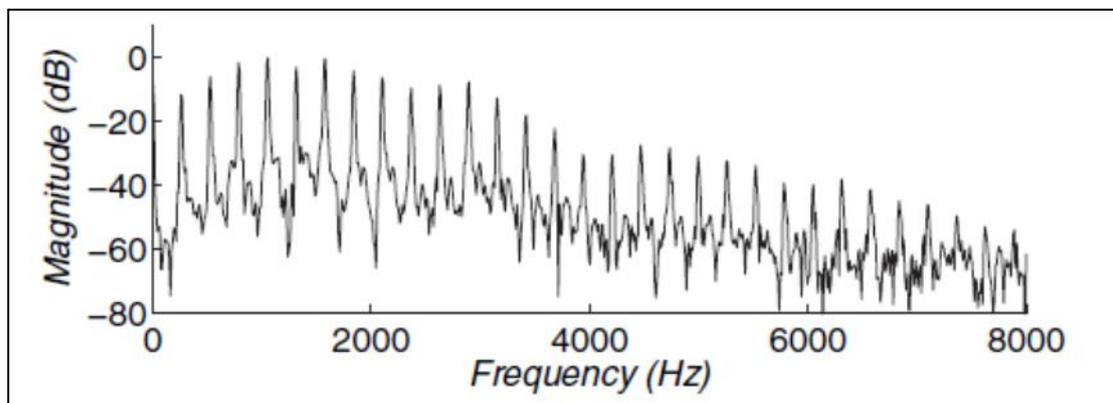


Figure 2.9: Spectrum of a single harmonic sound

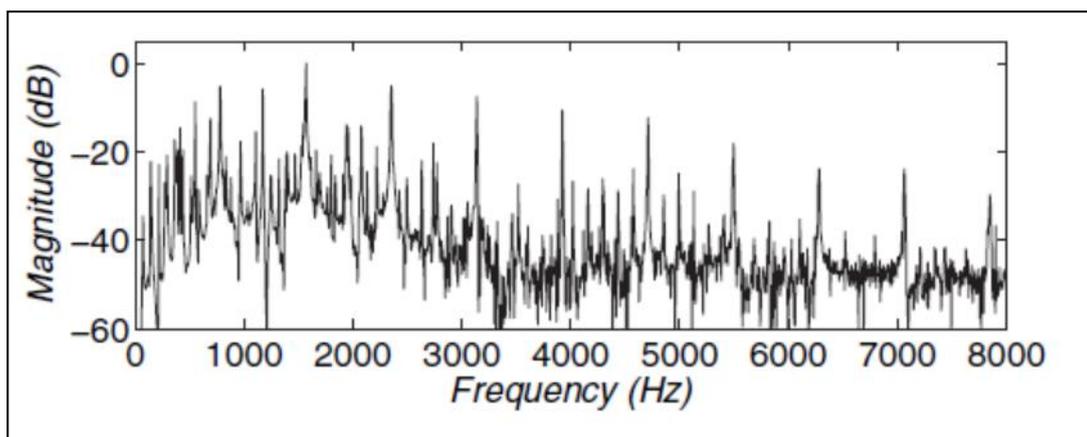


Figure 2.10: Spectrum of a mixture of four harmonic sounds

Many different approaches have been taken in an attempt to solve the problem of multiple-F0 estimation. Many of these approaches are fairly heuristic or consist of a

combination of several different techniques, and a detailed discussion of these fall outside of the scope of this study. Briefly however, some of the statistical techniques that have been utilised in this context and are touched upon in Benetos *et al.* (2012) are maximum likelihood estimation, non-negative matrix factorisation and probabilistic latent component analysis (PLCA). For the interested reader, the book by Christensen and Jakobsson (2009) gives a very detailed exposition of the multi-pitch estimation problem; good overviews are given by Christensen *et al.* (2008) and Klapuri (2004) (although the Klapuri paper is older and does not contain up-to-date references to the current state of the art in the field).

Onset detection is discussed in Müller *et al.* (2011) and in Bello *et al.* (2005); the aim here is to determine the physical starting times of individual notes within a music recording. The basic idea is to detect sudden changes in the audio signal, since such sudden changes are typically caused by the onset of a new note. These changes can be detected by way of a novelty curve, and the peaks in this curve indicate the most likely note onset positions. Different methods for computing such novelty curves are discussed in Bello *et al.* (2005). In non-percussive music such as Western classical music it is much harder to detect note onsets than in percussive music (which typically has strong beats), since note onsets are softer and often also masked by the presence of multiple instruments.

While multiple-F0 estimation and note onset detection without a doubt form the bulk of the automatic music transcription problem, these are not the only aspects under consideration. Even if the problems of multiple-F0 estimation and onset detection were solved (which is undoubtedly not the case), it would still not yield a complete transcription system. In order to provide output equivalent to that of a musical score or sheet music, aspects such as metre induction and rhythm parsing, key finding, dynamics and expression, fingering, articulation and typesetting also need to be considered (Benetos *et al.*, 2012). Many of these aspects have been considered in isolation, but to date no complete AMT system exists.

Another challenge faced in the field is the limitation of available training and testing data. The process of digitising and time-aligning musical scores to recordings is hugely time consuming and needs to be done by a human. Therefore, the datasets

currently used for evaluation of the majority of AMT models consist of only 12 tracks from the RWC database⁸. The data is therefore not representative and there is a real danger of overfitting (Benetos *et al.*, 2012).

An interesting remark by Benetos *et al.* (2012) relates to the fact that the majority of current approaches to AMT attempt to be fairly general, meaning that they are not restricted to specific instrument types or specific musical genres. They contend that this is surprising if one considers the fact that in automatic speech recognition – a much more mature field than musical signal processing – speech recognition systems are almost always language- and/or domain-specific. By creating more specific rather than general music transcription systems, prior knowledge about aspects such as instrument design or expert knowledge about a genre can be incorporated. This means that such a music transcription system will rely on a two-phased approach: first classifying the genre and / or instrumentation of a piece, followed by an appropriate transcription model.

2.11 Query-by-example

Personal music collections are rapidly becoming larger, as the available corpus of digital music keeps expanding. It is therefore becoming increasingly difficult to search through large music collections. Locating new music which might be of interest to a user is the domain of music recommendation (and relies on the concept of music similarity); searching for a specific fragment or piece of music is often referred to as “query by example”. This means that a user submits an audio fragment as input query; this input query can vary from a digital excerpt of a piece of music, to whistling, singing or humming a tune (query-by-singing/humming, or QbSH). The task is then to automatically retrieve all musical documents from the given database that “match” the query fragment. The word “match” is used in quotation marks here, because the notion of matching needs to be defined by the choice of similarity measure to use.

⁸ The RWC database is a large database widely used in the field of MIR and is discussed in more detail in Chapter 3.

Müller (2011) identifies three levels at which the matching of a query audio fragment can occur. At the one extreme – the most specific – is what is generally referred to as *audio fingerprinting*: given a short audio fragment as query input, the aim is to identify and locate the recording within a given music collection. Related to this is *audio matching*, where the aim is to automatically retrieve all fragments that musically correspond to a short query audio fragment from all documents within a music collection – but here allowing for variations in performance, arrangement, and so on. Audio fingerprinting and audio matching are examples of fragment-level audio retrieval, as opposed to document-level retrieval where documents are compared globally. An example of document-level retrieval is the task of *cover song identification*, which aims to retrieve different versions of the same piece of music.

According to Grosche *et al.* (2012), audio fingerprinting has received the most interest of all music retrieval tasks, and is also the most widely used in commercial applications. An audio fingerprint is a “compact content-based signature” used to summarise and compare audio recordings (Cano *et al.*, 2005). Such fingerprints should be robust against distortions due to noise and compression artefacts, should be scalable and efficient to compute and should be highly specific (so that only a short audio fragment is required to reliably identify a recording) (Grosche *et al.*, 2012). Several methods for audio fingerprinting exist, but one of the most widely used is a method introduced by Wang (2003). In Wang’s approach, peaks are extracted from a spectrogram of an audio recording and then used as fingerprints. A commercial implementation of Wang’s algorithm can be found in the Shazam⁹ music identification service which, as of September 2012, can be found on a quarter of a billion mobile devices across the world¹⁰. Some of the other query-by-example search engines currently available are Musipedia¹¹, Tunebot¹² and Midomi/SoundHound¹³, to name but a few. Two other popular audio fingerprinting methods are discussed by Chandrasekhar *et al.* (2011), while references to more techniques can be found in Grosche *et al.* (2012).

⁹ <http://www.shazam.com> (accessed 15 February 2013)

¹⁰ Shazam press release 17 September 2012

(<http://www.shazam.com/music/web/pressrelease.html?nid=NEWS20120917035849>, accessed 4 March 2013)

¹¹ <http://meertens.musipedia.org> (accessed 15 February 2013)

¹² <http://tunebot.cs.northwestern.edu> (accessed 15 February 2013)

¹³ <http://www.midomi.com> and <http://www.soundhound.com> (accessed 15 February 2013)

Audio fingerprinting is considered to be solved to a large extent (Müller, 2011); however, it cannot cope with differences in, for example, tempo or instrumentation. To retrieve audio at such a lower level of specificity and therefore cater for these types of differences, audio matching techniques are used. Most audio matching procedures rely on chroma-based features (Grosche *et al.*, 2012). Chroma-based features are discussed by Müller *et al.* (2005) and, in short, basically capture energy distributions in the twelve different pitch classes of Western tonal music. Grosche *et al.* (2012) give some detail on the different ways of computing chroma features and also discuss some post-processing steps that can be taken to increase robustness.

In cover song identification (also referred to as version identification) the starting point is often also chroma-based features. While the goal is to obtain a single similarity measure to globally compare audio tracks, in practice these global comparisons are often performed locally; that is, by comparing representative samples of a track, short random samples from a track or even longest matching subsequences (Grosche *et al.*, 2012). Since cover versions can often differ significantly from the original in terms of timbre, instrumentation, harmony, tempo, tonality and so forth, it is necessary to account for such differences in any similarity search. Techniques to approach this problem are outlined in Grosche *et al.* (2012) and in Ellis and Poliner (2007).

2.12 Music synchronisation

A single piece of music can have several digital files associated with it, such as an audio recording, MIDI file, music video, digitised musical score or sheet music as well as other representations such as lyrics, tablatures or chord sheets. In turn, each of these can consist of different versions as well; for example, different performances of the same piece of music, different instruments, different editions of scores (particularly in classical music) as well as differences in tempo, dynamics, articulation and tuning, to name but a few. The aim of music synchronisation is to link all of these representations together.

The formal definition given for music synchronisation by Müller (2011) is “... a procedure which, for a given position in one representation of a piece of music, determines the corresponding position within another representation”. Practical examples of music synchronisation are aligning an audio file to sheet music, linking two audio files of the same piece of music and aligning lyrics to audio.

According to Müller (2011), there are generally two steps involved in the synchronisation process:

1. Suitable features should be extracted from the music representations under consideration. These features should be robust to variations that may be present in the files under consideration, but should still capture enough distinctive information to enable synchronisation. Chroma-based features are often used in this context since they are suitably robust.
2. The extracted features should be time-aligned; techniques such as Dynamic Time Warping (DTW) and Hidden Markov Models (HMM) (Rabiner and Juang, 1993) are often used in the alignment process.

There are two different versions of the synchronisation problem to consider here: online and offline synchronisation. Offline synchronisation takes place if both data streams are known in its entirety before the start of the synchronisation process. In contrast, online synchronisation takes place when one of these data streams is not entirely known in advance. An example of offline synchronisation is synchronising two audio recordings (for instance two different performances of the same piece of music), while examples of online synchronisation are *score following* (aligning a musical score to a live music performance; recent publications in the field are Cont, 2010 and Chou *et al.*, 2012) and *automatic accompaniment* (where a computer is to provide real-time accompaniment for a musician playing a solo part; a recent work in the field is that of Cont *et al.*, 2012). Both score following and automatic accompaniment are discussed in Dannenberg and Raphael (2006). Some software implementations are SampleSumo¹⁴, Tonara¹⁵, Antescofo¹⁶ and Music + One¹⁷.

¹⁴ <https://www.samplesumo.com/music-following> (accessed 27 August 2013)

¹⁵ <http://www.tonara.com> (accessed 27 August 2013)

¹⁶ <http://repmus.ircam.fr/antescofo> (accessed 27 August 2013)

¹⁷ <http://music-plus-one.com> (accessed 27 August 2013)

2.13 Music structure analysis

Musical structure analysis is a task closely related to the field of musicology. Whereas an important task for musicologists is to divide a piece of music into segments and to then group these segments in a meaningful way (usually working from a printed score), in the MIR field the starting point is usually an audio recording and the aim is to perform this segmentation and grouping process automatically. A simple example of structure analysis would be the automatic grouping of a piece of popular music into its components such as intro, chorus, verse and bridge. In classical music, structure is often denoted by letters referring to distinct sections of the work, with subscripts denoting a slight variation on each section. So for example, a rondo form could be denoted by $A A_1 B A_2 C A_3 B A_4$.

Three important principles in music structure analysis are repetition, novelty (contrast) and homogeneity (Müller, 2011); corresponding to this, three classes of structure analysis can be distinguished: repetition-based, novelty-based and homogeneity-based methods. Until now, many methods for music structure analysis have relied on one of these approaches; the challenge in further research is to combine these approaches to derive more accurate segmentations (Paulus *et al.*, 2010). Some attempts at this have been made (for example Paulus and Klapuri, 2009) but the problem is far from solved. A brief overview of the three different approaches will now be presented, largely following Müller (2011).

Repetition refers to recurrent patterns in music, whether that be rhythmic, melodic, harmonic or otherwise, and the repetitive structure of a piece of music often gives a clear indication of the underlying musical form. The goal of repetition-based structural analysis methods is therefore to identify recurring patterns. The first step in these approaches is to convert the audio file into a sequence of suitable audio features; chroma-based features are a popular choice. Once this has been accomplished, the aim is to find repeating subsequences, and to this extent self-similarity matrices (also referred to as self-distance matrices – see Paulus *et al.*, 2010) are derived by comparing all of the elements in the feature sequence in a pairwise fashion based on some similarity measure. Details about calculation of these matrices can be found in Paulus *et al.* (2010). From a self-similarity matrix, repetitive patterns are fairly easy to

identify visually: they will show as diagonal stripes parallel to the main diagonal. However, despite the ease of identifying these repetitive patterns visually, it is more challenging to extract these automatically due to distortions caused by variations in dynamics, instrumentation and modulation. Some form of low-pass filtering is therefore often applied in order to smooth the matrix along the diagonals. Some other approaches are also suggested by Paulus *et al.* (2010).

Contrast is introduced into music to engage the attention of the listener. For instance, a loud passage of music might be followed by a softer one, or a fast one by a slower one. The goal of novelty-based structural analysis methods is to automatically determine where in the music such changes or contrasts occur. The standard approach for doing this is to use a self-similarity matrix (as in the case of repetition-based methods), but instead of looking for diagonal stripes parallel to the main diagonal, the aim is to find 2D corner points (Müller, 2011). These corner points help to identify segment boundaries, and are located with the help of a kernel matrix of smaller dimension, which is correlated along the main diagonal of the self-similarity matrix. The resulting novelty functions can be used in conjunction with a relevant feature representation in order to obtain indicators for changes in aspects such as instrumentation, harmony and tempo. Further details and references can be found in Paulus *et al.* (2010).

Homogeneity in music refers to the fact that aspects such as instrumentation, tempo and harmony are usually fairly similar within the same section of music. The goal of homogeneity-based structural analysis methods is therefore to detect sections of music that show some degree of consistency in terms of these aspects. It is often used in conjunction with novelty-based methods, since the two are fundamentally related: a change in some musical aspect is usually preceded by a period of homogeneity. Several different approaches have been suggested, using techniques ranging from spectral clustering to hidden Markov models; details and references again can be found in Paulus *et al.* (2010).

2.14 Performance analysis

Musical performers make a piece of music their own by varying aspects such as tempo, dynamics, articulation and other expressive parameters. Therefore, a computer rendition of a composition without allowing for interpretive expressive nuances and variation will sound mechanical (Nettheim, 1997). The goal of performance analysis is to capture what makes a particular performance of a piece of music unique (and in so doing determine what is unique about the style of a particular performer), and to some extent also to determine the commonalities between different performances (in order to derive general performance rules). It has been an active subdiscipline of MIR in recent years (Müller, 2011).

Perhaps the biggest challenge in performance analysis is to somehow annotate the performance so that the exact timing and intensity of each individual note in the performance is clear. In many cases this step is performed manually, but this is a very labour-intensive process and therefore not appropriate for large audio collections. Another option is to use a computer-monitored player piano (such as the Yamaha Disklavier), but the disadvantage is that only recordings made using these specialist instruments can be used (and it is also only possible for piano performances and not for other instruments). Ideally one would therefore want to be able to automatically annotate the performance from any audio recording as source. *Beat tracking* and *onset detection* (discussed in Section 2.10) are techniques used to estimate the precise timings of each note event under consideration, and while great research effort has been expended into these techniques, results are still unsatisfactory, especially for music with weak onsets and strongly varying beat-patterns (Müller, 2011). Müller *et al.* (2009) propose a technique in which a MIDI representation or musical score is used to obtain a “neutral” representation of tempo and then synchronised with an actual performance to obtain relative tempo differences; practically however, results from this approach can still be difficult to determine, since differences could be due to synchronisation error instead of actual differences in performance. Robust synchronisation techniques are therefore essential and this is an area which still requires much further research. As Müller (2011) states: “The computer-based performance analysis... is still in its infancy requiring interdisciplinary research efforts between computer science and musicology”.

2.15 Other areas of MIR research

There are several other fields of research that can be classified under the MIR umbrella which have not been discussed above. Some of these are:

- **Optical music recognition** – This is the musical equivalent of optical character recognition (OCR) and is concerned with the creation of a digital version of a printed score. A good overview of the current state-of-the-art and research issues is given by Rebelo *et al.* (2012).
- **Melody and bass line extraction** – Melody is roughly equivalent to the part a user would whistle or hum while listening to a piece of music (Poliner *et al.*, 2007), while a bass line refers to an organised sequence of notes usually played by an instrument such as a bass guitar or double bass (Ryynänen and Klapuri, 2007). The aim of melody and / or bass line extraction is to extract or isolate the melody and / or bass line from a polyphonic audio recording. Melody extraction is discussed in Poliner *et al.* (2007), while bass line extraction is discussed in Ryynänen and Klapuri (2007). A recent paper discussing both melody and bass line extraction is Uchida and Wada (2011).
- **Chord and key recognition** – Chords and keys are some of the basic building blocks of Western music. A chord is defined as a collection of simultaneously sounding notes (Pauwels *et al.*, 2011), in some sort of harmonic relationship to each other. Key depends on concurrent and sequential notes over a period of time (Pauwels *et al.*, 2011). The concepts of key and chord are also related to a very large degree, meaning that some researchers tackle the problems of chord estimation and key detection together. Key detection is discussed by Zhu *et al.* (2005), while chord recognition is discussed in Cheng *et al.* (2008). Pauwels *et al.* (2011) approach chord and key recognition simultaneously.
- **Composition** – From the mathematical puzzle of J.S. Bach's Musical Offering, Schoenberg's twelve-tone method and Xenakis' stochastic music (experimenting with game theory, set theory and Fibonacci sequences), many composers have used mathematical principles in their compositions and many continue to do so today. Cross (2003) gives examples of composers using mathematics in their compositions. Today, there is an area of research sometimes referred to as "computer music" or "computer-aided composition", which uses algorithms to compose new music. For more details on computer-

aided composition, see the special edition of *Contemporary Music Review*, 2009.

- **Campanology** – While not strictly an MIR problem, campanology (or change-ringing) is an interesting application of mathematics to the field of music. Change-ringing is an art peculiar to the English (of the more or less 5 500 sets of bells suitable for change-ringing, about 5 200 are in England, 200 in the rest of the British Isles and only about 100 in the rest of the world; Roaf and White, 2003) and refers to the ringing of large church bells, with no intent to form a melody, but in different orders with the condition that no bell can move more than one position in successive rows. Finding all possible permutations (without repetition) is done through group theory. A good introductory paper to change-ringing is Roaf and White (2003).

This list is by no means exhaustive: there are several other areas of research which have not been mentioned. For interested readers, a good introductory overview paper to the field of MIR and some of its sub-disciplines is Casey *et al.* (2008).

2.16 Summary

This chapter started with a very brief overview of the historical links between music and mathematics. We then discussed the relatively young interdisciplinary field of MIR, followed by an explanation of musical sound and the different aspects and attributes of such sounds. The process of audio feature extraction, which in essence transforms a piece of music from raw audio format to a dataset which can be used in a data mining context, was considered next. The rest of the chapter was devoted to a discussion of some of the research areas in the field of MIR, with a particular focus on music classification. Recent advances in each area were discussed, as well as the challenges facing researchers in the different areas.

CHAPTER 3

Instrument Recognition

“If there’s any object in human experience that’s a precedent for what a computer should be like, it’s a musical instrument: a device where you can explore a huge range of possibilities through an interface that connects your mind and your body, allowing you to be emotionally authentic and expressive.”

Jaron Lanier, American computer scientist and composer

3.1 Introduction

In the previous chapter we discussed some of the main sub-disciplines of music information retrieval (MIR), with special attention given to music classification: classification of music by emotion or mood and music genre classification. We omitted one of the main music classification problems in the discussion in the previous chapter – that is, musical instrument recognition – since that is one of the focus areas of this study. In this chapter we will take an in-depth look at the problem of musical instrument recognition.

We will start by revisiting the concept of musical timbre (Section 3.2), and then take a look at the goal of musical instrument recognition (Section 3.3), followed by a brief discussion of challenges inherent in the field in Section 3.4. We will then move on to an overview of the scope of the field in Section 3.5 and then review some of the classification methods regularly used for instrument recognition in Section 3.6. We will also emphasise multi-label methods used in the field. In Section 3.7 we will review previous work in the field, with a particular focus on the polyphonic case.

Lastly, in Section 3.8, we will examine some related aspects such as features that are regularly used for instrument recognition, work on feature selection in an instrument recognition context and some interesting applications of instrument recognition.

3.2 Timbre revisited

In Chapter 2 (Section 2.4.4) the concept of musical timbre was discussed. Timbre is everything in a sound which is not pitch, amplitude or duration; in other words, timbre is what causes two instruments playing exactly the same note to sound different. This implies that timbre is a crucial concept in instrument recognition, a fact that was clearly illustrated by means of Figure 2.3.

A major determinant of the timbre of a sound, is the harmonics. In Chapter 2 we explained that the harmonics (also referred to as partials or overtones) account for the colour of a tone, and that different instruments have different harmonic profiles. For example, Müller *et al.* (2011) explain that in the clarinet the odd harmonics will be stronger than the even harmonics, while in the vibraphone mainly the 1st and 4th harmonics are present, with a small amount of energy around the 10th harmonic.

To complicate matters even further, sound produced by an instrument is not only different from that produced by other types of instruments, but will also vary with other factors such as the performer, the instrument manufacturer, the temperature and the room acoustics, to name but a few. For instance, the following figure shows the time domain and partial magnitude spectra for two violins playing the same note; however, not only are the violins by different manufacturers, but they are also played by different performers. It is clear that the harmonic profile of the two instruments are very different. (Graphs from Barbedo, 2011).

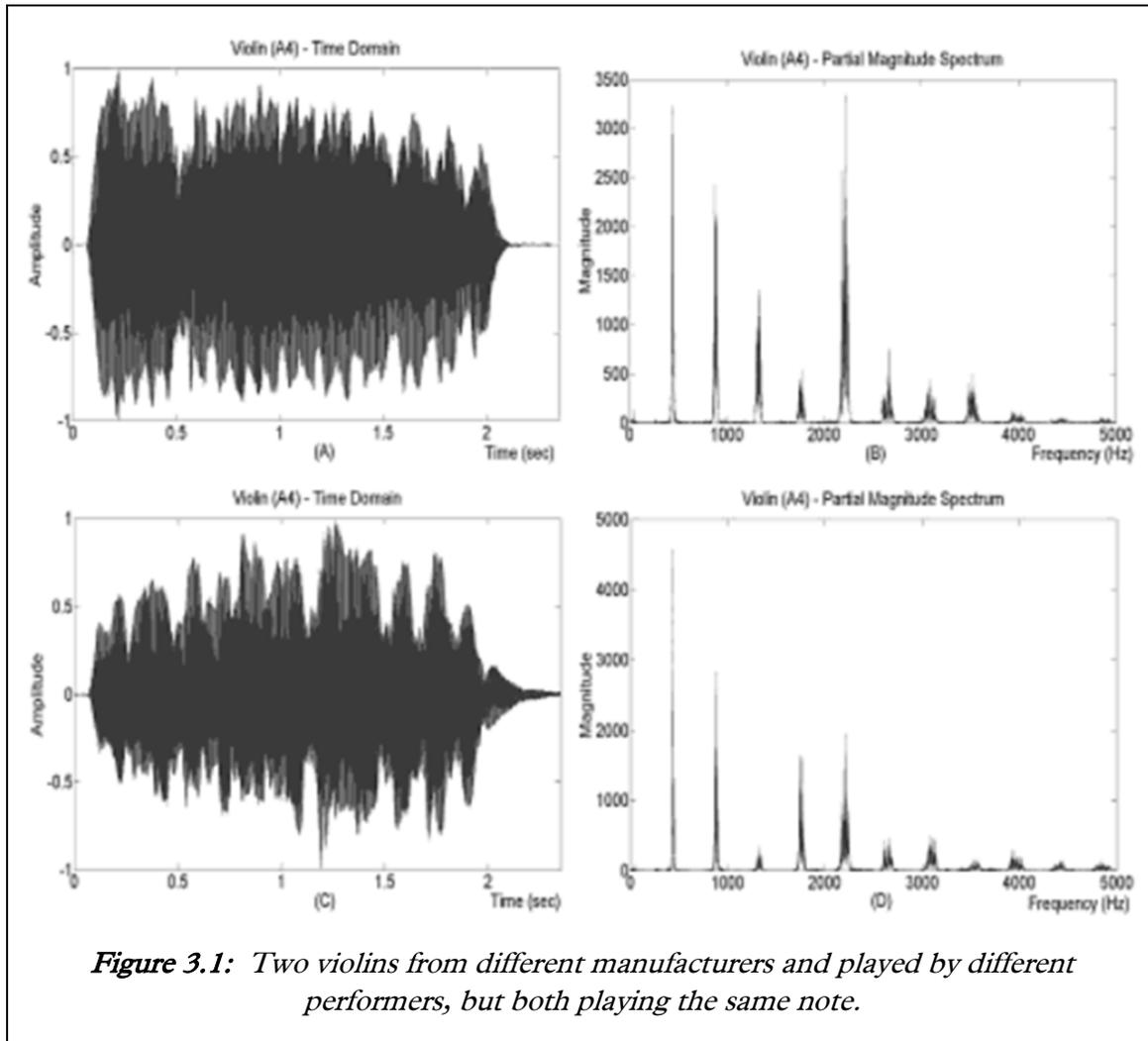


Figure 3.1: *Two violins from different manufacturers and played by different performers, but both playing the same note.*

Lastly, whereas in speech there is one sound producing mechanism, in music there can be several, for example vibrating strings, air columns, bars, etc. The source excitation can provide valuable information about the instrument identity (Müller *et al.*, 2011).

This all serves to illustrate the complexity of the instrument recognition problem. Not only should all of the above timbral characteristics be captured in the features used in instrument classification, but any method for successful instrument recognition should be able to generalise well enough to account for differences encountered between the same instruments.

3.3 Goal of musical instrument recognition

The goal of (machine) musical instrument recognition is to automatically determine the instrument or instruments playing in a given audio signal. There can be one instrument playing at a time, in which case the problem is referred to as *monophonic*, or there can be two or more instruments playing simultaneously, in which case the problem is called *polyphonic*. In addition, the audio signal can range from isolated notes, to musical phrases or entire compositions. Whereas humans can identify instruments with some ease under the right conditions (but with some restrictions; see for example Srinivasan *et al.*, 2002), automatic identification of instruments has proved to be a complex problem. Early work in the field focused on the monophonic case, and advances have been made up to a point where recognition rates as high as 100% are obtained (under certain restrictive conditions; see for example Cemgil and Grgen, 1997). However, the monophonic problem is still not considered to be solved (Barbedo, 2011). Recent work in the field has tended to focus more on the polyphonic case though, which is substantially more complex than the monophonic one. The work in this thesis is a contribution to this very active sub-discipline of MIR research.

Successful instrument recognition techniques could be used to aid in music retrieval. So, for example, a user could search for music containing “violin solos” or “piano and flute sonatas”. Work in instrument recognition could also aid research in other sub-disciplines of MIR; similarly, work in other fields could enhance research in instrument recognition, since – as was seen in Chapter 2 – there often exists an interdependence between sub-disciplines in the MIR field. In this manner, instrument recognition could serve as a first step towards genre classification, given that certain instruments or instrument combinations tend to be prevalent in certain musical genres. For example, a trio of piano, double bass and drums is much more likely to appear in jazz music than in other music. Similarly, vocals, lead guitar, bass guitar and drums often appear together in rock / pop music, while piano, violin and cello are usually found together in a piano trio (and hence classical music). Conversely, knowing the genre of the music could help narrow down the possible instrument combinations. The field of automatic music transcription also relies heavily on instrument recognition, as correctly identifying the instruments playing

would be one of the crucial first steps in transcribing the music. Other fields that are closely related to instrument recognition are sound source separation and multiple-F0 (pitch) estimation. As Barbedo (2011) states: "... the area of music processing as a whole can benefit from advances in instrument recognition".

3.4 Challenges in automatic instrument recognition

Several challenges inherent to MIR that were mentioned in the previous chapter present themselves in an instrument recognition context too. In addition, there are some other challenges which are specific to instrument recognition.

Possibly the biggest overall challenge in instrument recognition problems, lies in the fact that there are no standard, benchmark datasets which are used for such tasks. This makes the comparison of results across different studies difficult, since different researchers tend to use data with different instrument types, different numbers of instruments, different combinations of instruments, and so on. Objectively comparing classification methods, feature selection methods and other methodology is therefore all but impossible. Nevertheless, there are a number of databases that are very widely used across research in the field, namely:

- **RWC:** The Real World Computing (RWC) Database¹⁸ is a copyright-cleared database, available to researchers at a small fee. It consists of several different music collections suited to different tasks in the MIR field. The collection of most interest for the purpose of instrument recognition is the Musical Instrument Sound Database, which contains 50 instruments with 3 variations per instrument (i.e. different instrument manufacturers and different musicians). It also covers the full pitch range of each instrument, as well as incorporating different playing styles and dynamics.
- **MUMS:** The McGill University Master Sample (MUMS) collection is a commercial database consisting of 6 546 sound samples, also covering a range of instruments, pitches, playing styles and dynamics (Eerola and Ferrer, 2008). The database has recently been acquired by Garritan Music which means that

¹⁸ <http://staff.aist.go.jp/m.goto/RWC-MDB> (accessed 16 April 2013)

it is no longer freely available for research purposes. However, it is widely used in many instrument recognition studies (for example Wieczorkowska *et al.*, 2011, Loughran *et al.*, 2008a and Eronen, 2001).

- **IOWA:** The University of Iowa Musical Instrument Samples database¹⁹ contains recordings of a number of orchestral instruments, across a variety of playing styles and dynamics, and also across the full pitch range of the instrument. It is freely available to researchers.

These three databases (either individually or combined) have been used in a large proportion of research in the field of instrument recognition. However, these are not databases that are generally used in their entirety; researchers tend to extract data and artificially mix sound samples from them to suit their research requirements, which means that comparisons are still difficult. Some researchers also create their own datasets to use, but again this is problematic – not only because it means that direct comparisons of results with other research are not possible, but also because issues surrounding music copyright and intellectual property rights mean that the data can often not be freely distributed or shared.

Labelling training and test data is also challenging. Although instrument recognition does not suffer the same subjectivity challenge as disciplines such as music genre or emotion classification (in these fields, there is not necessarily a correct ground truth, and the labelling is therefore subjective), it can still be quite labour-intensive to manually annotate samples with the correct instrument(s).

A substantial difficulty concerning polyphonic instrument recognition, is the problem of overlapping harmonics (often referred to as overlapping partials). When two or more instruments are playing together, they will generally do so in harmony (recall from Chapter 2 that sounds are in harmony when their fundamental frequencies are related by small integer ratios, e.g. 2:1, 3:2, 5:4). In addition, harmonics of a musical sound tend to occur at integer multiples of the fundamental frequency (see Section 2.4.4). Consequently, when two or more instruments are playing in harmony, many of their harmonics will overlap which makes separation difficult. The following figure illustrates how overlapping harmonics make it difficult to recognise the individual

¹⁹ <http://theremin.music.uiowa.edu/MIS.html> (accessed 16 April 2013)

instruments once the sounds have been mixed. Graphs are taken from Wieczorkowska and Kubera (2010), and show the spectra for flute and trumpet playing the same pitch, first individually and then together.

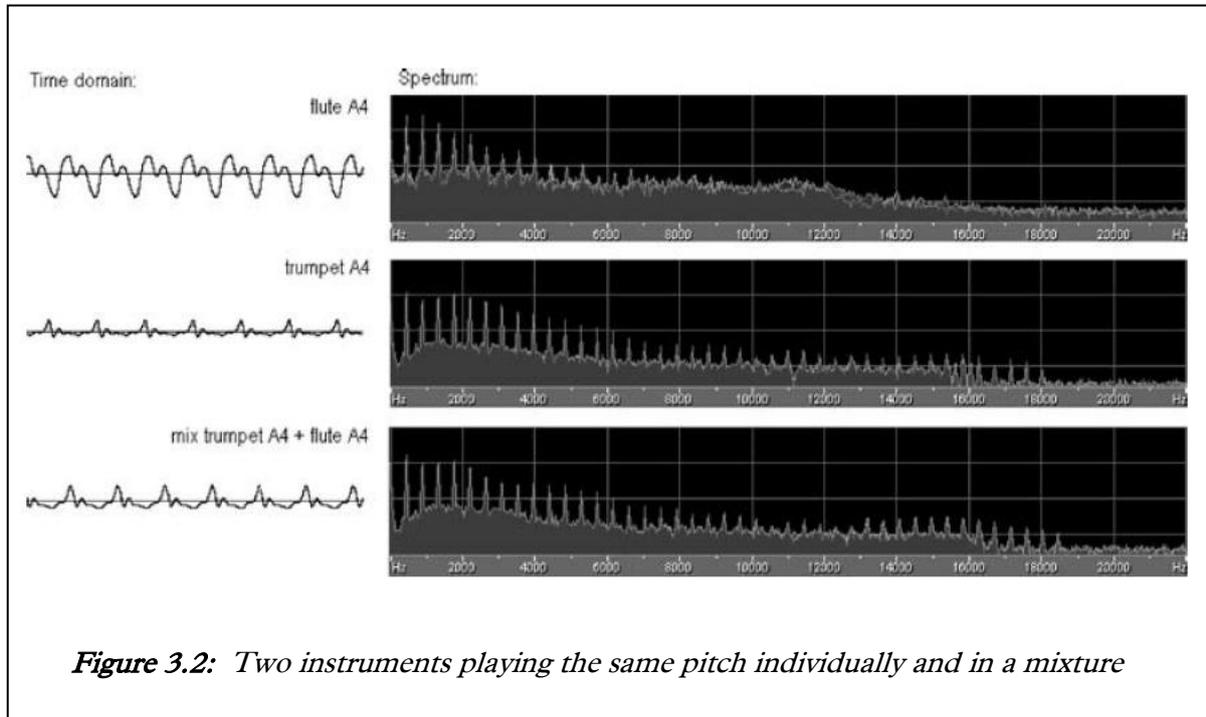


Figure 3.2: Two instruments playing the same pitch individually and in a mixture

The problem is even more pronounced if the instruments are not played at the same amplitude, since the louder instrument(s) may mask the softer instrument(s) at the overlapping harmonics (Kitahara *et al.*, 2007b; Heittola *et al.*, 2009). Some work has been done on overlapping harmonics in particular (see for example Fabiani, 2010), while some studies on instrument recognition approaches the problem of overlapping harmonics by means of weighting features based on how much they are affected by overlapping (Kitahara *et al.*, 2007b) or by excluding such frequency regions from the model altogether (Eggink and Brown, 2003).

Another dilemma with polyphonic problems centres around the number of instruments. In most “real-world” problems, the number of instruments present in any given sound segment or sample is not known a priori and needs to be estimated first. However, not many studies have focused on this aspect of the problem; Barbedo *et al.* (2009) is one of the few studies in this regard. Furthermore, in most pieces of polyphonic music, the number of instruments playing is not static throughout. In a typical classical orchestral work, there will generally be 10 or more types of

instruments present – often many more – and not all of them will be playing all of the time. In addition, the number of instruments in an orchestral work would generally be greater than the number of instrument classes, since there will be several of each instrument class playing; for instance, in a typical modern symphony orchestra there will be at least 30 violins. To add to the complexity, most orchestral pieces have a first and second violin part, which means that the violins will not all be playing the same part. The same is true of many other instruments as well. To illustrate this point, Figure 3.3 shows an excerpt from the score of Beethoven’s Ninth Symphony (opus 125, first movement, Unger edition). Depending on which part of the sound signal of this work was to be considered, the number of instrument classes would greatly vary. For instance, in the first bar of this excerpt, only 5 instrument classes are playing (oboe, clarinet, horn, violin and cello). The fifth bar contains 9 instrument classes playing together (flute, oboe, clarinet, bassoon, horn, violin, viola, cello and double bass). In the last bar of the excerpt, 11 instrument classes are playing (flute, oboe, clarinet, bassoon, horn, trumpet, timpani, violin, viola, cello and double bass). The entire symphony contains 17 instrument classes as well as voices in solo and choir, with some only present in the fourth movement of the work. This has major implications for the feature extraction stage of any track-level instrument recognition problem, as sampling should be done in such a way as to ensure that it is representative of the whole work.

First bar

Fifth bar

Last bar

The image displays a musical score excerpt for the first movement of Ludwig van Beethoven's Symphony Number 9, Opus 125. The score is arranged in two systems. The first system includes staves for Flute (Fl.), Oboe (Ob.), Clarinet (Cl.), Bassoon (Fag.), Cor Anglais (Cor. D.), Cymbals (Cyg. B.), Trumpet (Tr.), and Trombone (Tp.). The second system includes staves for Violin I, Violin II, Viola, and Cello/Double Bass. The score shows a crescendo from the first bar to the fifth bar, followed by a dynamic shift to fortissimo (ff) for the remainder of the excerpt. Red arrows point to the first, fifth, and last bars of the excerpt.

Figure 3.3: Excerpt from the score of the first movement of Ludwig van Beethoven's Symphony Number 9, opus 125. The excerpt is the second page of the Unger edition (Leipzig: Ernst Eulenburg, Ed.411, n.d. (ca.1935). Plate E.E. 3611).

3.5 Instrument recognition: scope and approaches

In any instrument recognition problem there are a number of factors that should be decided on at the outset of the study, which will determine and influence the scope of the study and also how the problem will be approached. Some of the main variabilities in any instrument recognition study are presented in the table below:

Table 3.1: Factors in instrument recognition studies

Signal complexity	Monophonic	Polyphonic
Instrument types	Pitched instruments	Non-pitched instruments
Feature extraction	First separate sources and then treat as monophonic problem	Work directly on polyphonic signal
Choice of data	Artificial or bespoke for purpose	Commercial (“real-world”) recordings
Taxonomy	Hierarchical	Flat

In addition, decisions need to be made regarding feature sets to be used, whether feature selection or some form of dimensionality reduction is needed, as well as the classifier used.

Some of these factors will now be discussed in a little more detail.

3.5.1 Signal complexity

As explained in Section 3.3, a piece of music can be considered *monophonic* (only one instrument playing) or *polyphonic* (more than one instrument playing simultaneously). Monophonic signals can consist of isolated sounds (one instrument playing one note) or solo phrases (one instrument playing more than one note). Polyphonic signals can vary with respect to degree of polyphony: from the simplest case of just two instruments playing together simultaneously (duet) to very complex polyphonies (for example a full orchestra of instruments). Instrument recognition algorithms tend to deal with one or the other, although more recent studies in the field have tended to focus on the polyphonic case.

In the monophonic case, there are advantages to working with isolated sounds: it is the simplest signal form – which makes feature extraction easier – and data is fairly easily obtainable from one of the publicly available databases discussed in Section 3.4. However, a disadvantage is that the task of instrument recognition may actually be more difficult for isolated tones than for solo phrases, since there are no note transitions which could aid in identifying instruments (Essid *et al.*, 2006c).

On the other hand, although note transitions could be helpful in identifying instruments, they complicate the feature extraction process, since features should be extracted from homogeneous sections of the signal. It is therefore important for the feature extraction process to identify note transitions correctly, which is not a trivial task, especially for instruments such as the violin which has fairly smooth note transitions (Barbedo, 2011).

Fairly good recognition rates are reported in the literature for monophonic problems, depending to a large extent on the number and type of instruments considered, as well as the type of audio signal used. As mentioned before, the polyphonic case is a lot more complex, due to a variety of reasons such as overlapping harmonics (which was discussed in Section 3.4). Although monophonic problems are often considered to be a training ground for the more complex polyphonic ones, as Richard *et al.* (2007) point out, in most cases the methods designed for the monophonic case will not directly work on the polyphonic one. This is due to the fact that the feature extraction process is not linear, so additivity of the different sources cannot be assumed.

3.5.2 Instrument types

Pitched instruments should be approached differently from non-pitched instruments, due to fundamental spectral differences: non-pitched instruments have noise-like spectral content (also see Section 2.4). As a bridge between pitched and non-pitched instruments there are also pitched percussion instruments, which have a defined pitch but partials which are non-harmonic. The following figure illustrates the difference between pitched and non-pitched instruments by way of their magnitude spectra (graphs from Barbedo, 2011):

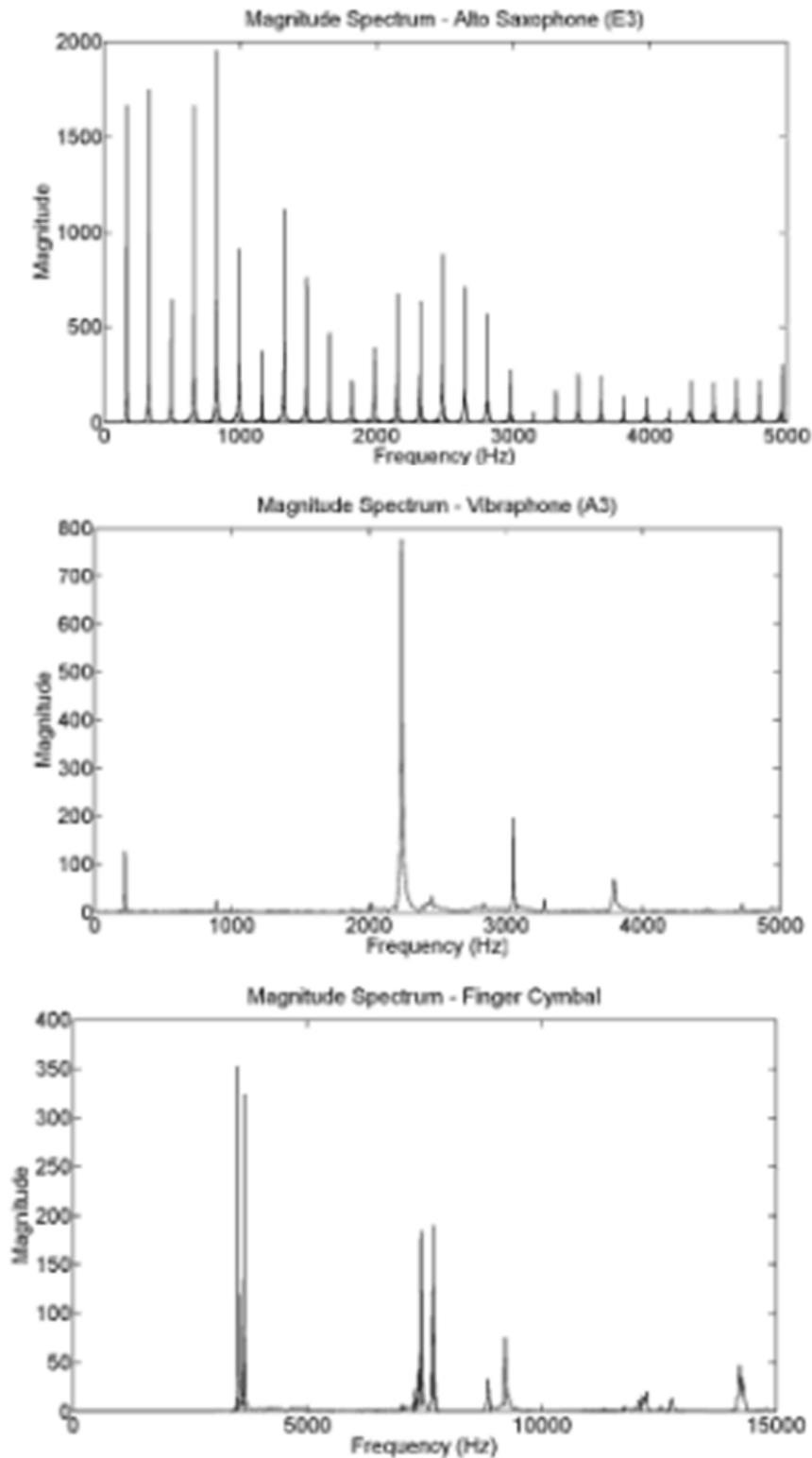


Figure 3.4: Magnitude spectra for pitched instrument (top), pitched percussion (middle) and non-pitched (bottom). The pitched instrument (saxophone) shows very clear and regular partials, while the pitched percussion (vibraphone) shows a definite fundamental frequency but no regularity to the partials. The non-pitched instrument (finger cymbal) shows no discernible pattern among its partials.

Figures from Barbedo (2011)

The type of instruments considered usually varies between studies. Most work has been done on pitched instruments, but there have also been successful attempts at classifying non-pitched instruments (see for example Herrera *et al.*, 2002). As yet, very few models are able to handle both pitched and percussive instruments in one model; one of the few studies in this regard is Fuhrmann *et al.* (2009) – few other studies have been able to generalise to this extent.

In terms of pitched instruments, most studies have focused on Western orchestral instruments, and relatively little has been done in terms of recognising ethnic instruments (see for example Gunasekaran and Revathy, 2008 and Lidy *et al.*, 2010). In addition, there is reason to believe that existing methods do not necessarily generalise well to non-Western music (Moelants *et al.*, 2006; Lidy *et al.*, 2010).

3.5.3 Feature extraction

One of the most important initial decisions in a polyphonic instrument recognition problem, is how audio features will be extracted. There are two approaches: features can be extracted directly from the mixed signal, or the features can be extracted for each instrument individually after an attempt has been made to separate the polyphonic signal into its components. The latter therefore depends on a sound source separation pre-processing step and effectively transforms the classification problem into a monophonic one.

Whereas sound source separation has the distinct advantage of reducing the complexity of the problem faced by the classifier, it does introduce additional complexity in terms of the separation algorithm. Sound source separation is also not a problem that is considered to be solved, which means that any errors inherent in the initial separation step will follow through into the classifier step. Fu *et al.* (2011) state that it is still an open question whether source separation methods can aid and improve the performance of instrument recognition. Nonetheless, some success with sound source separation as a prior step to instrument recognition has been obtained by authors such as Heittola *et al.* (2009) and Bosch *et al.* (2012), and it should be kept in mind that even a flawed separation process can still lead to satisfactory results

(Barbedo, 2011). Non-negative matrix factorisation (NMF) is often used for sound source separation (Smaragdis and Brown, 2003; Wang and Plumbley, 2006; Virtanen, 2007).

When extracting features directly from the mixed signal, it is important to keep in mind that features should be designed in such a way that the effect of interference between instruments is minimised (Barbedo, 2011). Approaching instrument recognition from this perspective effectively positions the problem as a multi-label classification one (although many authors do not use traditional multi-label classification approaches). Examples of instrument recognition approaches that avoid the need for prior sound source separation are Richard *et al.* (2007) and Spyromitros-Xioufis *et al.* (2011) (see Section 3.7).

A related approach is to identify regions where there are little or no interference between instruments; this is done by either locating regions in the signal where a single instrument is playing in isolation (Barbedo and Tzanetakis, 2011), or searching for regions where the effect of overlapping harmonics are minimal and a single instrument dominates (Eggink and Brown, 2003).

Little and Pardo (2008) take a novel approach to learning from polyphonic data, avoiding trying to mitigate the effect of interfering instruments but instead working directly from weakly labelled polyphonic data; that is, only the presence or absence of the target instrument is indicated in an audio clip. This makes it possible for them to train a classifier directly on mixture data. In their study, they obtained significantly better results learning from weakly-labelled mixture data than learning from isolated examples.

3.5.4 Choice of data

As was mentioned in Section 3.4, one of the biggest challenges in automatic musical instrument recognition is the lack of benchmark datasets and the difficulty in obtaining suitable datasets to work with. To this extent, many researchers tend to create datasets to suit their own purposes, often with one of the publicly available

datasets as starting point, extracting instruments, playing styles and other parameters as required. However, since samples in these databases are monophonic, they cannot be used in their given format for polyphonic problems. Many researchers therefore artificially mix samples from these databases to obtain polyphonic training or testing cases (for example, Wiczorkowska *et al*, 2008).

The most popular, publicly available datasets for instrument recognition were described in Section 3.4. Many of the studies in the field use one or more of these datasets in one form or another. While they are extremely useful for the purpose at hand, the conditions under which these were recorded were artificially controlled to include as little noise interference as possible. This of course is not the case when commercial, or “real-world” recordings are used. More and more authors are starting to test their models on commercial recordings, for example Kubera *et al.* (2010), Barbedo and Tzanetakis (2011) and Bosch *et al.* (2012), but as yet accuracy is much lower on such recordings than when using samples from custom-built databases.

Another important point to consider when training classifiers to identify instruments from polyphonic signals, is whether single instruments or instrument combinations should be used as training entities. When approaches such as pre-processing by sound source separation or attempting to mitigate the effect of overlapping harmonics are taken, training from single instrument data is a logical choice. However, there is mounting evidence that when the aim is to identify instruments from mixtures, it is best to learn from mixtures as well. Evidence in this respect is found in Wiczorkowska and Kubera (2010) and Spyromitros-Xioufis *et al.* (2011).

In addition, it is also important to ensure that samples for testing and training come from different databases, as was shown by Livshin and Rodet (2003).

3.5.5 Taxonomy

There is a natural taxonomy to musical instruments; that is, certain instruments are often grouped together. Western orchestral instruments are often informally grouped into strings, woodwinds, brass and percussion. However, although such a taxonomy

is appealingly simple, a drawback is that some instruments do not fit neatly into this classification system. For instance, the piano has strings, but the strings are struck by a hammer – should it therefore be classified with strings or percussion? A fifth category, keyboards, is subsequently sometimes added, but this is not necessarily more logical: the harpsichord and piano are both keyboard instruments, but produce sound in very different ways. In addition, with modern-day instruments this classification system technically no longer rings true: for instance, instruments that are classified as woodwinds (such as flutes and clarinets), are predominantly made from metal these days, and not wood.

A very widely used and accepted system to classify musical instruments, is the Hornbostel-Sachs system (Von Hornbostel and Sachs, 1961), which classifies instruments according to the way they produce sound. It consists of five top-level categories, with several levels below each, and more than 300 categories overall. One of the main advantages of this taxonomy is that it is not limited to western orchestral instruments, but can be used for any instrument.

The five top-levels (see Figure 3.5 for examples of each) are:

- Idiophones – these are mainly percussion instruments, which produce sound by the instruments themselves vibrating (instead of strings, a column of air or a membrane vibrating).
- Membranophones – sound is produced by the vibration of a tightly-stretched membrane. Drums fall into the membranophone category.
- Chordophones – sound is produced by the vibration of one or more strings.
- Aerophones – sound is produced by vibrating air.
- Electrophones – this category was added at a later stage, and covers all instruments involving electricity (Sachs, 1940).

The following diagram shows the top two levels of the Hornbostel-Sachs system, and gives examples of Western orchestral instruments in each category.

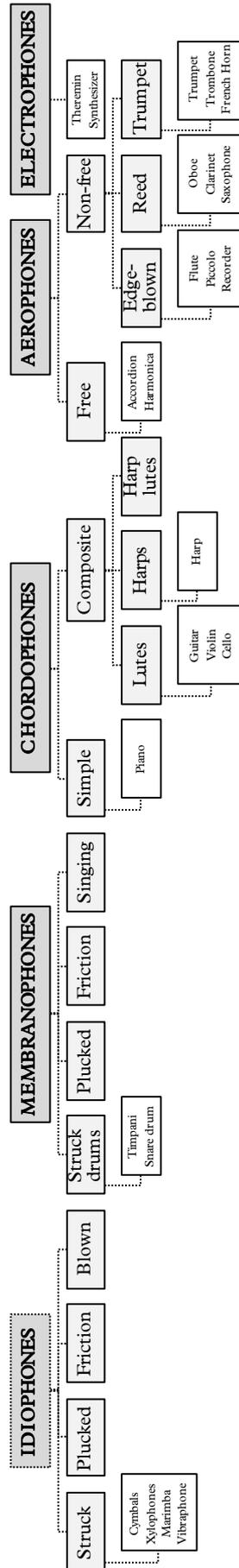


Figure 3.5: Hornbostel-Sachs taxonomy for musical instrument classification

Intuitively, a strong argument can be made for the use of hierarchical classification systems incorporating taxonomies such as Hornbostel-Sachs for the automatic classification of musical instruments. Classification at a higher level in the classification system (that is, classifying instruments into groups rather than attempting to identify them at an instrument level) is often easier and it also reduces the number of possible categories that have to be considered at any given point during classification. The major drawback of such a hierarchical classification system is that errors made at a higher level are then propagated down to the next level in the system, leading to cascading errors.

Some of the studies that have looked at hierarchical classification in instrument recognition are Eronen and Klapuri (2000), Zhang and Ras (2007a) and Jiang *et al.* (2010), and results are mixed. Eronen and Klapuri (2000) find no significant advantages in using a hierarchical approach in their study. Zhang and Ras (2007a) find some improvement with hierarchical classification, but not for instruments in the string and woodwind families. In Jiang *et al.* (2010) on the other hand, hierarchical classifiers (based on the Hornbostel-Sachs taxonomy) are found to outperform standard classifiers.

Essid *et al.* (2006b) compare a natural taxonomy (based on organisation by instrument families) with an automatic one (built by hierarchical clustering), and find that classification based on the automatic taxonomy is only slightly better than classification using a natural taxonomy. They conclude that the hierarchy used for classification should be designed in such a way that instruments which are often confused are placed in nodes which are far apart. Jiang *et al.* (2010) also recommend basing a clustering system for hierarchical classification on a machine-perspective rather than a human one; a similar finding was made by Kubera *et al.* (2013).

Barbedo (2011) summarises some of the advantages and disadvantages of hierarchical classification systems versus flat classification systems, but concludes that the choice of one or the other seems to be down to personal preference rather than enhanced performance or efficiency.

3.6 Classification methods

3.6.1 Commonly used classifiers

As was discussed in the previous section, the scope of instrument recognition studies tends to vary widely with respect to a number of different aspects. Consequently, the classifiers used also tend to vary widely.

To the best of our knowledge, no study exists with the specific aim of comparing a wide choice of different classifiers in an instrument recognition context – other than an unpublished study of limited scope by Simmermacher *et al.* (2006). However, many studies have looked at a few classifiers under similar conditions and differences found have not been substantial. Barbedo (2011) states that in instrument recognition it might be difficult to make advances regarding the classifier used and suggests that the choice of classifier may not have such a big effect on accuracy.

The support vector machine (SVM) appears to be the most widely-used classifier for instrument recognition problems. Other often-encountered classifiers are k-Nearest Neighbours (kNN), Gaussian mixture models (GMM) and decision trees. Each of these will now be discussed briefly; more details on each of these methods can be found in Hastie *et al.* (2009).

3.6.2 Support vector machines

Over the past decade or so, support vector machines (SVMs) have found their place as one of the most popular machine learning methods for classification and regression. Although training an SVM can be complex and time-consuming, once this has been accomplished classification of new cases can be done very quickly. SVMs are also able to generalise well.

The SVM can be characterised in different ways, for example as the solution to a regularised loss function minimisation problem. In this section, however, we present what is probably the best known of these, namely characterisation as the margin

maximising hyperplane. Consider first a binary classification problem for which training data $\{(\mathbf{x}_i, y_i), i = 1, 2, \dots, N\}$ are available. The class indicator values are $y_i \in \{-1, 1\}, i = 1, 2, \dots, N$, while $\mathbf{x}_i, i = 1, 2, \dots, N$ are data or input vectors in a p -dimensional space. The simplest example of an SVM arises when the training data in the two classes are linearly separable, implying that at least one hyperplane can be found which will perfectly separate the training cases into their respective classes. A hyperplane, a generalisation of the concept of a line to higher dimensions, is a set $\{\mathbf{x}: \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$, where $\langle \mathbf{w}, \mathbf{x} \rangle$ denotes the usual inner product between two vectors. Specifying a hyperplane entails specifying its slope vector \mathbf{w} and its intercept b . The rationale underlying the SVM for binary, linearly separable data is to find the separating hyperplane which has maximum margin. The margin of a separating hyperplane is defined to be the (positive) distance between the hyperplane and the data point closest to it. Straightforward arguments (see for example Chapter 5 of Hastie *et al.*, 2009) lead to the conclusion that the SVM hyperplane (that is, the maximal margin separating hyperplane) can be found by maximising $2/\|\mathbf{w}\|$, subject to the constraints

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \forall i \quad (3.1)$$

This is a constrained optimisation problem which can be solved by introducing Lagrange multipliers, denoted by $\alpha_i, i = 1, 2, \dots, N$. The SVM hyperplane is found to be of the form

$$\left\{ \mathbf{x}: \sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b = 0 \right\}.$$

An interesting and important property of the SVM is that typically a sizeable proportion of the Lagrange multipliers $\alpha_1, \alpha_2, \dots, \alpha_N$ turn out to be zero. In the expression above only those data points for which $\alpha_i > 0$ therefore contribute to the summation. These points are called support vectors and it is this sparseness property which makes fast computation of the SVM possible. Classification of a new data case with input vector \mathbf{x} is accomplished by computing

$$\text{sign} \left(\sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right).$$

In the case of non-linearly separable training data, the constraints in (3.1) are relaxed by introducing non-negative slack variables $\xi_i, i = 1, 2, \dots, N$, thereby allowing data points to lie on the “wrong” side of the margin, or even the “wrong” side of the decision boundary. The constraints therefore become

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \forall i$$

Solving the constrained optimisation problem in this case is once again accomplished by introducing non-negative Lagrange multipliers. It should also be noted that the total amount of slack, $\sum_{i=1}^N \xi_i$, is incorporated into the quantity to be optimised in order to exclude hyperplanes for which data points stray too far on the wrong side. This implies the specification of a so-called cost parameter, C , which essentially controls the balance between goodness-of-fit and complexity of the classifier. Although the process is slightly more complicated than before, the solution turns out to be of exactly the same form as previously. More details can be found in Chapter 12 of Hastie *et al.* (2009).

By construction the SVM classifier considered so far turned out to be a hyperplane in input space; in other words, a linear function of the input vector \mathbf{x} . More flexible classifiers can be obtained by allowing these to be non-linear functions. Conceptually this entails mapping the input space (in which the input vectors reside) to a higher dimensional space of features and then fitting an SVM hyperplane as before, but now in feature space (in other words the transformed version of input space). The fact that the SVM classifier

$$\text{sign} \left(\sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right)$$

only requires evaluation of inner products between input vectors, enables one to make the transition from input to feature space quite smoothly. In fact, the well-known *kernel trick* can be applied, implying that every inner product $\langle \mathbf{x}_i, \mathbf{x} \rangle$ is simply replaced by a kernel function $K(\mathbf{x}_i, \mathbf{x})$ (Hastie *et al.*, 2009). Deriving the SVM proceeds as before, and the resulting classifier turns out to be

$$\text{sign}\left(\sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right).$$

There are many options available as far as choosing a kernel function is concerned, details of which will not be discussed here, other than to mention that the radial basis function (RBF) and polynomial kernels are popular choices. In these kernel functions, as in most others, parameters appear which need to be specified beforehand or determined from the data, usually by means of cross-validation. Chapter 12 of Hastie *et al.* (2009) provides a more detailed discussion.

The SVM, as defined above, is a binary classification technique; to facilitate multi-class classification the most common approaches are to transform the problem into several binary problems and then use a one-versus-one or one-versus-all approach. A technique by Hastie and Tibshirani (1998) allows for the coupling of pairwise decisions. To derive posterior class probabilities after SVM classification, an approach by Platt (2000) is often adopted. A good source for further reading on SVMs (other than Hastie *et al.*, 2009) is Burges (1998).

Kim *et al.* (2005) list the advantages and disadvantages of SVMs, but a major advantage of SVMs is that once an SVM has been trained, computation only depends on a small number of support vectors and is usually fast. This also implies that an SVM is robust to changes of all vectors other than the support vectors.

SVM implementations in the instrument recognition literature can be found, amongst others, in Essid *et al.* (2006a, 2006b, 2006c), Simmermacher *et al.* (2006), Benetos *et al.* (2007), Deng *et al.* (2008), Little and Pardo (2008), Joder *et al.* (2009), Morvidone *et al.* (2010), Barbedo and Tzanetakis (2011), Fuhrmann and Herrera (2011), Wu *et al.* (2011), Wiczorkowska *et al.* (2011) and Bosch *et al.* (2012).

The RBF kernel appears to be the most widely used in instrument recognition literature (Essid *et al.*, 2006b and 2006c; Deng *et al.*, 2008; Little and Pardo, 2008; Joder *et al.*, 2009; Morvidone *et al.*, 2010; Wiczorkowska *et al.*, 2011 and Wu *et al.*, 2011). Polynomial kernels are used by Simmermacher *et al.* (2006), Essid *et al.* (2006c), Benetos *et al.* (2007) and Bosch *et al.* (2012).

Finally, values of the cost parameter C used in the instrument recognition studies mentioned range from 0.1 (Bosch *et al.*, 2012) to 100 (Little and Pardo, 2008; Deng *et al.*, 2008). As with the tuning parameters for the kernels, C is often found through cross-validation.

3.6.3 k-Nearest Neighbours

k-Nearest Neighbours (kNN) is a very popular classification technique, and Herrera *et al.* (2006) suggest that it should be used as a benchmark when comparing different classification algorithms in instrument recognition problems.

It is an instance-based learning technique, which means that it is not based on any statistical model. Briefly, for every data point x_0 , the aim is to find the k data points $x_{(r)}$, $r = 1, \dots, k$ which are closest to x_0 in terms of some chosen distance measure. Point x_0 is then classified using a majority voting system among its k neighbours; in other words, it is assigned to the class most common among its neighbours. Ties are generally broken at random and features are usually standardised. kNN is often successful where the number of possible classes is very large, or not known beforehand.

A major advantage of kNN is the ease with which it can be implemented, which is probably largely why it is so often utilised for instrument recognition problems. It requires very few parameters that need to be specified or tuned: only the number of neighbours k and the distance measure to be used need to be specified. It does have drawbacks though; Herrera *et al.* (2006) list these drawbacks as:

- Big demand on memory, since all training instances need to be stored.
- Significant computational load whenever a new query is processed.
- Highly sensitive to irrelevant features.
- Only based on local information, so does not provide a generalisation technique.

Some of the studies that have utilised kNN for instrument recognition are Jinchaitra (2004), Röver *et al.* (2005), Simmermacher *et al.* (2006), Deng *et al.* (2008), Little and Pardo (2008) and Jiang *et al.* (2010).

The number of neighbours k is usually determined empirically; choices of k that have been used in instrument recognition studies are $k = 1$ (Deng *et al.*, 2008), $k = 3$ (Simmermacher *et al.*, 2006; Jiang *et al.*, 2010), $k = 5$ (Little and Pardo, 2008; Jiang *et al.*, 2009b), $k = 7$ (Jinchaitra, 2004) and large values of k , varying from 33 to 79, depending on the circumstances under consideration (Livshin and Rodet, 2004).

In terms of the distance measure, Euclidean distance is the most common one used. It is sometimes useful to weight the contributions of the neighbours, so that more importance is attached to votes of the neighbours who are closer; Deng *et al.* (2008) implement such a weighting scheme by using the reciprocals of distances as weights. Jinchaitra (2004) uses Mahalanobis distance to deal with the different scaling and correlation among audio features and finds that it consistently gives 2-3% better results than using a Euclidean distance measure.

3.6.4 Gaussian mixture models

Gaussian mixture models (GMMs) model the probability density function of an observed feature vector \mathbf{x} as a weighted combination of a number of Gaussian component probability density functions. Theoretically, mixture models can use any component densities in place of Gaussians, but Gaussians are the most popular (Hastie *et al.*, 2009). Mathematically:

$$f(\mathbf{x}) = \sum_{m=1}^M \alpha_m \Phi_m(\mathbf{x}, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \quad (3.2)$$

where each of the M Gaussian densities (also called centres) Φ_m are characterised by mean vector $\boldsymbol{\mu}_m$, covariance matrix $\boldsymbol{\Sigma}_m$ and mixing coefficient α_m (with $\sum_{m=1}^M \alpha_m = 1$). These parameters are estimated during training, usually by means of the Expectation Maximisation (EM) algorithm (Hastie *et al.*, 2009, pp. 272-279). The covariance matrix $\boldsymbol{\Sigma}_m$ is often assumed to be diagonal – frequently an invalid assumption, but nevertheless widely used for simplification.

Each GMM is used to estimate the probability that the feature vector was generated by the instrument associated with that GMM, and the final classification is given by the instrument with the greatest probability. In other words, classification is performed using the maximum a posteriori (MAP) decision rule, written as

$$\hat{K} = \arg \max_{1 \leq k \leq K} \sum_{l=1}^L \log p(\mathbf{x}_l)$$

where K is the number of classes, L is the total number of observations considered, \mathbf{x}_l is the feature vector observed at time l and $p(\mathbf{x})$ is the function $f(\mathbf{x})$ as defined in (3.2) above. More details can be found in Hastie *et al.* (2009).

Kim *et al.* (2005) consider the major advantages of GMMs to be that they are computationally inexpensive, and are based on a well-understood statistical model.

Instrument recognition studies which have used GMMs for classification include Eggink and Brown (2003 and 2004), Jinchahitra (2004), Essid *et al.* (2006a and 2006c), Heittola *et al.* (2009), Joder *et al.* (2009) and Zlatintsi and Maragos (2011).

Eggink and Brown (2003) have extended GMMs to account for their missing feature approach. In their approach, in the case where some components of the feature vector are missing or unreliable, the probability density function can be computed from partial data, where only the reliable features are included. Eggink and Brown (2004) also tested use of the full covariance matrix versus a diagonal one, and found that models using the full covariance matrix tended to outperform models using a diagonal one by 10 – 20%. They found that performance using a diagonal covariance matrix could be improved by using more centres, but at the cost of requiring more training iterations.

The number of Gaussian densities used in the instrument recognition studies mentioned above are 1 (Eggink and Brown, 2004), 3 (Zlatintsi and Maragos, 2011), 8 (Essid *et al.*, 2006a; Joder *et al.*, 2009), 32 (Heittola *et al.*, 2009), 40 (Jinchahitra, 2004) and 120 (Eggink and Brown, 2003).

GMMs have also been used for the modelling of timbre features, where they are used to compute song-level similarity; this is different from the use of GMMs for classification (Aucouturier and Pachet, 2002 and 2004).

3.6.5 Decision trees

The goal of a classification tree is to segment data in a top-down construction, through recursive partitioning, into subgroups that are as homogeneous as possible with respect to the response variable, with the first split choosing the most informative feature that can best differentiate the dataset. The process of splitting into nodes is continued until no further splits are possible, or until some stopping criterion is satisfied. To prevent the number of splits that should be considered from becoming unmanageable, a common restriction is to allow only binary splits. Splits are evaluated by considering the reduction in diversity achieved by each possible split. A very popular method of measuring the diversity is the Gini index, although other measures of diversity can also be used. Tree size is controlled by pre- or post-pruning.

Some advantages of trees are that they are easy to interpret, can detect interactions between variables and can automatically select input variables. They also scale well, and can use categorical or numerical variables.

Trees are implemented in an instrument recognition context by Röver *et al.* (2005), Zhang and Ras (2007a) and Jiang *et al.* (2010). The most popular implementation appears to be C4.5 (Quinlan, 1993) (implemented by Zhang and Ras, 2007a and Jiang *et al.*, 2010 through J48, an open source Java implementation of C4.5). Jiang *et al.* (2009a) propose a multi-label extension to a decision tree, while Little and Pardo (2008) use Extra Trees (Geurts *et al.*, 2006), an ensemble method using randomised decision trees, thus building a tree completely independent of the training data.

Random forests are an ensemble learning version of decision trees, first implemented by Breiman (2001). Trees are constructed in such a way as to minimise bias and correlation between individual trees. This is achieved through random bootstrap sampling of the training dataset; sampling is done with replacement, which means that

a proportion of the training data is not used in the bootstrap sample for any given tree, and leads to the use of out-of-bag samples for error estimation (Hastie *et al.*, 2009). Classification is made by a vote among all of the trees grown.

An important advantage of random forests is that they give an estimate of the importance of different features for the final prediction. In addition, very little tuning is required, and random forests are not prone to overfitting.

Random forests have been used for instrument recognition by Kursá *et al.* (2009 and 2010b), Kubera *et al.* (2010 and 2013) and Spyromitros-Xioufis *et al.* (2011). Kursá *et al.* (2009) make a strong argument for the use of random forests over SVMs for instrument recognition. They argue that SVMs are not well-positioned to deal with a sparse distribution of feature values which cannot be mapped onto large continuous intervals (as is the case for instrument data), whereas random forests can handle such data fairly easily. They also found that random forests far outperformed SVMs in their study. In Kursá *et al.* (2010b) they further extend their work on random forests, by extending their study to the polyphonic case. Again they achieved good results with random forests. Kubera *et al.* (2013) and Spyromitros-Xioufis *et al.* (2011) use random forests in a multi-label context. In the latter case, random forests were combined with Asymmetric Bagging – so instead of taking a bootstrap sample from the whole training set, it is executed only on examples of the majority class. They obtained very promising results.

3.6.6 Other classifiers

Other classifiers which are encountered in instrument recognition studies are neural networks (Kostek, 2004; Simmermacher *et al.*, 2006; Benetos *et al.*, 2007; Loughran *et al.*, 2008a; Hamel *et al.*, 2009; Jiang *et al.*, 2009b; Taweewat and Wutiwiwatchai, 2013), linear discriminant analysis (LDA) (Livshin and Rodet, 2004; Röver *et al.*, 2005; Kitahara *et al.*, 2007b; Zhang *et al.*, 2007), naive Bayes classifiers (Röver *et al.*, 2005; Zhang and Ras, 2007a; Deng *et al.*, 2008; Jiang *et al.*, 2010), and non-negative matrix factorisation (NMF) (Benetos *et al.*, 2006; Benetos *et al.*, 2007). Details of all of these techniques can be found in Hastie *et al.* (2009).

3.6.7 Boosting

Many instrument recognition approaches – such as those described above – obtain a subset of optimal features through some form of feature selection, and then apply one or more classifiers to this subset of features. If more than one classifier is applied, results from different classifiers are compared to find the best performing classifier. Multiple classifiers could be implemented through some form of boosting, but straightforward boosting algorithms such as AdaBoost (Freund and Schapire, 1997) are not really suitable in an instrument recognition context (Jiang *et al.*, 2010; Wu *et al.*, 2011) where training data could be noisy and the number of data samples per class could be small. Wu *et al.* (2011) therefore propose a new boosting algorithm based on probabilistic (instead of deterministic) decision making and obtained good results. Jiang *et al.* (2010) take a different approach, where they train different classifiers on different feature sets instead of different data samples. They find that certain features, or groups of features, perform well with different classifiers; for instance, they find that harmonic peaks fit decision trees better than kNN. They also conclude that kNN is more sensitive to feature selection than decision trees, which may be because decision trees perform automatic feature selection. To extend their idea of different classifiers for different feature sets, they also train different classifiers on different feature sets at different levels of a hierarchical classification system.

3.6.8 Multi-label methods

Although multi-label techniques (see Chapter 4) have been used with music data for a while (Wieczorkowska *et al.*, 2006; Trohidis *et al.*, 2008), it is only in the last few years that more and more authors have been using multi-label classification methods for polyphonic instrument recognition problems. This is a natural fit to the problem, since it means that classification can be done using features extracted directly from the mixed signal, without any prior sound source separation or similar pre-processing. Multi-label classification methods will be discussed in detail in Chapter 4; here we will only present a brief overview of instrument recognition studies that have used multi-label classification methods.

Jiang *et al.* (2009a) derive multi-label versions of kNN and decision trees to use in their study of polyphonic instrument recognition. They find that it gives better recognition rates than using single-label classifiers based on sound source separation as a first step.

Kubera *et al.* (2013) train a number of binary classifier random forests to deal with the multi-label nature of polyphonic instrument recognition

Another study utilising multi-label learning approaches for polyphonic instrument recognition, is Spyromitros-Xioufis *et al.* (2011). Although their approach is very relevant for our study (since it uses exactly the same dataset; see Chapter 8), it is not an approach that will generalise well, since they utilised a lot of information (such as prior probabilities) specific to this dataset (their approach was developed in a competition context). Although the specifics of their approach will therefore not necessarily fare well on other datasets, it does illustrate the usefulness of multi-label learning methods in an instrument recognition context. (The two best-ranking entries in the competition both used multi-label learning methods; details of the competition can be found in Chapter 8). Another point raised in their paper which is worth noting, is that they found that it is better to use mixture data (that is, a mixed polyphonic signal) for training rather than single instrument data if the aim is to classify instruments from mixtures. The same finding was made by Kubera *et al.* (2010), although in their case it did not extend through to all metrics (for details on multi-label metrics, see Chapter 4).

While some authors do not use multi-label methods to solve the polyphonic instrument recognition problem, there has been an increasing tendency to use multi-label evaluation measures such as Precision and Recall to report on model accuracies. These measures will be discussed in detail in Chapter 4; some of the authors who have used multi-label metrics in their studies are Hamel *et al.* (2009) and Fuhrmann and Herrera (2010).

The use of multi-label classification methods for instrument recognition remains fairly limited so far, maybe in part due to the fact that multi-label classification methods have only recently started coming to the foreground. Although it has been used with

success in an instrument recognition context (as was discussed above), Fu *et al.* (2011) state: “Whether multi-label learning can be used to improve the performance of instrument recognition is still an open question though.”

3.7 Previous work

Despite the fact that instrument recognition is a relatively recent field, the body of published work is already substantial. Barbedo (2011) gives a very good and extensive literature review and also neatly summarises studies in tables detailing classifier and database used, number of instruments and number of features. The aim of this section is therefore not to repeat what he has already done; instead, we will focus on giving an overview of the more important studies in the field.

Herrera *et al.* (2003) provide an exhaustive review of the early work in instrument recognition (all monophonic), and it is a study that is very often referenced. Not only do they provide extensive detail on the different features and feature types used in different studies, but they also list the different classifiers that have been used in previous studies in the field.

In Herrera *et al.* (2006), a further review of instrument recognition is provided, this time including some of the earlier work done on polyphonic recognition. They give a very comprehensive overview of instrument recognition, touching on everything from features to classifiers and everything in between, and their article is probably the best starting point for introductory reading into the field. They conclude by identifying a number of open issues in the field, such as the need for reference test collections and the need for systems able to cope with realistic polyphonic signals.

As mentioned before, one of the main complexities of instrument recognition in polyphonic music is the phenomenon of overlapping harmonics. Overlapping harmonics occur when more than one instrument is playing simultaneously, and their harmonics overlap and interfere with each other, making the acoustic features different from monophonic ones. One of the earliest attempts at polyphonic instrument recognition, was Eggink and Brown (2003). They follow a “missing

feature” approach, in which they exclude frequency regions that are dominated by energy from an interfering tones; in other words, they exclude areas with confusing or overlapping harmonics from their classification process. Consequently they only employ spectral features, as cepstral features do not fit naturally into the missing feature approach. They obtained promising results for duet recordings from commercially available CDs. Kitahara *et al.* (2007b) attempt to overcome the problem of overlapping partials by weighting features based on how much they are affected by these overlaps. Their basic idea is to look at the ratio of within-class variance to between-class variance for each feature in the training dataset, their reasoning being that large overlaps will lead to large variation. They then weight features accordingly in order to minimise the effect of the overlapping harmonics. Another attempt to address overlapping harmonics is presented by Barbedo and Tzanetakis (2011). They propose finding regions in the time and / or frequency domains where one given instrument appears to be isolated. They identify such isolated harmonics and extract features from them, which is then used for instrument identification. A necessary pre-condition for the success of their method is that at least one isolated harmonic needs to exist for each instrument somewhere in the signal under consideration. They take a pairwise approach, where the classification of each isolated harmonic is performed for every possible pair of instruments, and summarise results to provide an overall estimate of the instruments present in the audio signal. Their results look promising, although a drawback of their approach is that it is dependent on note onset detection, pitch estimation and estimation of the number of instruments as a first step. A pairwise classification strategy was also adopted by Essid *et al.* (2006c).

In polyphonic studies, the number of possible instrument combinations can be substantial. For example, a database of 10 possible instruments, playing together in orchestrations ranging from solos to quartets (which is a fairly modest number of instruments and possible orchestrations by the standards of most music databases), already yields 385 possible instrument combinations – clearly not a realistic number of categories to expect any classification system to be able to deal with. A logical first step therefore seems to be attempting to reduce the number of possible instrument combinations. Some studies have tried to achieve this by using a priori knowledge of for example the genre. Essid *et al.* (2006b) focus on jazz music and rely on the idea

that certain combinations of instruments will be highly unlikely in the jazz genre. For example in jazz music, pieces involving piano, double bass and drums are much more likely to occur than for example, oboe and bassoon duets. They implement their ideas through a hierarchical taxonomy.

Pei and Hsu (2009) attempt to identify the whole instrument set in polyphonic music, but also attempt to determine whether each instrument is dominant at a particular moment in time or not. They do this without F0-estimation, and also do not perform prior sound source separation. Instead, they use fuzzy clustering in conjunction with a beat-tracking algorithm. In brief, their process works as follows. First, they extract MPEG-7 and MFCC features, as well as beat data using BeatRoot (an algorithm to estimate the beginning and ending time of each beat) (Dixon, 2007). They then average frames inside the same beat interval, and apply a fuzzy clustering algorithm (Pedrycz and Gomide, 2007) to these integrated features, with the number of clusters equal to the number of instruments. Lastly a modified SVM classifier is used to allocate an instrument label to every cluster. A major drawback of their method is that the correct number of instruments has to be manually inputted into the system.

Jincahitra (2004) uses independent subspace analysis (ISA) to perform a form of separation on a polyphonic instrument mix. Although this method does not provide actual sound source separation, it does approximate a decomposition of the mixture into statistically independent components, which are intuitive on a physiological level. The results, however, are fairly disappointing.

Kitahara *et al.* (2006, 2007a) attempt to avoid onset detection and F0-estimation as a prerequisite for instrument recognition and propose a visual technique called *instrogram*, which visualises the probability that the sound of a specific instrument exists at a specific time and frequency. Although they found instrograms to be useful in aiding instrument recognition, there does not seem to have been any follow-up research published relating to this technique.

Eggink and Brown (2004) focus on predominant instrument recognition in polyphonic music, specifically accompanied sonatas and concertos where there is a solo instrument accompanied by a keyboard instrument or full orchestra. Their work

is based on the premise that it should be possible to extract the most prominent F0 and its corresponding harmonics from a polyphonic audio signal, and that these will most often belong to the solo (or predominant) instrument. Their approach does not rely on features such as MFCCs; instead, they extract spectral peaks and determine the most prominent F0. Peaks belonging to the harmonic series of this estimated F0 are then used as input features for a Gaussian classifier; in essence therefore they reduce the polyphonic problem to a monophonic one. Their results are promising, although they concede that the approach has certain shortcomings – for example, the piano is often the solo instrument in a concerto and is a polyphonic instrument, which means that it is not possible to find a single dominant F0.

Wieczorkowska *et al.* (2011) also focus on identifying the dominant instrument playing in a polyphonic mixture, where the instruments are playing a sound of the same pitch. They highlight the importance of constructing an appropriate set of features, since “the results of sound recognition may vary depending on the applied parameterisation”, and they provide extensive descriptions of the feature set they chose to work with. They train their models on isolated sounds as well as on sound mixtures with artificial sounds added at different volume levels, and find that, generally, higher levels of added sounds lead to poorer classification results. They also find that training on isolated sounds only lead to worse results than training on both isolated sounds and mixtures.

Bosch *et al.* (2012) focus on identifying the predominant instrument playing in a polyphonic mixture. They work from the premise that the more instruments present in the polyphonic audio, the more difficult it is to identify the instruments. They therefore use sound source separation as a first step in order to reduce the number of instruments to be taken into account. They consider two different separation scenarios: a simple separation of the audio into left, right, mid and side streams, as well as a more complex source separation. Although they achieve better results (on some metrics) using the complex source separation, this comes at a cost in terms of computational complexity, so they recommend the simple separation as a fast and efficient alternative. They further point out that models incorporating sound source separation as a first step need to take into account the limitations and errors of

separation algorithms, otherwise instrument recognition will not necessarily be enhanced.

Wu *et al.* (2011) take a unique approach in that they do not only consider the harmonic partials of notes, but also account for the non-harmonic attack sound of each note. This is a logical extension to other approaches, since it has been shown (Eronen, 2001) that the attack part of a note is a crucial part of being able to distinguish between different instruments. Their approach outperformed other algorithms based on harmonic modelling alone.

The aim of Fuhrmann and Herrera (2011) is to reduce the computational complexity of the instrument recognition problem by trying to reduce the amount of data fed to classifiers. They feel that in many instrument recognition approaches there is a high level of data redundancy, so they suggest some pre-processing to reduce the amount of data needed for analysis without having to sacrifice recognition accuracy. As such, they present 4 different pre-processing approaches. A definite advantage of their approach is that it operates at track level, enabling efficient labelling of entire pieces of music. It also means more efficient computation – even by reducing the amount of data used by 50% or more through pre-processing, they were able to maintain recognition accuracy.

While most of the above approaches focus on a traditional feature-classifier approach, some researchers have attempted to approach the problem from a completely different angle. Instead of extracting features and following a traditional classification route, they attempt to employ a strategy called *template matching*. Barbedo (2011) defines the goal of template matching as “...to find, for each possible instrument, one or more representations (templates) that are consistently valid despite all the variability between instrument samples”. Earlier studies in this regard were Kashino and Murase (1999) and Yoshii *et al.* (2007). Burred *et al.* (2009) group sinusoidal trajectories based on common onsets and then analyse the amplitude evolution of each group with pre-trained templates. Röver *et al.* (2005) use the Hough-Transformation, which is a pattern recognition technique which is sometimes used to detect specific curves or shapes in digital pictures. They apply it to audio data, since they argue that it may be possible to identify certain instruments by certain oscillation

patterns. Although their approach certainly has merit from a conceptual point of view, and achieves lower misclassification rates than humans, it underperforms other automatic instrument recognition methods and therefore has not been widely implemented.

3.8 Related aspects

We close this chapter with a brief discussion of further important aspects.

3.8.1 Commonly used features

The set of features used for instrument recognition tends to be quite varied between studies. Some researchers design their own features, some use common features such as MPEG-7 features and MFCCs. Other spectral features are also often included. Subsets of these feature sets – or combinations thereof – tend to be used, depending on the feature selection used in the study. Müller *et al.* (2011) list some of the features often used for instrument recognition.

There is some evidence that MFCCs are well suited to instrument recognition (Simmermacher *et al.*, 2006), although other studies are critical of MFCCs for a variety of reasons; for example, they only provide a rough description of the spectral shape (Burred *et al.*, 2010) and they are non-linear in nature, so cannot separate simultaneously occurring phenomena (Morvidone *et al.*, 2010). Loughran *et al.* (2008b) looked at the use of MFCCs for musical instrument identification in order to determine how many coefficients should be used. Their conclusion is that at least 10 MFCCs should be used, and they found that using 4 principal components from the first 15 MFCCs gave the best result. This is slightly different from results of using MFCCs in a speech recognition context, where they report that it has been determined that 8 – 14 MFCCs are sufficient to use, and that 12 are quite often chosen.

Most studies tend to use broadly the same collection of features, and relatively few studies propose the use of new features – although this is one area of research that has

been suggested to improve the accuracy of current instrument recognition methods (Barbedo, 2011). One of the very few recent studies proposing the use of new features for instrument recognition, is Morvidone *et al.* (2010). They propose two new sets of features, namely OverCs and SparCs, to overcome some limitations they feel are inherent in the use of MFCCs. However, they concede that it is still an open question whether these new features will work in a polyphonic setting. Sturm *et al.* (2010) also define mean multiscale MFCCs (MSMFCCs), where they compute MFCCs over multiple time scales, with promising results in instrument identification tasks.

Zlatintsi and Maragos (2011) define multiscale fractal features, motivated by the successful use of such features in speech recognition. However, when compared to MFCCs, these features do not fare well when used on their own, although when combined with MFCCs they yield slightly better results than MFCCs alone.

Whereas most instrument recognition approaches rely on the assumption that features extracted from different frames are statistically independent, a number of researchers have been focusing on integration of the mid-term temporal properties of the signal. This approach is referred to as *temporal integration* and it is defined by Joder *et al.* (2009) as “the process of combining several different feature observations in order to make a single decision”. Joder *et al.* (2009) give a good overview of temporal integration techniques with specific application to the instrument recognition problem; other approaches have been suggested by Dubnov and Rodet (1998), Eichner *et al.* (2006), Martins *et al.* (2007) and Tjoa and Liu (2010). While these studies all show that using temporal information can significantly increase performance, it comes at the cost of increased dimensionality.

3.8.2 Feature selection in an instrument recognition context

In an ideal world, features extracted for the purpose of instrument recognition will have a small range of values for each instrument, with no overlap between these ranges. However, this is obviously not the case. Features should therefore be selected so that the overlap between instruments (in the feature space) is minimised, while

keeping in mind that not too many features should be selected in order to avoid the well-known “curse of dimensionality” (Barbedo, 2011). In Chapter 5 we will discuss feature selection in more detail; for now, we will just discuss in brief some of the studies where feature selection was applied in an instrument recognition problem context. One strategy is to choose a large number of features and then use a technique such as principal component analysis (PCA) to reduce the number of dimensions. This approach was taken by authors such as Eggink and Brown (2004), Kaminskyj and Czaszejko (2005) and Loughran *et al.* (2008a).

A fairly comprehensive study on feature selection in an instrument recognition context, is Deng *et al.* (2008). They evaluate different feature schemes, based on human perception, cepstral features and MPEG-7 features, and also use dimension reduction techniques to learn more about the embedded dimensionality for feature selection (which they find to be quite low in most instances). Their study suggests that there is a high degree of redundancy between the different feature schemes, which highlights the importance of feature selection for instrument recognition problems. The authors conclude that additional research into feature extraction and selection is needed, since an optimal and compact feature scheme will enable quicker and more accurate classification. Their feature selection approach is also used by Barbedo and Tzanetakis (2011).

Simmermacher *et al.* (2006) take a correlation-based approach to feature selection for the classification of classical musical instruments. They also find that different feature sets have differing performance for different instrument families; for example, they find that, from single instrument samples, piano could be classified correctly in almost all instances by all of the feature subsets they considered, while MPEG-7 features fared poorly on the brass instrument family. Considering the overall results across all the different feature subsets they considered, they find that MFCCs are the most important features to consider for instrument classification.

Fuhrmann *et al.* (2009) also use correlation-based feature selection to reduce the dimensionality of the problem and therefore lower the computation time required, but do not report on the effect this had on accuracy.

Zhang *et al.* (2007) use discriminant analysis to assist in feature selection, and then use a hierarchical classifier in the classification process. They find that different features have differing degrees of influence on classification performance for different instrument families. Fewer features are also needed for top family-level classification than for lower-level classification.

Essid *et al.* (2006a) propose a new algorithm for feature selection on audio data. They cluster features in such a way that the most redundant ones are put in the same clusters. They then select one feature from each cluster to represent that cluster in the classification process; this representative feature is selected by estimating the weights through Linear Discriminant Analysis.

In Essid *et al.* (2006c), a pairwise approach to feature selection is taken. They use two feature selection techniques (inertia ratio maximisation with feature space projection and genetic algorithms) and show that performing pairwise feature selection and classification not only results in better recognition rates, but can also aid in understanding the differences in timbre between different instruments. They conclude that it is very advantageous to perform pairwise feature selection, since it enables one to look for subsets of features which are best able to discriminate between different pairs of instruments. Although a drawback of such an approach is the large number of possible instrument pairs for databases with a large number of instruments, they argue that a pairwise approach can still be practical in such a case, as long as it is used in conjunction with a hierarchical classification system.

Benetos *et al.* (2007) use a branch-and-bound feature selection approach on a large audio database to reduce the number of features to a more manageable number. They combine this with a novel classifier based on non-negative matrix factorisation and obtain very good results.

Kursa *et al.* (2009) focus on the use of random forests for polyphonic music instrument classification. Their rationale is that random forests work well for high-dimensional feature vectors, which makes it suitable for use with audio data. They use scores obtained from the random forest procedure to assist with feature selection, and use the Boruta Algorithm (Kursa *et al.*, 2010a) to estimate the importance of

features. They find that there is no clear cut-off value between important and non-important features; in fact, they conclude that all MPEG-7 features are important for classification.

Livshin and Rodet (2004) implement their own feature selection method, Gradual Descriptor Elimination (GDE), which uses LDA. They find that recognition rates using their smaller feature set are very close to that of the complete feature set.

3.8.3 Some related applications

Barbedo *et al.* (2009) use a computationally complex technique to estimate the number of sources in a single-channel musical signal, and obtain an average accuracy of almost 80%. Their work could be very relevant in an instrument recognition context, as it could be useful to apply as a first step in an instrument recognition problem, since the number of instruments present is generally not known beforehand. If the number of instruments in a signal could be estimated reliably a priori, the task of training a classifier could be considerably easier.

Fuhrmann and Herrera (2010) use instrument recognition as the first step in calculating the similarity between different tracks.

Benetos and Dixon (2013) developed a multiple-instrument polyphonic music transcription model which includes an instrument recognition component.

3.9 Summary

In this chapter we have summarised the main goal of instrument recognition and discussed some of its inherent challenges. We have also attempted to define the scope of the field and factors which should be considered at the outset. The classifiers often encountered in instrument recognition studies were discussed, with a specific focus on the polyphonic case and multi-label classification.

In discussing the previous work done in the field, the focus was on “statistical” machine learning approaches. The aim was therefore not to evaluate whether feature extraction approaches were appropriate, or to approach the problem on a signal level; instead, the aim was to look at prior approaches at a statistical level. Results between studies were also not directly compared, since studies vary too much in terms of datasets used, number of instruments considered, and so forth.

Lastly, some additional aspects such as commonly used features and feature selection were discussed.

Instrument recognition is clearly a complex field with a very wide scope, and is far from mature. This has implications for the research currently being done in the field; as Barbedo (2011) states: “... instrument recognition research is still at an early stage where coming up with new ideas may be more important than figuring out which algorithm works best”.

CHAPTER 4

Multi-Label Learning

4.1 Introduction

An area of data mining that has been receiving considerable attention recently is the field of multi-label learning. The 2009 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2009) included a workshop and a tutorial on learning from multi-label data, while the 2010 International Conference on Machine Learning (ICML 2010) included a similar workshop. The Machine Learning Journal also recently (July 2012) published a special issue called “Learning from Multi-Label Data” (volume 88, nrs. 1-2).

In a standard *binary classification* problem, each example (or data observation) in a dataset is associated with one of two possible labels; this can be extended to the *multi-class classification* problem, where each example is associated with only one label from a possible set of (more than two) labels. *Multi-label classification* is a further generalisation of the multi-class classification problem, and is concerned with classification problems where each example can be associated with a set of labels instead of just one. A related problem is that of *multi-label ranking*, where instead of

predicting a label or set of labels associated with each example, the goal is to calculate a ranking of all possible labels.

Multi-label learning methods have been applied in fields such as the semantic annotation of images (Yang *et al.*, 2007) and video (Qi *et al.*, 2007), functional genomics (Blockeel *et al.*, 2006), text classification (Yang *et al.*, 2009) and direct marketing (Zhang *et al.*, 2006). Text-related applications are especially prevalent in the multi-label field, and according to Tsoumakas *et al.* (2010), the categorisation of textual data is perhaps the dominant multi-label application.

In the field of music information retrieval specifically, multi-label learning methods have been applied to the problems of music categorisation into emotions (Trohidis *et al.*, 2008 and Trohidis *et al.*, 2011), musical genre classification (Sanden and Zhang, 2011) as well as the problem of instrument recognition (Spyromitros-Xioufis *et al.*, 2011).

In this chapter, the concept of multi-label learning will be formally defined (Section 4.2). In Section 4.3 we will introduce the different multi-label learning methods, and then discuss each of these categories in more detail in Sections 4.4 to 4.6, also explaining the different algorithms in each of these categories. Evaluation measures suitable for use in a multi-label context will be discussed in Section 4.7 and some other multi-label statistics in Section 4.8. The chapter concludes with a look at some multi-label software and benchmark datasets in Sections 4.9 and 4.10.

4.2 Formal definition and notation

Let $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_K\}$ be the set of all possible labels in a multi-label learning task; in other words, each entity in a dataset can be associated with a subset of labels from Λ instead of only a single label.

Let the training data be of the form $\{(x_i, y_i), i = 1, 2, \dots, N\}$, where $x_i: p \times 1$ contains observations on p classification features X_1, X_2, \dots, X_p . Depending on the required formulation of the data, we can either have $y_i \subseteq \Lambda$ indicating the set of labels for

entity i , $i = 1, 2, \dots, N$, or we can consider $\mathbf{y}_i: K \times 1$ as a vector of zeroes and ones indicating the labels assigned to entity i , $i = 1, 2, \dots, N$. If \mathbf{y}_i is defined as a vector of zeroes and ones as in the latter case, the training data can be summarised in matrix form as an $N \times (p + K)$ matrix, viz. $[\mathbf{X} \mathbf{Y}]$, with $\mathbf{X}: N \times p$ containing the observations of X_1, X_2, \dots, X_p and $\mathbf{Y}: N \times K$ the indicator row vectors describing the label subsets assigned to the different entities.

4.3 Categorisation of multi-label methods

The first comprehensive overview of multi-label learning methods was presented by Tsoumakas and Katakis (2007). They divide multi-label learning methods into two categories, namely problem transformation methods and algorithm adaptation methods.

Problem transformation methods transform the multi-label data in some way, so that the problem may be approached as one or more single-label classification problems. These methods are therefore algorithm independent, since any one of a number of traditional single-label classification algorithms can be used after the data has been transformed. The most widely used problem transformation methods are binary relevance (BR), label powerset (LP), classifier chains (CC) and calibrated label ranking (CLR). Two additional variants of the LP method are pruned problem transformation (PPT) and hierarchy of multilabel classifiers (HOMER).

Algorithm adaptation methods provide extensions to some existing single-label classification methods to make them suitable for use with multi-label data. Examples of these are ML-kNN (a multi-label extension of k-Nearest Neighbours), ML-C4.5 (an extension of the C4.5 decision tree algorithm) and Predictive Clustering Trees (PCT).

A more recent overview of multi-label learning was presented by Madjarov *et al.* (2012), and here they extend the two-tier categorisation to include a third category of multi-label learning methods, namely *ensemble methods*. These methods use ensembles of classifiers to make predictions on multi-label data, and the classifiers

used can be either problem transformation or algorithm adaptation methods. The most widely used ensemble method is probably RAKEL (RANdom k-labELsets); others are ensembles of classifier chains (ECC), ensembles of pruned sets (EPS) and random forest extensions to ML-C4.5 and PCT.

We will now proceed with a more detailed discussion of the methods referred to above.

4.4 Problem transformation methods

4.4.1 Binary relevance

Although the **binary relevance (BR)** problem transformation method is extremely straightforward, it remains one of the most popular methods for multi-label learning. In the BR transformation, the original multi-label dataset is split into K datasets, corresponding to the K labels, for each of which a binary classifier is learned. In other words, one classifier is learned for each label; all examples with that particular label is labelled positive and all the rest are labelled negative. In this way the K -label multi-label classification problem is transformed into K binary classification problems. The classifier thus predicts separately whether any label is relevant for a particular entity or not. To illustrate, consider a dataset consisting of five data points together with their corresponding labels as below. For this small-scale illustrative example, $K = 4$.

Data point	Labels
x_1	$\{\lambda_1, \lambda_4\}$
x_2	$\{\lambda_3, \lambda_4\}$
x_3	$\{\lambda_1\}$
x_4	$\{\lambda_2, \lambda_3, \lambda_4\}$
x_5	$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$

The binary relevance transformation will transform this dataset into four separate datasets as follows (where $\neg\lambda_k$ means label λ_k is not present in the dataset):

Data point	Label						
x_1	λ_1	x_1	$\neg\lambda_2$	x_1	$\neg\lambda_3$	x_1	λ_4
x_2	$\neg\lambda_1$	x_2	$\neg\lambda_2$	x_2	λ_3	x_2	λ_4
x_3	λ_1	x_3	$\neg\lambda_2$	x_3	$\neg\lambda_3$	x_3	$\neg\lambda_4$
x_4	$\neg\lambda_1$	x_4	λ_2	x_4	λ_3	x_4	λ_4
x_5	λ_1	x_5	λ_2	x_5	λ_3	x_5	λ_4

Formally, for a given label set $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_K\}$ and a multi-label training dataset of the form $\{(x_i, y_i), i = 1, 2, \dots, N\}$ and $y_i \subseteq \Lambda$, K separate datasets D_{λ_j} , $j = 1, 2, \dots, K$ are constructed where each D_{λ_j} contains all of the entities x_i from the original dataset, but now with $y_i = 1$ if label λ_j is present and $y_i = -1$ if it is not. A binary classifier is then constructed for each dataset D_{λ_j} .

In assigning labels to new cases, the output is given as the union of all labels that were positively predicted by the K binary classifiers.

Advantages of the BR method are its relatively low computational complexity (it scales linearly with respect to the number of distinct labels K), the fact that it is simple to implement and also fairly intuitive. Also, any one of the many well-developed binary classifiers can of course be used. Its major disadvantage is the fact that it assumes labels to be independent and therefore does not take label correlations into account. It can also encounter problems because of imbalanced datasets when for some of the binary datasets there are many more negative than positive examples. Some workarounds have been proposed to deal with these disadvantages of BR, and these will now be discussed briefly.

An approach detailed by Tsoumakas *et al.* (2009) to incorporate label dependencies into the BR framework, is the **2BR** method. This method learns a second level of K binary classifiers after the first round of training (again one for each label), with the output from the first level of binary classifiers taken as input for the second level of

classifiers. In other words, the original dataset is extended by K additional features containing the predictions for the training x_i from the first level of binary classifiers. 2BR can therefore be considered as a form of *stacking* (see for example Hastie *et al.*, 2009, pp. 288-290). For the classification of a new instance, the first round of classifiers is used and the output of this is appended to the original features to form a new appended example. This appended example is then classified using the second round of classifiers.

Cherman *et al.* (2011 and 2012) introduce the **BR+** method to overcome the limitation of assumed label independence in the BR method. In the BR+ method, K binary classifiers are constructed, one for each label λ_j as in the normal BR method; however, the feature space of these K datasets is augmented with $K - 1$ additional features corresponding to the other labels in the dataset. In other words, each dataset D_{λ_j} is augmented with ω_j binary features where $\omega_j = \Lambda - \{\lambda_j\}$. A set of K binary classifiers is then constructed in the augmented feature space, but this introduces the additional complexity that the unlabeled examples must now also be considered in the augmented feature space, and the values of the additional features are unknown for new cases and therefore need to be estimated. Their solution is to predict these values using the BR method as well, and these predictions are then used in BR+ to complete the augmented feature space for new cases.

Hierarchical BR methods have also been proposed to exploit the underlying label structure; see for example Tsoumakas *et al.* (2010).

Despite the shortcomings of the BR method, it fares fairly well in comparative studies. In the extensive comparative study conducted by Madjarov *et al.* (2012), where they evaluated the performance of 12 different multi-label methods on 11 benchmark datasets, BR comes out third overall in most instances. Given its fairly low complexity and relative computational efficiency combined with relatively good performance compared to other methods, BR – or one of its variants – should be given serious consideration when tackling multi-label problems.

4.4.2 Classifier chains

Another method based on BR but aiming to incorporate label dependencies is the **classifier chain (CC)** method proposed by Read *et al.* (2009b). As in the case of BR, K binary classifiers are constructed, but these classifiers are linked along a chain where each classifier takes into account all prior predictions of the input vectors \mathbf{x}_i of preceding binary classifiers in the chain. That means that a chain C_1, C_2, \dots, C_K of binary classifiers is constructed. Each C_j learns and predicts label λ_j , but the feature space of C_j is augmented by all prior predictions of the input vectors \mathbf{x}_i for labels $\lambda_1, \lambda_2, \dots, \lambda_{j-1}$. In this way label dependencies are taken into account while still retaining the relatively low computational complexity of the BR method. The order of the chain will clearly have an impact on the classification accuracy achieved by the CC method. To overcome this shortcoming, Read *et al.* (2009b) propose the use of **ensembles of classifier chains (ECC)**; this will be discussed in Section 4.6.2.

Read *et al.* (2009b) obtain good results (better than BR) for classifier chains in their study, but they only look at limited evaluation measures. In the more comprehensive comparative study conducted by Madjarov *et al.* (2012), they still find that classifier chains perform well and they recommend it to be used as a benchmark method for multi-label learning. However, in their study CCs are outperformed by BR in the majority of instances.

4.4.3 Calibrated label ranking

Label ranking provides an extension to multi-class classification by not only predicting the most likely label, but also providing a ranking of all labels. The problem with extending the concept of label ranking to a multi-label environment, however, is that a “zero-point” is needed; that is, a split of the ranked labels into relevant and irrelevant labels. **Calibrated label ranking (CLR)** (Fürnkranz *et al.*, 2008) transforms a multi-label learning problem into a label ranking problem, and also introduces such a “zero-point” by the introduction of an additional neutral label to the original set of labels.

CLR takes as its starting point a method known as **Ranking by Pairwise Comparison (RPC)** (Hüllermeier *et al.*, 2008). The multi-label dataset is transformed into $\frac{K(K-1)}{2}$ binary label datasets, one for each distinct pair of labels λ_i, λ_j from $\lambda_1, \lambda_2, \dots, \lambda_K$, $1 \leq i < j \leq K$. These datasets only contain cases which contain at least one of the two corresponding labels, but not both. A binary classifier is then trained to discriminate between the 2 labels. If the RPC transformation is applied to the illustrative dataset used in Section 4.4.1, the following six datasets are obtained, and a binary classifier is constructed for each of these datasets:

Data point	Label
x_1	$\lambda_{1,-2}$
x_3	$\lambda_{1,-2}$
x_4	$\lambda_{-1,2}$

Data point	Label
x_1	$\lambda_{1,-3}$
x_2	$\lambda_{-1,3}$
x_3	$\lambda_{1,-3}$
x_4	$\lambda_{-1,3}$

Data point	Label
x_2	$\lambda_{-1,4}$
x_3	$\lambda_{1,-4}$
x_4	$\lambda_{-1,4}$

Data point	Label
x_2	$\lambda_{-2,3}$

Data point	Label
x_1	$\lambda_{-2,4}$
x_2	$\lambda_{-2,4}$

Data point	Label
x_1	$\lambda_{-3,4}$

To classify a new instance, a ranking is obtained by counting the votes received by each label for each binary classifier constructed. However, some thresholding function should still be specified to split the labels into two subsets of relevant and irrelevant labels.

CLR extends RPC to a multi-label framework by adding a virtual label λ_0 for calibration purposes. This “neutral” label acts as the split-point between relevant and irrelevant labels and is assumed to be preferred over all irrelevant labels, while relevant labels are preferred over the virtual label. Binary classifiers are trained to discriminate between the virtual label and each of the other labels. In this sense, CLR can be seen as a combination of RPC and BR. For the illustrative dataset considered in Section 4.4.1, the datasets that will be constructed when a CLR transformation is applied will therefore be the same as those constructed by RPC (as discussed above) together with

those constructed by the BR method. To predict a new instance, the ranking over $K + 1$ labels is therefore obtained.

While CLR uses a majority voting scheme, a more efficient voting strategy is proposed by Mencia *et al.* (2010). The authors use a multi-label adaptation of a Quick Weighted voting method introduced by Park and Fürnkranz (2007) (referred to as QWML by Madjarov *et al.*, 2012), which in effect stops the computation of rankings when the separation of labels into relevant and irrelevant subsets has already been determined. This approach is especially efficient in the case of a large number of possible labels, i.e. for problems with a large K .

Experimental studies by Fürnkranz *et al.* (2008) and Fürnkranz and Hüllermeier (2010) show that CLR outperforms the binary relevance method; however, CLR and QWML did not perform consistently well in the study by Madjarov *et al.* (2012).

4.4.4 Label powerset

The **label powerset (LP)** method transforms a multi-label dataset into single-label datasets by treating each unique set of labels as a distinct class in a multi-class single-label problem. Using the small-scale illustrative dataset from Section 4.4.1 as an example yet again, the dataset resulting from an LP transformation would be:

Data point	Labels
x_1	$\lambda_{1,4}$
x_2	$\lambda_{3,4}$
x_3	λ_1
x_4	$\lambda_{2,3,4}$
x_5	$\lambda_{1,2,3,4}$

Although – unlike the BR method – LP takes label dependencies into account, it means that there is a potentially huge number of possible classes to consider – the number of distinct label sets is upper bounded by $\min\{N, 2^K - 1\}$ – which has a profound effect on computational complexity. There might also be limited training

examples for many classes, and unseen label combinations cannot be predicted by the LP method.

To address these shortcomings, Read (2008) proposes **pruned problem transformation (PPT)**, in which only the distinct label sets which occur more than a predefined number of times φ ($\varphi > 0$) are included in the analysis. Entities with label sets occurring fewer than φ times can be discarded, with training then taking place on the pruned datasets. Alternatively, to avoid the inevitable information loss when discarding a number of examples, the label sets occurring less than φ times can be split into disjoint subsets where these *subsets* occur at least φ times; Read (2008) refers to this approach as PPT-n (with the n standing for “no information loss”). The pruning value φ must be specified by the user, with larger values of φ implying more pruning. In Read (2008) values ranging from 1 to 15 are evaluated, and he finds that in nearly all cases the ideal value ranges from 1 to 5.

Tsoumakas *et al.* (2008) introduce a hierarchical variant of LP, which they call **Hierarchy Of Multilabel classifierS (HOMER)**. The aim of HOMER is to reduce the computational complexity of the LP method owing to the large number of possible label combinations, and HOMER works especially well for datasets with a large number of labels K . It transforms the dataset into a tree-shaped hierarchy, with each node containing a much smaller subset of labels $\Lambda_n \subseteq \Lambda$. The tree consists of K leaves, each containing a different single label λ_j . Every internal node consists of the union of label sets of its children, with the root node containing all labels. A classifier is trained for each node in the tree (except for the leaves), which means that there is a number of simpler multi-label classification tasks. An important issue which needs to be decided is how to allocate labels to each node. Tsoumakas *et al.* (2008) argue that labels should be evenly distributed to subsets in such a way that labels in the same subset are as similar as possible; this is equivalent to a balanced clustering approach, and to this extent they introduce an approach called balanced k-means. However, HOMER can operate with any balanced clustering algorithm. HOMER fares well in initial empirical studies by Tsoumakas *et al.* (2008), and also in the comparative study by Madjarov *et al.* (2012) – in fact, HOMER is one of their 4 recommended benchmark methods for multi-label learning.

4.5 Algorithm adaptation methods

4.5.1 Multi-label kNN

A number of multi-label variations of k-Nearest Neighbours (kNN) have been proposed. The most widely used of these seems to be the approach by Zhang and Zhou (2007), which will be referred to as **Multi-Label k-Nearest Neighbours (ML-kNN)**.

In ML-kNN classification the first step is to calculate the nearest neighbours of the case to be classified – exactly as would be done in a single-label classification problem. Based on prior and posterior probabilities which are estimated using the frequency of each label among the nearest neighbours, the maximum a posteriori principle is used to then determine the label set of an unseen case.

Mathematically, for an unseen case \mathbf{x} , with $\mathcal{N}(\mathbf{x})$ its set of nearest neighbours, ML-kNN calculates the statistic C_j , which records the number of \mathbf{x} 's nearest neighbours with label y_j . The predicted label set is determined by

$$Y = \left\{ y_j \mid \frac{P(H_j|C_j)}{P(\neg H_j|C_j)} > 1, \quad 1 \leq j \leq q \right\}$$

where H_j is the event that \mathbf{x} has label y_j , $P(H_j|C_j)$ is the posterior probability that H_j is true under the condition that \mathbf{x} has exactly C_j neighbours with label y_j and conversely $P(\neg H_j|C_j)$ is the probability that H_j is not true given the same condition. To calculate Y , prior probabilities and likelihoods need to be computed; details of how to accomplish this can be found in Zhang and Zhou (2007).

Spyromitros-Xioufis *et al.* (2008) propose a combination of binary relevance and kNN which they call **BRkNN**. Although this is conceptually equivalent to using BR in conjunction with kNN, it has the advantage of doing it K times faster since BRkNN makes independent predictions for each label after searching just once for the nearest neighbours.

Advantages of multi-label kNN algorithms are that the time complexity scales linearly with respect to the number of labels K , and that the computational complexity is limited to the calculation of the nearest neighbours (which does not depend on K) (Spyromitros-Xioufis *et al.*, 2008). However, a disadvantage associated with lazy learning methods such as kNN is the amount of memory required to store the entire training dataset.

Although the simplicity – and ease of implementation – of ML-kNN is appealing, it does not perform well in the Madjarov *et al.* (2012) comparative study. While empirical work by Spyromitros-Xioufis *et al.* (2008) suggests that their BRkNN approach outperformed ML-kNN, it has not been extensively tested or implemented in other work.

4.5.2 Multi-label C4.5

Clare and King (2001) adapt the C4.5 decision tree algorithm (Section 3.6.5) for use with multi-label data. They adapt the tree structure by allowing multiple labels at the leaves of the tree, and also modify the entropy formula used in calculating information gain when deciding how to grow the tree. For single-label data, the entropy formula used is given by:

$$Entropy(S) = - \sum_{i=1}^K p(c_i) \log p(c_i)$$

where $p(c_i)$ is the probability of belonging to class c_i and S is the set of training examples under consideration.

The adjusted entropy formula for multi-label data assumes independence among labels and is given by:

$$Entropy(M) = - \sum_{i=1}^K ((p(c_i) \log p(c_i)) + (q(c_i) \log q(c_i)))$$

where M is the set of (multi-label) training examples under consideration, $p(c_i)$ is the probability of belonging to class c_i and $q(c_i)$ is the probability of not being a member of class c_i ; that is, $q(c_i) = 1 - p(c_i)$. The adjusted entropy therefore sums the

entropies for each individual class label, weighted in the sense that if an item belongs to two classes then it is counted twice.

4.5.3 Predictive Clustering Trees

Blockeel *et al.* (1998) introduced the concept of **Predictive Clustering Trees (PCTs)**. Clustering trees can be viewed as a hierarchy of clusters; in other words, a clustering tree is a decision tree where the leaves do not represent classes, but where each node and leaf corresponds to a cluster. PCTs are versatile enough to allow application to a variety of problems, amongst which is multi-label learning.

The main difference between the PCT algorithm and a standard decision tree is that in the case of PCTs, the variance function and the prototype function (which calculates a label for each leaf) can be varied according to the purpose. This allows for several different types of outputs such as discrete or continuous variables, time series or hierarchical classes. For example, in the case of tuples of discrete variables, the sum of the Gini indices (Hastie *et al.*, 2009, p. 309) can be used as variance function. In the case of multi-label learning, the prototype function returns a vector of probabilities that an entity is labelled with a given label (Madjarov *et al.*, 2012).

While PCTs only give average performance in terms of evaluation measures of predictive performance in the comparative study by Madjarov *et al.* (2012), it is the most efficient algorithm in terms of training and testing time among all the algorithms evaluated in their study.

4.5.4 Other algorithm adaptation methods

Some other algorithm adaptation methods that are often encountered in the literature are **AdaBoost.MH** and **AdaBoost.MR** (which are extensions of AdaBoost for multi-label data), back-propagation multi-label learning (**BP-MLL**) and **RankSVM** (which is a ranking approach for multi-label learning based on SVMs). A brief description of

all of these methods is given in Tsoumakas *et al.* (2010) and Madjarov *et al.* (2012); they also give comprehensive references for more details on these methods.

4.6 Ensemble methods

4.6.1 Random k-labelsets

The random k-labelsets method (RAkEL) is based on a label powerset (LP) transformation, but in ensemble form. It was first proposed by Tsoumakas and Vlahavas (2007) as a way of retaining the advantage of LP (that is, taking label correlations into account), but overcoming its disadvantages by working with a more manageable number of combinations of labels and also with an adequate number of examples per label.

The basic idea is to construct an ensemble of m LP classifiers. A k -labelset $Y \subseteq \Lambda$ is defined with cardinality $k = |Y|$. Λ^k is defined to be the set of all distinct k -labelsets in Λ , and $|\Lambda^k| = \binom{K}{k}$. For each $i \in \Lambda^k$ a k -labelset Y_i is randomly selected (without replacement) from Λ^k , and an LP classifier is trained for this set of labels. A new instance is classified by considering all binary classifiers, and calculating the average of the decisions for each label $\lambda_j \in \Lambda$. The final decision for the label is deemed positive if the average decision is greater than a threshold value t .

The threshold value t must be specified by the user, and a value of 0.5 is usual (and intuitive), but RAkEL has been shown to perform well across a range of values of t (Tsoumakas and Vlahavas, 2007). Other values which must be specified by the user are the number of iterations m and the size of the labelsets k . Tsoumakas and Vlahavas (2007) state the acceptable range of m to be values from 1 to $|\Lambda^k|$, and of k to be values from 2 to $K - 1$. The authors further hypothesise that using small k -labelsets in conjunction with an adequate number of iterations m will lead to effective modelling of label correlations; their experimental study provides evidence to support this hypothesis.

The special case $k = 1$ and $m = K$ simply corresponds to an ensemble of BR classifiers, while the special case $k = K$ and $m = 1$ is equivalent to a single-label LP classifier.

Initial empirical studies by Tsoumakas and Vlahavas (2007) show that RAKEL performed better than BR and LP; however, RAKEL generally performs relatively poorly in the comparative study by Madjarov *et al.* (2012).

4.6.2 Ensembles of classifier chains and pruned sets

Ensembles of classifier chains (ECC) (Read *et al.*, 2009b) have classifier chains as their base classifier; it trains an ensemble of m CC classifiers C_1, C_2, \dots, C_m and each C_k is trained with a random chain ordering of labels L and a random subset of the multi-label dataset. Since each C_k is likely to be unique and give different predictions for each label, the predictions are summed by label and a threshold value is used to select the most popular labels.

While Read *et al.* (2009b) find that ECCs perform better than some other ensemble methods, especially for large datasets, their study is limited in scope (in terms of datasets, evaluation measures, etc.). In the larger scale study by Madjarov *et al.* (2012), ECCs do not perform very well – in fact, overall it performs worse than CCs. They reason that it could be due to the fact that CCs is a stable classifier and ensembles therefore cannot improve much over their predictive performance.

Read *et al.* (2008) also introduce **ensembles of pruned sets (EPS)**, in which ensembles are used to reduce the computational complexity of LP, as well as an instance duplication method to reduce the error rate compared to LP and other methods.

4.6.3 Random forests

Random forest extensions for ML-C4.5 (**RFML-C4.5**) (Madjarov *et al.*, 2012) and PCT (**RF-PCT**) (Kocev *et al.*, 2007) are briefly described and evaluated in Madjarov *et al.* (2012). In these extensions, multi-label predictions made by individual base classifiers are combined using some voting scheme.

RF-PCT performs extremely well across different datasets and evaluation measures and is suggested by Madjarov *et al.* (2012) as a benchmark method for multi-label learning. RFML-C4.5 fares less well, and in fact even performs worse than standard ML-C4.5 (i.e. the non-ensemble version). The authors hypothesise that RFML-C4.5 does not perform competitively because it selects feature subsets with a logarithmic size compared to the complete set of features, and since their study looked at datasets with a large number of features, the feature space is under-sampled and some useful information is therefore missed by RFML-C4.5.

4.7 Multi-label evaluation measures

4.7.1 Overview

Multi-label classification models cannot be evaluated in the same way as single-label classification models, since the multi-label setting introduces additional degrees of freedom (Madjarov *et al.*, 2012). Multiple and contrasting measures are therefore required, and to this extent a large number of evaluation measures suitable for multi-label classification has been used in the literature. A good summary and categorisation of many of these measures are given by Tsoumakas and Vlahavas (2007), Tsoumakas *et al.* (2010) and Madjarov *et al.* (2012).

Multi-label evaluation measures can generally be divided into *bipartitions-based* and *rankings-based* measures. Bipartition-based measures can further be divided into example-based and label-based measures.

Bipartition-based measures compare predicted labels to actual labels; example-based measures consider the average differences of predicted and actual labels over all examples, while label-based measures consider the predictive performance for each label separately and then average over all labels. Ranking-based measures compare the predicted ranking of labels to actual labels. Tsoumakas *et al.* (2010) also make use of a *hierarchical loss measure* which takes a possible hierarchical structure of the labels into account.

Some of the bipartition-based measures will be discussed briefly in Sections 4.7.2 and 4.7.3, and some ranking-based ones in Section 4.7.4. Notation used is as set out in Section 4.2, with additionally the set of labels predicted for \mathbf{x}_i denoted by \mathbf{z}_i and the rank predicted for a label λ denoted as $r_i(\lambda)$, where the most relevant label according to the classification method receives rank 1 and the least relevant one rank K . Throughout this discussion we use $|\cdot|$ to denote the cardinality of a vector or a set.

4.7.2 Example-based measures

Hamming Loss is a measure of how many times a label pair is misclassified; in other words, how many times a label not belonging to the subset of correct labels for the example is predicted, or a label belonging to the subset of correct labels is not predicted. Smaller values of Hamming Loss equal better performance, with perfect performance in the case when the Hamming Loss is equal to 0. It is defined as

$$\text{Hamming Loss} = \frac{1}{N} \sum_{i=1}^N \frac{|\mathbf{y}_i \Delta \mathbf{z}_i|}{K}$$

where Δ is the symmetric difference between two sets, in this case the actual label vector \mathbf{y}_i and the predicted label vector \mathbf{z}_i .

Accuracy is defined as

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \frac{|\mathbf{y}_i \cap \mathbf{z}_i|}{|\mathbf{y}_i \cup \mathbf{z}_i|}$$

which is simply the Jaccard similarity coefficient for the subsets \mathbf{y}_i and \mathbf{z}_i averaged over all examples.

Precision calculates the average proportion of predicted labels which are relevant, and is defined by

$$Precision = \frac{1}{N} \sum_{i=1}^N \frac{|\mathbf{y}_i \cap \mathbf{z}_i|}{|\mathbf{z}_i|}$$

while the related measure **Recall**, which calculates the average proportion of relevant labels which are predicted as such, is defined by

$$Recall = \frac{1}{N} \sum_{i=1}^N \frac{|\mathbf{y}_i \cap \mathbf{z}_i|}{|\mathbf{y}_i|}$$

Precision and Recall are commonly encountered in an information retrieval context, where Recall is also known as sensitivity or true positive rate. Precision is also referred to as positive predictive value; a related value is the specificity or true negative rate.

There is an inherent trade-off between Precision and Recall; an increase in one of these measures usually happens at the expense of a decrease in the other. This trade-off is captured by the harmonic mean of Precision and Recall which is called the **F₁ Score**. It is defined as

$$F_1 \text{ Score} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Subset Accuracy (also sometimes referred to as Classification Accuracy) is a very strict measure, since it requires the predicted and actual labels to be an exact match. It is defined as

$$Subset \text{ Accuracy} = \frac{1}{N} \sum_{i=1}^N I(\mathbf{y}_i = \mathbf{z}_i)$$

where $I(true) = 1$ and $I(false) = 0$.

4.7.3 Label-based measures

Any known measure for binary evaluation could be used as a label-based measure in a multi-label classification context by simply averaging such a measure over labels. Micro- or macro-averaging operations can be used; micro-averaged measures are averaged over all example/label pairs, while macro-averaged measures are averaged across all labels. These averages are usually calculated for Precision, Recall and their harmonic mean, i.e. the F_1 -score.

For any label λ_j considered as a binary class, let tp_j be the number of true positives, tn_j the number of true negatives, fp_j the number of false positives and fn_j the number of false negatives after application of a multi-label method to a test dataset. In table form:

		TRUE	
		Positive	Negative
PREDICTED	Positive	tp	fp
	Negative	fn	tn

Micro-precision is defined as

$$\text{Micro-precision} = \frac{\sum_{j=1}^K tp_j}{\sum_{j=1}^K tp_j + \sum_{j=1}^K fp_j}$$

and **Macro-precision** as

$$\text{Macro-precision} = \frac{1}{K} \sum_{j=1}^K \frac{tp_j}{tp_j + fp_j}$$

Similarly, **Micro-recall** is defined as

$$\text{Micro-recall} = \frac{\sum_{j=1}^K tp_j}{\sum_{j=1}^K tp_j + \sum_{j=1}^K fn_j}$$

with **Macro-recall** as

$$\text{Macro-recall} = \frac{1}{K} \sum_{j=1}^K \frac{tp_j}{tp_j + fn_j}$$

Macro- and micro-versions of the F_1 -score could also be calculated by considering the Macro- and Micro-precision and Macro- and Micro-recall. Some measures, such as Accuracy, have the same macro- and micro-version. While there are no clear guidelines in the literature as to which measure (micro- or macro-) is preferred in which situation, it should be kept in mind that macro-measures give equal weight to predictions for each label, so might not be as suitable in situations where the label distributions are very uneven. It is, however, useful in situations where the goal is to compare results across different datasets with differing label densities.

4.7.4 Rankings-based measures

Rankings-based measures evaluate the accuracy of a label ranking produced or implied by a multi-label classifier.

One-Error gives an indication of how often the top-ranking predicted label is not in the set of true labels for the example. One-Error can take values between 0 and 1, with a smaller value corresponding to better performance. It is defined as

$$\text{One-Error} = \frac{1}{N} \sum_{i=1}^N \delta(\arg \min_{\lambda \in \Lambda} r_i(\lambda))$$

where $\delta(\lambda) = 1$ if $\lambda \notin \mathbf{y}_i$ and 0 otherwise.

Coverage gives an indication of how far we need to go, on average, down the list of ranked labels in order to cover all the relevant labels of the example. Smaller values of Coverage correspond to better performance. The smallest possible value for Coverage is equal to the label cardinality of the dataset. In the literature, Coverage is usually defined as

$$Coverage = \frac{1}{N} \sum_{i=1}^N \max_{\lambda \in \mathbf{y}_i} r_i(\lambda) - 1$$

However, in our opinion this does not give a true portrayal of Coverage, since it does not take the label cardinality into account. For example, using this definition of Coverage, a Coverage value of 5 in a dataset with only 5 possible labels will surely not be as good as a Coverage value of 5 in a dataset with 10 possible labels. For the purposes of our study, we have therefore redefined Coverage as

$$Coverage = \frac{1}{N} \sum_{i=1}^N \frac{\max_{\lambda \in \mathbf{y}_i} r_i(\lambda)}{|\mathbf{y}_i|} - 1$$

which, in our opinion, gives a better reflection of true Coverage.

The average number of times that irrelevant labels are ranked higher than relevant labels are given by the **Ranking Loss**, which is defined as

$$Ranking Loss = \frac{1}{N} \sum_{i=1}^N \frac{1}{|\mathbf{y}_i| |\bar{\mathbf{y}}_i|} |\{(\lambda_a, \lambda_b) : r_i(\lambda_a) > r_i(\lambda_b), (\lambda_a, \lambda_b) \in \mathbf{y}_i \times \bar{\mathbf{y}}_i\}|$$

where $\bar{\mathbf{y}}_i$ is the complementary set of \mathbf{y}_i with respect to Λ . Smaller values for Ranking Loss equate to better performance.

4.8 Other statistics

The number of distinct labels K in a dataset could influence the performance of different multi-label learning methods, as could the number of labels of each separate entity in a dataset. To provide an indication of “how multi-label” a multi-label dataset is, Tsoumakas *et al.* (2010) introduce the concepts label cardinality and label density.

Label cardinality of a dataset is defined as the average number of labels of entities in a dataset:

$$\text{Label Cardinality} = \frac{1}{N} \sum_{i=1}^N |Y_i|$$

Label density is the average number of labels of entities in a dataset, divided by the total number of distinct labels in the dataset; in other words, label density is the label cardinality divided by K :

$$\text{Label Density} = \frac{1}{N} \sum_{i=1}^N \frac{|Y_i|}{K}$$

4.9 Multi-label software

While multi-label problems approached via the problem transformation method can generally be solved using any existing machine learning or data mining software, there exists a number of specific software packages and implementations for algorithm adaptation methods.

Probably the most widely used software is *Mulan*²⁰, which is an open-source Java library for multi-label learning. It includes implementations of a large number of state-of-the-art multi-label algorithms, such as BR, CLR, HOMER, ML-C4.5, ML-kNN and RAKEL. It also has basic feature selection capabilities and an extensive evaluation framework.

*Meka*²¹ (based on the WEKA Machine Learning Toolkit of the University of Waikato) is another open-source implementation of methods for multi-label learning and includes implementations for methods such as CC, ECC and EPS.

Other available multi-label learning options are Matlab implementations for ML-kNN and BP-MLL, as well as *Clus*²², which is a predictive clustering system that allows for hierarchical multi-label classification.

²⁰ <http://mulan.sourceforge.net> (accessed 4 June 2013)

²¹ <http://meka.sourceforge.net> (accessed 4 June 2013)

²² <http://clus.sourceforge.net> (accessed 4 June 2013)

4.10 Benchmark datasets

There exists a number of benchmark multi-label datasets which are widely used across multi-label studies. These datasets come from different domains and have a range of values (albeit somewhat limited) for the number of labels K as well as differing label cardinalities and densities.

Some of the datasets used most often are presented in the table below, together with statistics such as the size of the dataset (N), number of features (p), number of labels (K), label cardinality and label density.

Table 4.1: Some publicly available multi-label benchmark datasets

Dataset	Domain	N	p	K	Label cardinality	Label density
Bibtex	text	7395	1836	159	2.402	0.015
Bookmarks	text	87856	2150	208	2.028	0.010
corel5k	images	5000	499	374	3.522	0.009
Delicious	text (web)	16105	500	983	19.020	0.019
Emotions	music	593	72	6	1.869	0.311
Enron	text	1702	1001	53	3.378	0.064
genbase	biology	662	1186	27	1.252	0.046
mediamill	video	43907	120	101	4.376	0.043
medical	text	978	1449	45	1.245	0.028
scene	image	2407	294	6	1.074	0.179
tmc2007	text	28596	49060	22	2.158	0.098
yeast	biology	2417	103	14	4.237	0.303

Source references for all of these (and a few other) datasets can be found on the Mulan website²³.

²³ <http://mulan.sourceforge.net/datasets.html> (accessed 4 June 2013)

4.11 Summary

In this chapter we examined the field of multi-label learning. We presented a formal definition of the multi-label classification problem and categorised the different multi-label learning algorithms. We then proceeded with a more detailed presentation of different multi-label learning algorithms.

Since the performance of multi-label classification algorithms cannot be evaluated in the same way as that of single-label classification algorithms, we presented a number of the most-often encountered multi-label evaluation measures, as well as two descriptive statistics for multi-label datasets, namely label cardinality and label density. We concluded this chapter by discussing software packages for implementing multi-label algorithms and listing some publicly available benchmark datasets.

CHAPTER 5

Multi-Label Feature Selection

“Less is more.”

Ludwig Mies van der Rohe, German-American architect and icon of minimalist design

“Don’t use a lot when a little will do.”

Proverb

5.1 Introduction

Feature and / or variable selection have become increasingly important in data mining environments. Guyon and Elisseeff (2003) point out that around the year 1997, having more than 40 features in a dataset was practically unheard of; today, many datasets have hundreds or even thousands of variables or features. This is largely due to the fact that with the advent of computers and the increase in computing capabilities, it has become very easy to gather data, and it has also become easier and cheaper to store data (as was discussed in Chapter 1). Consequently, often more data than is actually required is collected and therefore selection is necessary to filter out unnecessary information.

Datasets with many more features (p) than observations (N) (known in the literature as “wide” datasets) are also becoming more commonplace, especially in areas such as genomics and computational biology. Hastie *et al.* (2009) devote an entire chapter to such $p \gg N$ problems, since these require special approaches due to the fact that

traditional data mining and statistical approaches are not necessarily valid in such feature spaces.

At the outset, it is important to distinguish between the terms “variable” and “feature”. Although the two are often used interchangeably in the literature, there is an important distinction between the two terms when using kernel methods such as SVMs: in such cases, “variables” refer to the original attributes in input space whereas “features” refer to the transformed variables in feature space. In this thesis we will use the term *feature* throughout, unless we specifically want to refer to input variables in the context of kernel methods.

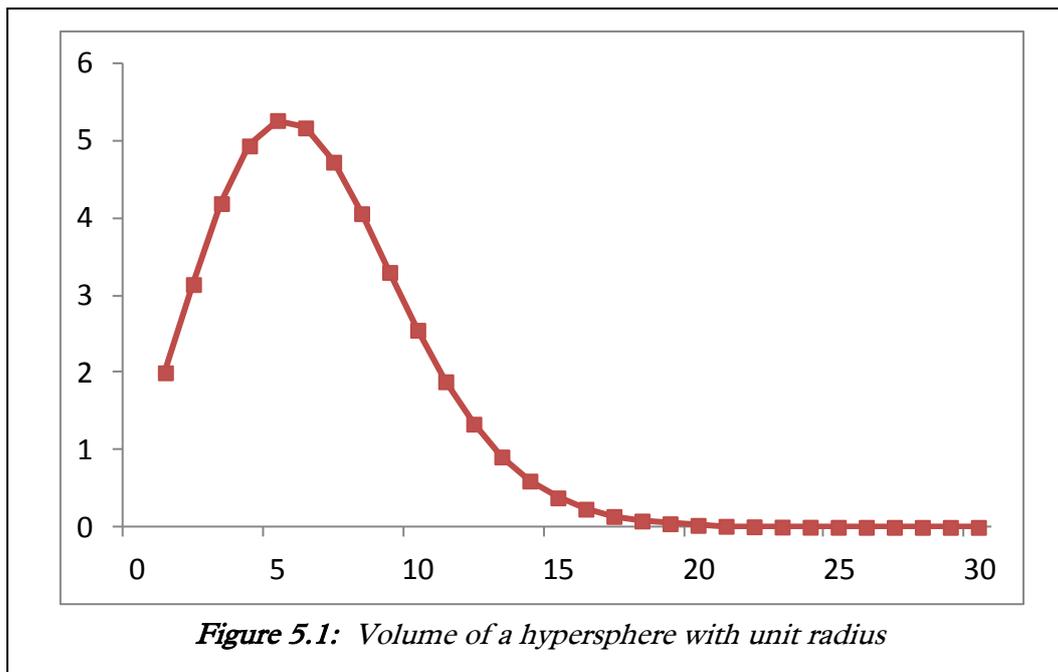
We will start this chapter by describing the aim and benefits of feature selection in Section 5.2, followed by a description of some ways in which the efficacy of feature selection can be evaluated (Section 5.3). A brief summary of general approaches to feature selection will be presented in Section 5.4, while existing multi-label feature selection approaches will be discussed in some detail in Section 5.5. In Section 5.6, we will conclude the chapter by introducing a new technique for multi-label feature selection based on the concept of probe variables used by Tuv *et al.* (2008).

5.2 Aim and benefits of feature selection

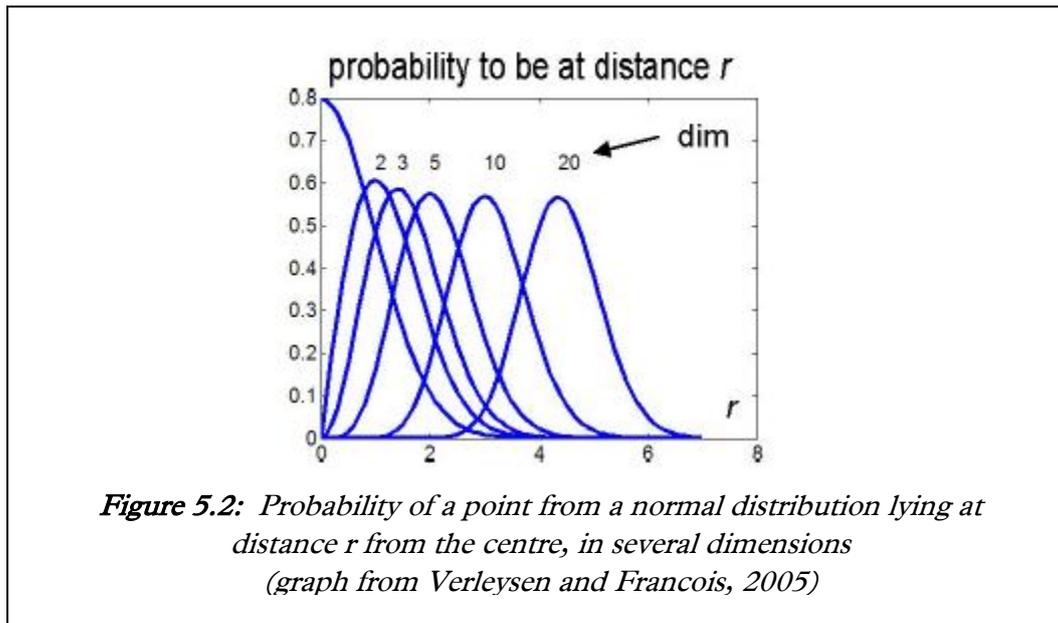
The curse of dimensionality is a well-known concept in statistics and machine learning: every feature in a dataset represents a separate dimension, so a large number of features leads to high-dimensional (or even ultra-high dimensional²⁴) spaces. This implies that there needs to be enough training data to fill the feature space; in other words, the number of training data samples should grow exponentially as the dimension increases. Furthermore, high-dimensional spaces have geometrical properties that are not necessarily intuitive, which can be illustrated by way of the following example. Consider a hypersphere with unit radius, the volume of which is plotted in Figure 5.1 below. The figure shows that as the dimension of the hypersphere increases from 1 to 5 the volume increases as well, but it then starts

²⁴ Fan and Lv (2010) use the term high-dimensional to refer to the general case of growing dimensionality, whereas they reserve the term ultra-high dimensional to refer to the case where dimensionality grows at an exponential (not polynomial) rate as the sample size increases.

decreasing up to the point where it almost reaches 0 when the dimension is greater than 20. A statistical implication of this result is that any procedure based on data in a local spherical environment of a target point will break down in high dimension (e.g. k -nearest neighbours based on the Euclidean metric) (Verleysen and Francois, 2005).



Such counter-intuitive geometrical properties affect the behaviour and performance of learning algorithms as well. Verleysen and Francois (2005) show that such properties of high-dimensional spaces also have an effect on the concentration of norms, meaning that even uniformly distributed data can concentrate in unexpected parts of the feature space, and that in such instances norms do not follow intuitive distributions. To illustrate the concentration-of-norm phenomenon, Verleysen and Francois (2005) use the following example. Consider the normal distribution with a standard deviation of 1. Figure 5.2 shows the probability density functions of finding a point drawn according to a normal distribution at distance r from the centre of that distribution, for several dimensions of space.



In one dimension, the probability density function is monotonically decreasing, while in more than one dimension it has a bell shape, with its position shifting to the right as dimension increases. The graph shows that, in 20 dimensions, the probability that a point will lie within 2 units from the centre is negligible, even though the distribution has a standard deviation of 1. This implies that the distances between all points and the centre of the distribution is concentrated in a very small interval.

The implication of these results is that, when working in high-dimensional spaces, either a strong assumption about the structure of the data needs to be made, and / or the dimension of the data needs to be reduced in some way. Feature selection addresses the latter of these two options.

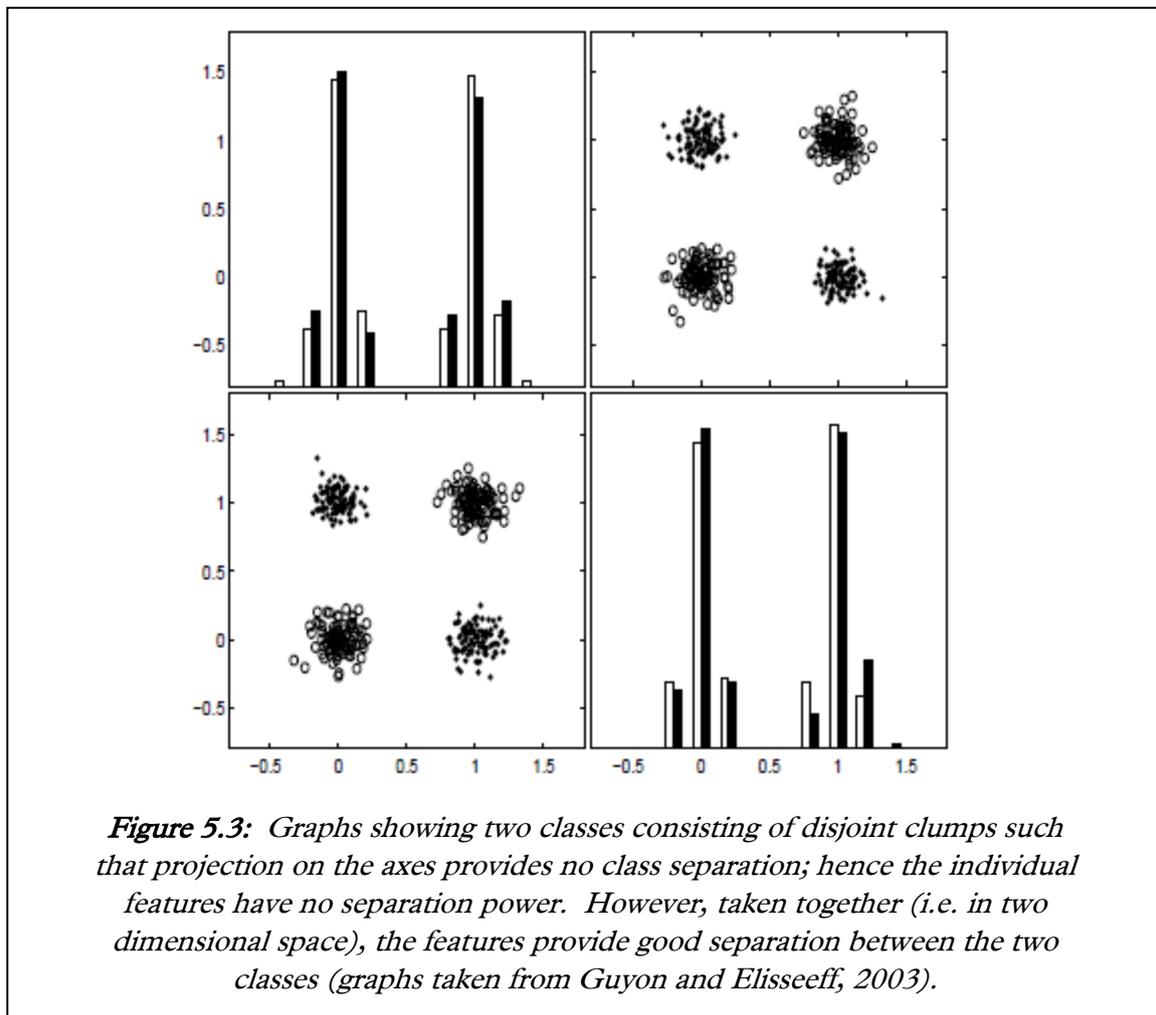
Spolaôr *et al.* (2013) succinctly state the aim of feature selection as “...to find a small number of features that describes the dataset as well as the original set of features does”. Gheyas and Smith (2010) refer to the *principle of parsimony* (or Occam’s razor, as it is more often referred to in popular literature and media): a model with the smallest possible number of features that adequately represents the data is preferred above any other model. Naturally, for this principle to be useful, the meaning of “adequately represents” should be clear.

Feature selection in a supervised learning context reduces the dimensionality of a dataset by removing irrelevant and / or redundant features. *Irrelevant features* are features which carry no information about the task at hand and can therefore often be excluded from the dataset without affecting performance. *Redundant features* on the other hand, are relevant to the problem at hand, but a redundant feature effectively conveys the same information as one or more other features; redundant features can therefore also be removed without affecting performance. *Interacting features* should also be taken into account – these are features which on their own contribute nothing to the prediction task, but when used in conjunction with another feature (or features) they are useful for prediction. Successful feature selection algorithms therefore need to be able to eliminate irrelevant and redundant features from a dataset, while retaining relevant features as well as the right combinations of interacting features.

A major benefit of feature selection is enhanced predictive performance. Including unnecessary features (or noise) adversely impacts on classification performance; in fact, Fan and Fan (2008) show that in high-dimensional feature space, classification using all features can be just as bad as classification by random guessing.

Identifying interacting features is also important for the sake of enhancing predictive performance. Guyon and Elisseeff (2003) demonstrate that a feature which is completely useless when used on its own can provide a significant performance improvement when used in conjunction with other features. They do this by constructing an example based on the XOR (exclusive OR) problem. They draw examples for two classes using four normal distributions placed on the corners of a square at coordinates (0;0), (0;1), (1;0) and (1;1), with class labels attributed to the truth table of the logical XOR function. This example is illustrated in Figure 5.3 below. Consider first the bottom left graph. Here we have data points from four bivariate normal distributions representing two different groups: points in the lower left and upper right groups form one group, and those in the lower right and upper left form another group. In the upper left histogram, the projections of these points onto the X_1 axis are represented, and show that the two groups cannot be distinguished. A similar description and conclusion hold for the two rightmost graphs. This figure therefore shows that the projections on the axes provide no class separation. However, in the two-dimensional space the classes can easily be

separated, demonstrating that two features that are useless by themselves can be useful together.



As a further illustration of the potential benefit of interacting features, consider two features X_1 and X_2 used as classification features in a binary classification problem, with response $Y, Y \in \{-1, 1\}$. Assume X_1 and X_2 both follow a normal distribution, but with X_1 providing good separation between the two classes while X_2 does not (Figure 5.4(a) and Figure 5.4(b)). Furthermore, assume a fairly large positive correlation between X_1 and X_2 . Although X_1 provides fairly good separation between the two classes, in the case of a new observation \mathbf{x} with measurement x_1 lying in the region of overlap between the two classes, X_1 provides no discriminatory power (Figure 5.4(c)), since in this case x_1 could indicate a large value of X_1 from class 1 or a small value of X_1 from class 2. However, since there is a fairly large positive correlation between X_1

and X_2 , if x_2 is a below average value for X_2 it implies that x_1 is also likely to be a below average value for X_1 . This implies that x_1 is more likely to be a small value of X_1 from class 2 than a large value from class 1, and hence we would tend to classify $\mathbf{x} = (x_1, x_2)$ into class 2. Hence, although variable X_2 has no discriminatory power on its own, it can improve classification accuracy through its correlation with variable X_1 .

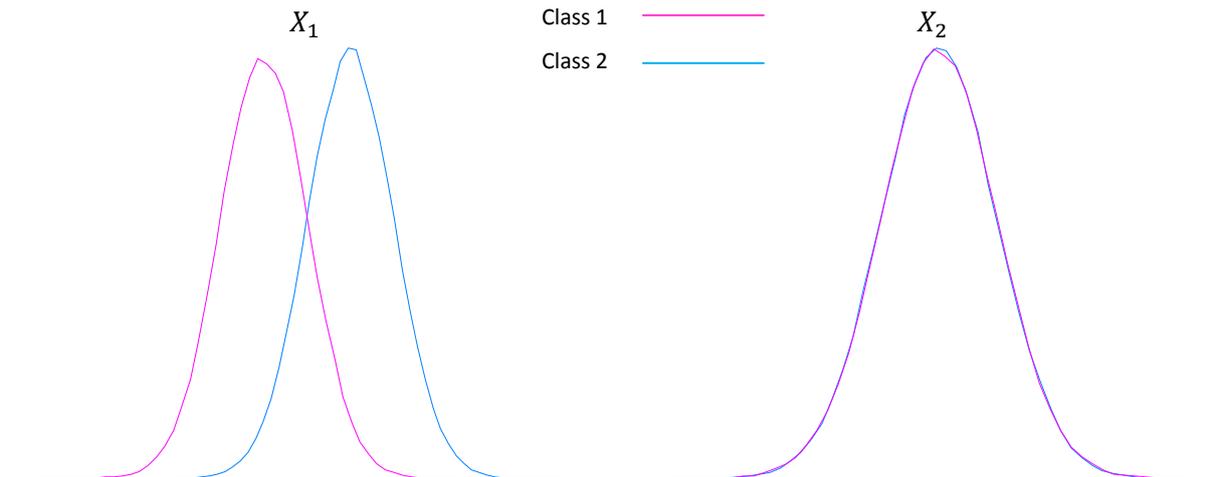


Figure 5.4(a): Feature X_1 provides good separation between the two classes

Figure 5.4(b): Feature X_2 does not discriminate between the two classes

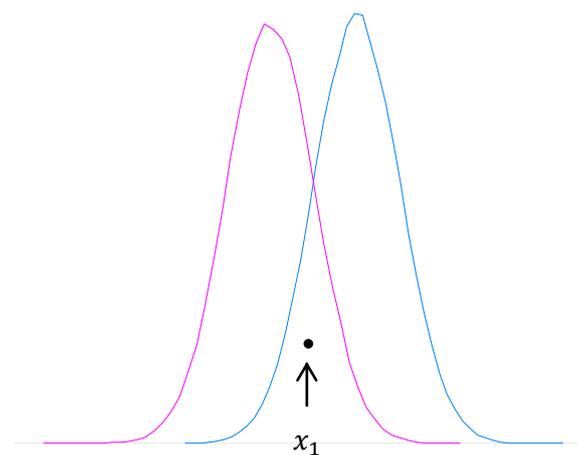


Figure 5.4(c): If new observation x_1 falls in the region of overlap between the two classes, X_1 does not provide good separation. However, knowing that the features are highly correlated can help with determining the class of the new observation.

Another substantial benefit of reducing the dimensionality of a problem is increased efficiency due to decreased computational complexity, which means shorter model

training and prediction times. This can be especially beneficial in cases where complex learning algorithms are employed, and have to be implemented in real time.

A benefit of feature selection which is sometimes overlooked is the fact that it can lead to a better understanding of the data, and of the underlying processes that generated the data: knowing which features are important for discriminative purposes can help with interpretability of the problem. Such an understanding of important features can also be of benefit in circumstances where data collection is difficult and / or expensive.

5.3 Measuring the efficacy of feature selection

Once a subset of features has been selected, this subset needs to be evaluated in some way to determine if the intended objectives of feature selection have been met. There are many possible approaches, and a good overview is presented in Dreyfus and Guyon (2006). However, in brief, some of the ways to measure the efficacy of the selection are:

1. Measure the effect of feature selection on *classification accuracy*. If a subset of features leads to higher classification accuracy than using all the features, feature selection can clearly be considered beneficial. Even if using a subset of features leads to performance similar to that when using a full set of features, in terms of efficiency this means that feature selection is beneficial. It is also important that classification accuracy is evaluated on the training data as well as test data, otherwise the classifier might overfit the training data. Cross-validation is often applied to split a dataset into training and test parts, and in this case a decision needs to be made whether to perform feature selection inside or outside of the cross-validation loop. Refaeilzadeh *et al.* (2007) perform an extensive evaluation of the advantages and disadvantages of each, and give recommendations on which approach to follow depending on the end goal of the study. In brief, their recommendation is to perform feature selection inside the cross-validation loop if the goal is to compare two different algorithms, and to perform selection both inside and outside of the cross-validation loop if the goal is to determine which set of features is best for a particular dataset. Also see and refer to Hastie *et al.* (2009), pp. 245-248.

2. Measure the effect of feature selection on *efficiency*; in other words, ascertain what the fewest number of features are that are necessary for acceptable results. Often, the number of features needs to be considered against the effect on classification accuracy, as the two aspects could be inversely related.
3. In simulation studies, where the truth in terms of which features are relevant is known, it is also possible to estimate the *probability of correct selection (PCS)*. This measures the likelihood of selecting the appropriate features from a set of features which includes both relevant and irrelevant features (noise).

Another aspect that should be mentioned here, is that of *feature selection bias*. Feature selection bias can occur when the same training dataset is used for both feature selection and classifier learning. For instance, selecting only features that have a high correlation with the response will lead to overly-confident predictions, as such correlations could occur purely by chance. Singhi and Liu (2006) point out that this bias can exacerbate overfitting and also negatively affect classification performance. One possible way of avoiding such a bias is to split the training dataset into two separate parts, one used for feature selection and one for learning. However, in practice, such an additional split in data is seldom performed since as much data as possible should ideally be used for both feature selection and classifier learning. The authors performed an extensive empirical study of the effect of feature selection bias, and find that the effect is not as detrimental in classification problems as in regression problems, mostly because selection bias has a limited impact on the decision boundary in classification problems while in regression the impact on sample regression coefficients is more severe. A good approach to limit the effect of feature selection bias, is to use cross-validation when splitting a dataset into training and test components, and to make sure that feature selection is performed inside the cross-validation loop and not outside (Li *et al.*, 2008).

5.4 General approaches to feature selection

In this section we briefly discuss feature selection contributions for cases where every entity is assigned a single label.

5.4.1 Exhaustive subset search

An exhaustive search of all possible subsets of features will ensure that the best subset (in the context of Section 5.3 above) is found. However, this is usually computationally impractical, even when the number of features is not too large – for n features, the number of possible subsets that would need to be evaluated is $2^n - 1$. In fact, Gheyas and Smith (2010) point out that the problem of finding the best feature subset is known to be an NP-complete problem. Such an exhaustive subset search is therefore seldom performed in practice, and some other way of feature selection needs to be found.

5.4.2 Filter approach

The filter approach selects feature subsets as a pre-processing step, meaning that features are selected independently of the learning algorithm employed. The most general approach is to rank features according to some scoring criterion and then select the top k features. Filters are computationally inexpensive and are simple to implement (since only p scores need to be calculated, where p is the number of features in the dataset). Filters are also robust against overfitting, since although they increase bias, they may have considerably less variance. A drawback of the filter approach is that redundant features will not be identified, as they are likely to have similar rankings.

When implementing a filter approach, the scoring criterion used for ranking features needs to be specified, as should the threshold point for determining relevance (or alternatively the number of features k that are to be selected). Correlation coefficients are widely used for ranking features, and provide an easy and interpretable way of understanding the relative importance of features.

Other criteria that have been used and are referred to in Spolaôr *et al.* (2013) and Gheyas and Smith (2010) are the chi-square test, reliefF, the Gini Index, mutual information, information gain and the Wilcoxon Mann-Whitney test. We will now

briefly discuss ReliefF and information gain, as these are the criteria used in the multi-label feature selection studies discussed later in this chapter.

The ReliefF measure belongs to the family of relief algorithms which are based on feature weighting. These algorithms estimate the quality of features according to how well the value of a given feature helps to distinguish between instances that are near to each other. These algorithms have several benefits such as low bias and the ability to include interactions among features (Sánchez-Marño *et al.*, 2007). ReliefF is a specific implementation of a relief algorithm, specifically designed for multiclass problems. It is robust in the presence of noise, and includes an approach for estimating missing data (Duch, 2006). The basic idea of ReliefF is to reward an attribute for having different values on a pair of examples from different classes, and to penalise it for having different values on examples from the same class (Spolaôr *et al.*, 2013). Its values range from -1 to 1, with larger positive values corresponding to features deemed to be important.

Information Gain (IG) measures the dependence between one feature and the class label. It calculates the difference between the entropy of the dataset and the weighted sum of the entropies of subsets of the data. A high IG value for a feature implies that there is strong dependence between the feature and the label. Mathematically:

$$IG(D, X_j) = entropy(D) - \sum_{v \in X_j} \frac{|D_v|}{|D|} entropy(D_v)$$

where $X_j, j = 1, \dots, p$ are the features in the dataset D , $D_v \subseteq D$ where D_v consists of all the examples where $X_j = v$. Also, $|D|$ denotes the cardinality of the set D .

For any chosen filter method, an open question is the specification of the threshold to use in deciding which features to include for selection. One often-used proposal in this regard is to use cross-validation.

Despite the drawbacks of filter methods, they are widely used due to their simplicity. Guyon and Elisseeff (2003) report that good empirical success has been obtained with filter methods. Duch (2006) and Sánchez-Marño *et al.* (2007) give good overviews of the use of filter methods for feature selection.

5.4.3 Wrapper approach

Wrapper methods select a subset of features by using a specific learning algorithm to evaluate features and to determine which features should be selected. This is done by evaluating the performance of the learning algorithm using different subsets of features, which means that the wrapper approach is computationally expensive since the learning algorithm needs to be called multiple times. Performance assessments are usually done using a validation dataset or by cross-validation.

The choice of learning algorithm is not relevant to the implementation of a wrapper method; any learning algorithm could be used, since the performance of the algorithm is simply used to determine how useful the different subsets of features are. Gheyas and Smith (2010) state that the SVM is the most commonly used learning algorithm for wrappers. Other popular choices are naive Bayes and least-squares linear predictors (Guyon and Elisseeff, 2003).

Since an exhaustive search of all possible subsets is not practical (as explained in Section 5.4.1), an efficient search strategy has to be devised. For this purpose, greedy search strategies are often utilised, as they are computationally efficient. Well-known examples of greedy search strategies are forward selection and backward elimination. In forward selection, features are progressively added until a point is reached where adding additional features makes no significant difference to the performance of the learning algorithm. In backward elimination, the starting point is the full set of features; features are then progressively eliminated. There is some evidence that using coarse search strategies such as forward selection or backward elimination may alleviate the problem of overfitting (Guyon and Elisseeff, 2003). Other search strategies which may be used include branch-and-bound, floating and randomised search.

A benefit of wrapper methods is that, unlike filter methods, the number of features can be automatically determined, and redundant and interacting features can be detected. These methods are also often more effective than filter methods (Gheyas and Smith, 2010). A major drawback, as mentioned before, is that wrapper methods

are computationally expensive – however, this can be partly overcome by using efficient search strategies.

5.4.4 Embedded approach

The embedded approach is employed by some learning algorithms such as decision trees, in which feature selection is incorporated as part of the training process. In such algorithms, the feature which is best at discriminating between classes is determined at each stage of the iterative training process.

5.4.5 Other approaches

Some other approaches that have been suggested for reducing dimensionality of a dataset include clustering and singular value decomposition (SVD). These are unsupervised methods; that is, they do not use the information provided by the response. With the use of clustering for feature construction, the idea is to replace a group of “similar” variables by a cluster centroid, which becomes a feature (Guyon and Elisseeff, 2003). In the case of SVD, the goal is to form a set of features that are linear combinations of the original variables, which provide the best possible least-squares reconstruction of the original data (Guyon and Elisseeff, 2003).

Hybrid (combined filter and wrapper) approaches have also been suggested. The idea here is to first apply a filter method to reduce the number of features by eliminating the most irrelevant ones, and then to use a wrapper method to find the optimal subset among the remaining features (Gheyas and Smith, 2010). Duch (2006) also describes a ‘filtrapper’ approach, where features are ranked by a filter method, but the number of features that are eventually selected is determined by a wrapper method. This leads to faster selection, but interacting features could still be excluded.

5.5 Multi-label feature selection

In this section we focus on the problem of feature selection when multiple labels can be assigned to each entity.

5.5.1 Overview of multi-label feature selection

Multi-label feature selection is a complex matter. Not only do correlations and interactions between different features and with more than one label have to be taken into account, but there are generally also correlations between labels.

Despite the importance of feature selection, and the relevance of multi-label learning, as yet relatively little has been published regarding multi-label feature selection. A systematic review by Spolaôr *et al.* (2012) found only 49 papers related to multi-label feature selection.

Multi-label feature selection can be based on filters, wrappers or an embedded method. According to Spolaôr *et al.* (2013), the filter approach is the one most commonly used for multi-label feature selection methods. In addition, the problem can either be approached by transforming the multi-label dataset into several single-label ones by applying one of the problem transformation methods described in Chapter 4 (such as binary relevance or label powerset), or selection can take place directly on multi-label data without transforming the data to single-label.

5.5.2 Problem transformation approaches

In a problem transformation approach to feature selection, the multi-label dataset is transformed into one or more single-label datasets through one of the problem transformation approaches outlined in Chapter 4, Section 4.4. This generally means that either a binary relevance (BR) transformation is applied (Section 4.4.1), in which the K -label multi-label dataset is transformed into K binary single-label datasets, or a label powerset (LP) transformation can be applied (Section 4.4.4), in which each

unique combination of labels in the multi-label dataset is seen as a separate class in a corresponding multi-class single-label dataset. When a BR transformation is used, features are independently selected for each binary dataset. If a multi-label classifier is to be used, the results from these feature selection steps are combined in some way, usually by averaging results over all binary datasets. Both of these two methods present their own challenges; for instance, BR does not take label correlations into account, while LP can lead to a sparse and unbalanced dataset (see Chapter 4 for details).

Spolaôr *et al.* (2013) evaluate the use of two different scoring criteria (information gain and ReliefF) in a filter approach using both binary relevance (BR) and label powerset (LP) problem transformations. In the case of BR, they average the ReliefF and IG measures over all K binary datasets obtained through the BR transformation, and select the features with average values greater than or equal to a very conservative threshold of 0.01. They do an empirical evaluation of these two scoring criteria combined with the two different problem transformation approaches, using 10 multi-label benchmark datasets. They evaluate the performance of each approach using different multi-label evaluation measures, and also consider the feature reduction, which they calculate as the average reduction of features obtained by the feature selection method; in other words

$$\text{Feature Reduction}(D, X') = 100 - \frac{100 \times |X'|}{p}$$

where $X' \subseteq X$ is the subset of features selected from dataset D with p features, and $|X'|$ denotes the cardinality of the subset. The authors find that there is a very high variation in the feature reduction measure, ranging from 0% (meaning that all features were deemed relevant) to 99.55% in the case of one of the datasets for one of the selection methods (meaning that only 0.45% of the features were considered relevant). They even found such large variations in average feature reduction within the same dataset across different selection methods, highlighting the importance of choice of problem transformation approach and scoring criteria when performing feature selection.

For each of the four approaches considered, Spolaôr *et al.* (2013) evaluate predictive performance by implementing a BRkNN classifier (see Chapter 4, Section 4.5.1).

They compare this to the predictive performance obtained by implementing a BRkNN classifier on the full feature set. They find that ReliefF performs better than Information Gain, possibly because ReliefF takes feature interactions into account whereas Information Gain does not. They find little difference, however, between the measures obtained using the different problem transformation methods with the same scoring criterion.

Trohidis *et al.* (2008) take a label powerset approach to feature selection: they transform the multi-label data to single-label, and then use a chi-square statistic to determine the best features. Their method – unlike binary relevance – takes label correlations into account, and empirical evidence shows that this approach outperforms two other approaches where feature selection is done separately for each label and then combined using an averaging approach.

Doquire and Verleysen (2011) also base their multi-label feature selection approach on the label powerset transformation, but they use the pruned problem transformation (PPT) adaptation of label powerset proposed by Read (2008) (see Section 4.4.4). They then use a greedy forward feature selection algorithm based on mutual information. Their approach leads to improved performance when compared to the approach of Trohidis *et al.* (2008), possibly because their selection procedure takes feature redundancy into account, whilst the simple ranking procedure employed by Trohidis *et al.* (2008) does not.

5.5.3 “True” multi-label approaches

Zhang *et al.* (2009) use a multi-label Naive Bayes classifier which they adapt to incorporate feature selection; according to them, their study is the first to incorporate feature selection techniques into the design process of multi-label algorithms. They first use principal component analysis (PCA) to eliminate irrelevant and redundant features and thereby reduce the size of the feature pool. They then use a genetic algorithm (GA) to select a subset of the features, using a fitness function which ensures that correlations between labels are taken into account. Their approach can therefore be considered a hybrid filter-wrapper approach, first employing PCA as a

filter method and then applying a GA with a multi-label Naive Bayes classifier as a wrapper method. Their empirical study considers synthetic and real-world data, and for the former they propose an algorithm for generating synthetic multi-label data. The results of their empirical study show consistently better performance utilising feature selection techniques compared to cases where no feature selection is applied.

Lee and Kim (2013) select a feature subset by maximising the dependency between selected features and labels. They accomplish this by decomposing the calculation of high-dimensional entropy into a cumulative sum of multivariate mutual information. They claim that their approach is the first where a feature filter criterion takes label interactions into account in evaluating the dependency of the given features without resorting to transforming the multi-label problem into a single-label one.

Gu *et al.* (2011) attempt to learn label correlations at the same time as feature selection. They built their model on LaRank SVM (a modified SVM which allows ranking of labels). They incorporate label correlations by placing a matrix-variate normally distributed prior on the weight vectors of the LaRank SVM. For feature selection they introduce a binary variable for each feature which indicates whether a particular feature is selected or not. Their aim is to find a subset of features such that the label correlation regularised loss of LaRank SVM is minimised. The size of the feature subset to be selected is estimated through regression.

Lastra *et al.* (2011) extend the Fast Correlation-Based Filter (FCBF) (Yu and Liu, 2004) to the multi-label case. In doing this, they use Symmetrical Uncertainty (a normalised version of mutual information) and maximum spanning trees to obtain a graphical representation of the relevance relationship between labels and features. They find that label rankings learned from a direct multi-label point of view (as opposed to rankings obtained through problem transformation methods) perform better.

Kong *et al.* (2012) adapted ReliefF to be used directly with multi-label data. They achieve this by decomposing ReliefF to a collection of two-class problems with an adaptation to account for ambiguous cases, and on the image annotation datasets they

considered, their approach generally yielded better results compared to problem transformation approaches.

5.6 Multi-label feature selection based on probe variables

5.6.1 Probe variables

In any feature selection method, a difficult aspect is the decision of how many features should be selected. With a filter approach, a threshold needs to be determined as a cut-off point for which features should be included and which should not. Alternatively, the number of features to select has to be specified, preferably in a data dependent manner.

Tuv *et al.* (2008) use independent probes to assist with this decision. However, they are not the first to employ the concept of probes for feature selection (see for example Bi *et al.*, 2003 and Stoppiglia *et al.*, 2003). The basic idea of probes is to add a number of randomly generated features – which are independent of the response variable Y – to the original set of features. It is assumed that an effective feature selection method which evaluates the relative importance of features would rank relevant features higher than these probes, which means that the probes could act as a baseline to determine the cut-off point for determining relevant features.

Bi *et al.* (2003) draw values for these probe features from a normal distribution. However, according to Tuv *et al.* (2008) this is not sufficient, since the original feature values may exhibit some special structure which needs to be taken into account. They therefore follow Tusher *et al.* (2001) in instead employing randomly permuted values of the original features. Something similar is done in random forests; see for example Hastie *et al.* (2009), p. 593.

Tuv *et al.* (2008) use ensemble-based classifiers – specifically, random forests – to derive a measure of feature importance by averaging (across all trees in the forest) how often different features were used in constructing the splits of the trees. This leaves them with a relative feature ranking. As mentioned previously, a stable feature

ranking method – such as random forests – should assign a significantly higher ranking to relevant features than to the independent probes. These probes can therefore be used to determine the ranking cut-off point for inclusion of features. However, any measure of feature importance could potentially be used. For small sample sizes, Tuv *et al.* (2008) recommend that the process of generating independent probe features and ranking features should be performed several times in order to obtain statistical significance.

5.6.2 Multi-label feature selection using independent probes

We propose a multi-label feature selection method, based on a binary relevance (BR) problem transformation, and using correlation as a measure of feature importance, together with probes generated by randomly permuting feature values. To the best of our knowledge, this is the first time that the idea of independent probes has been used in a multi-label feature selection context.

Consider training data $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N\}$, where $\mathbf{x}_i: p \times 1$ contains observations on p predictor features X_1, X_2, \dots, X_p , and $\mathbf{y}_i: d \times 1$ denotes an unordered subset of labels from a label set $\mathcal{L} = \{1, \dots, K\}$. We assume, somewhat restrictively, that exactly d labels are associated with each data case. The intention is to use the binary relevance (BR) strategy to assign a label subset $\mathbf{y}(\mathbf{x})$ to a new data case \mathbf{x} . We suspect that not all of X_1, X_2, \dots, X_p are important, and the problem is to use the data to identify the relevant features.

It is useful to consider the data in the following way. Let $X: N \times p$ contain the \mathbf{x}_i^T as rows, $i = 1, \dots, N$, and let $L: N \times K$ be a matrix of zeroes and ones, with $\ell_{ik} = 1$ if and only if $k \in \mathbf{y}_i, i = 1, \dots, N; k = 1, \dots, K$. We therefore have

$$[X \quad L] = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} & \vdots & \ell_{1k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} & \vdots & \ell_{Nk} \end{bmatrix}$$

where each row of L contains exactly d ones. Also, $\sum_{i=1}^N \ell_{ik} = \ell_{+k}$ is the number of data cases in which label k appears, $k = 1, \dots, K$. We write $\mathbf{x}_{(j)}$ for the (column) vector

containing the N observations of $X_j, j = 1, \dots, p$, and $\boldsymbol{\ell}_k$ for the k^{th} column of $L, k = 1, \dots, K$.

In the BR approach, K binary classifiers are constructed, using the datasets $\{X, \boldsymbol{\ell}_k\}, k = 1, \dots, K$. Let $f_k(\boldsymbol{x})$ denote a measure calculated for a new case \boldsymbol{x} from the k^{th} binary classifier. For example, $f_k(\boldsymbol{x})$ can be the posterior probability of a positive response when classifier k is applied to \boldsymbol{x} . In the BR approach we construct $\boldsymbol{y}(\boldsymbol{x})$ as follows. Let $f_{(1)}(\boldsymbol{x}) < f_{(2)}(\boldsymbol{x}) < \dots < f_{(K)}(\boldsymbol{x})$ be the ordered $f_k(\boldsymbol{x})$ values, and suppose k_1, \dots, k_K is the permutation of $1, \dots, K$ corresponding to this ordering. Then we take

$$\boldsymbol{y}(\boldsymbol{x}) = [k_K, k_{K-1}, \dots, k_{K-d+1}]^T.$$

The following is an obvious idea for feature selection in this context. Let $R_{LX} \equiv [r_{LX}(k, j)]$ be the matrix of absolute correlations between L and X , i.e. $r_{LX}(k, j) = |\text{corr}(\boldsymbol{\ell}_k, \boldsymbol{x}_{(j)})|, k = 1, \dots, K; j = 1, \dots, p$. We order each row of R_{LX} decreasingly, thereby obtaining a ranking of X_1, \dots, X_p in terms of their importance for label $k, k = 1, \dots, K$. The familiar difficult question in feature selection problems arises: how many of X_1, \dots, X_p should be selected? In the present context this question is relevant for each label, as well as possibly in an overall sense.

We propose the following approach to answer this question. It is based on the probe variable approach used by Tuv *et al.* (2008). A probe variable Z_j for X_j is obtained by randomly permuting the values in $\boldsymbol{x}_{(j)}, j = 1, \dots, p$. If X_j is a relevant variable for label k , this should be reflected in $|\text{corr}(\boldsymbol{\ell}_k, \boldsymbol{x}_{(j)})|$ being significantly larger than $|\text{corr}(\boldsymbol{\ell}_k, \boldsymbol{z}_{(j)})|$. This consideration is implemented in the following proposal.

Let $Z: N \times p \equiv [\boldsymbol{z}_{(1)}, \dots, \boldsymbol{z}_{(p)}]$ be the matrix obtained by randomly permuting the rows of X , and write R_{LZ} for the $K \times p$ matrix of absolute correlations between $\boldsymbol{\ell}_k$ and $\boldsymbol{z}_{(j)}, k = 1, \dots, K; j = 1, \dots, p$. We generate B such matrices, Z_1, \dots, Z_B and compute the corresponding correlation matrices $R_{LZ}(b), b = 1, \dots, B$. Let $r_{LZ}^{(b)}(k, j)$ denote the $(k, j)^{\text{th}}$ element of $R_{LZ}(b)$. The values $r_{LZ}^{(b)}(k, j), b = 1, \dots, B$ can be used as follows to decide whether X_j is a relevant feature for label k . Denote the $r_{LZ}^{(b)}(k, j)$ -values

ordered increasingly by $w_{kj}(b)$, i.e. $w_{kj}(1) < w_{kj}(2) < \dots < w_{kj}(B)$. Suppose a value $\alpha, 0 < \alpha \leq 1$ is given. Then we calculate a value for judging the relevance of X_j for label k from

$$c_{kj} = w_{kj}([\alpha B])$$

where $[x] =$ the largest integer $\leq x$. We view $r_{LX}(k, j) < c_{kj}$ as an indication that X_j is not relevant for label k . We can now compute a matrix $A: K \times p$ of indicator variable relevancies by taking $a_{kj} = \text{Ind}(r_{LX}(k, j) > c_{kj})$, $k = 1, \dots, K; j = 1, \dots, p$.

This matrix A can be used for feature selection as follows. Consider an example of such a matrix,

$$A: K \times p = \begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \dots & 1 \end{bmatrix}$$

The column total $a_{+j} = \sum_{k=1}^K a_{kj}$ gives the total number of times X_j was deemed relevant for a label, $j = 1, \dots, p$. Clearly we should select the features having large column totals. The simplest approach in this regard is to select X_j if and only if $a_{+j} = K$. Alternatively, if we decide to use $p_1 < p$ variables, we can select those corresponding to the p_1 largest column totals. If in this process ties occur, we can increase α and / or B . However, the choice of p_1 still is arbitrary. It was stated earlier that probe variables are introduced to specifically answer this question. Note also that even if we select X_j if and only if $a_{+j} = K$, the number of selected variables depend on α . Ideally therefore we should have a method for determining α from the data.

In summary therefore, the proposed technique is as follows:

1. For any given multi-label dataset, transform the data into K binary classification problems using the binary relevance method.
2. Let $f_k(\mathbf{x})$ denote a measure calculated for a new case \mathbf{x} from the k^{th} binary classifier. Arrange these values $f_k(\mathbf{x})$ in ascending order and let k_1, \dots, k_K be the permutation of $1, \dots, K$ corresponding to this ordering.

3. Let R_{LX} denote the matrix of absolute correlations between L and X , and order each row of R_{LX} decreasingly, thereby obtaining a ranking of X_1, \dots, X_p in terms of their importance for label k .
4. Randomly permute the rows of X to obtain matrix Z .
5. Let R_{LZ} denote the matrix of absolute correlations between ℓ_k and $\mathbf{z}_{(j)}$. Generate B such matrices Z_1, \dots, Z_B and compute the corresponding correlation matrices $R_{LZ}(b)$.
6. Let $r_{LZ}^{(b)}(k, j)$ denote the $(k, j)^{\text{th}}$ element of $R_{LZ}(b)$, and denote these values $r_{LZ}^{(b)}(k, j)$ ordered increasingly by $w_{kj}(b)$.
7. For a given value α , $0 < \alpha \leq 1$, calculate $c_{kj} = w_{kj}([\alpha B])$, where $[x]$ is the largest integer $\leq x$.
8. Compute a matrix A by taking $a_{kj} = \text{Ind}(r_{LX}(k, j) > c_{kj})$.
9. If the column total $a_{+j} = \sum_{k=1}^K a_{kj}$ gives the total number of times X_j was deemed relevant for a label, we select features with large values of a_{+j} . Some suggestions for selection in this context are:
 - a. Select X_j if and only if $a_{+j} = K$. This is a very strict way of selecting features.
 - b. Select X_j if and only if $a_{+j} >$ some fraction of K ; for instance, select X_j if and only if $a_{+j} > 0.75 \cdot K$. In this example, a feature will only be selected if it is relevant for at least 75% of the labels.
 - c. If $p_1 < p$ features are required, select only those feature corresponding to the p_1 largest values of a_{+j} .
10. If ties occur, increase the values of α and / or B .

Some areas for further research could be:

1. Relevant features can be selected for each label k separately, with only these features being used in the separate steps of the BR scheme.
2. The correlation coefficient is only one possibility regarding a measure of dependence to use in the selection. It can be replaced by a more general measure, for example a measure based on the binary classifier implemented in the BR scheme. For instance, let v_{kj} denote a measure of the importance of X_j when the (binary) base classifier is applied to the data $\{X, \ell_k\}$, $k = 1, \dots, K; j =$

- 1, ..., p . In the discussion above we had $v_{kj} = r_{kj}$, the (absolute) correlation between ℓ_k and $x_{(j)}$. If for example our base classifier is (binary) logistic regression, we could replace the correlation coefficients by the logistic regression coefficients. Such an approach would of course be much more computationally intensive than the approach based on correlation coefficients.
3. If K is not too large, or if the number of distinct label sets \mathbf{y}_i appearing in the data is relatively small, the individual labels k can possibly be replaced by these label sets. A problem with this approach could be that some of the distinct label sets appear only a small number of times in the data.
 4. Cross-validation might be a way of selecting α from the data, provided the ℓ_{+k} 's are not too small.
 5. Instead of a simple BR scheme, the proposed feature selection method could also be implemented using various variations on the BR scheme such as 2BR, BR+ and classifier chains. (These variations on BR were discussed in Sections 4.4.1 and 4.4.2.)

5.7 Summary

In this chapter, we have looked at feature selection, briefly defining its aims and benefits and presented a short overview of feature selection for single-label problems. We then examined some prior work done on multi-label feature selection and pointed out that this is a relatively new area of research, with as yet relatively few publications addressing the problem. We presented a novel technique for multi-label feature selection based on independent probes, and highlighted some possibilities for further research in this regard. This technique for multi-label feature selection will be employed and evaluated in the empirical work in Chapters 7 and 8 of this dissertation.

CHAPTER 6

Generating Multi-Label Data

6.1 Introduction

Since the field of multi-label data has only really come to the foreground in the past 5-6 years, the number of available benchmark datasets is still fairly limited (see Section 4.10 for a description of widely used publicly available multi-label benchmark datasets).

Benchmark datasets also tend to be limited in terms of certain aspects, especially with regards to label cardinality and density. For instance, of the 22 benchmark datasets formatted for use with Mulan²⁵ (an open-source Java library for multi-label learning), only 4 have a label density of more than 0.1 and only 2 have a label density of more than 0.2. In addition, only 3 have a cardinality of more than 5, while 15 have label cardinality of less than 3.

²⁵ <http://mulan.sourceforge.net> (accessed 4 June 2013)

As Luaces *et al.* (2012b) point out, the low cardinality of these datasets means that many multi-label benchmark problems are almost multi-class learning tasks rather than true multi-label problems. In addition, the fairly low density of the benchmark datasets means that classifying no labels for any new input will lead to a fairly low misclassification rate. This is an important aspect to take into account when deciding which loss function or measure to use when evaluating classification accuracy. For instance, Hamming loss is fairly widely used as a measure across comparative multi-label studies; however, given that Hamming loss is simply the proportion of misclassifications, a favourable (low) value for the Hamming loss in such studies may very well be a result of the underlying structure of the benchmark datasets used rather than an indication of a well-performing algorithm.

Ideally, to investigate the different aspects and concerns relating to multi-label data, one would need to do a carefully controlled simulation study. One of the main advantages of working with simulated data is that it gives full control of the desired properties of the data, without any noise obscuring some characteristics or outcomes, therefore allowing one to focus on the most relevant issues.

Simulating multi-label data is not an easy task. Simply concatenating or aggregating several single-label datasets means that there will be no dependencies between labels, which is generally not the case in multi-label problems. However, as yet relatively little has been published about the simulation of multi-label data.

In this chapter we will review some previous contributions regarding the generation of synthetic multi-label datasets and highlight some shortcomings of these approaches (Section 6.2). We will then present a new technique for simulating multi-label data in Section 6.3, which allows for explicit control over the number of labels, label density as well as approximate control of label correlations. It also allows for the inclusion of both relevant and irrelevant features, so that feature selection strategies can be evaluated.

6.2 Previous approaches to simulating multi-label data

Although some earlier works have looked at generating simple synthetic multi-label data, none of these methods generalise well and, according to Read *et al.* (2009a), seem only to highlight certain characteristics of the algorithm(s) that the authors present, presenting data with few features, labels and examples and therefore not intended for large scale multi-label evaluation.

Some more extensive work on the generation of multi-label data has been done by Read *et al.* (2009a) and Read *et al.* (2012), but their focus has been on generating synthetic multi-label data streams. Examples of data streams are data from sensor applications, measurements in network monitoring or traffic management, log records or click-streams in web exploration, emails and social networks. (Read *et al.*, 2012). Due to the sequential and continuous nature of data streams, such data cannot be handled in a traditional batch learning environment.

In generating synthetic multi-label data streams, Read *et al.* (2012) focus on dependencies between labels, and the generator method they present is able to incorporate both conditional and unconditional dependencies between labels.

To synthesise unconditional dependence, the authors require the specification of two parameters: the average number of labels per example z , $z \in [0, L]$ as well as the “amount” of dependence among the labels u , $u \in [0, 1]$. They then generate a prior probability mass function π , where $\sum_{j=1}^L \pi_j = z$, L is the number of labels and where π_j gives the prior probability of the j^{th} label, i.e. $\pi_j = P(Y_j = 1)$. From π they can then generate a conditional distribution θ over all possible label pairs with $\theta_{jk} = P(Y_j = 1 | Y_k = 1)$ by randomly setting u of these values to be $\in [\min(\pi_j, \pi_k), \max(0, (\pi_j + \pi_k - 1))]$ (i.e. labels are dependent) and the rest to $\theta_{jk} \approx \pi_j$ (i.e. labels are independent). Then, using Bayes’ rule so that $\theta_{kj} = (\theta_{jk} \cdot \pi_k) / \pi_j$, label dependence can be modelled by calculating the joint distribution as follows:

$$\mathbf{p}_{\theta}(\mathbf{y}) = p(y_1) \prod_{j=2}^L p(y_j | y_{j-1})$$

where θ is a matrix with diagonal $\theta_{jj} = \pi_j$.

Their suggestions for values for π and u , after analysing real-world data, is for π to be generated from a uniform (0;1) distribution, followed by normalising it by z (the label cardinality, which needs to be specified beforehand), and for $u = zL / (\frac{L(L-1)}{2})$.

To synthesise conditional dependence, they use a binary generator g to get $\mathbf{p}_{g;\xi}(\mathbf{y} | \mathbf{x})$, where ξ is a mapping in which all A variables are mapped to the A most likely occurring label combinations. These most likely occurring label combinations can be obtained from sampling $\mathbf{p}(\mathbf{y})$. The mapping ξ defines the relationships $\xi[a] \rightarrow \mathbf{y}_a$ where \mathbf{y}_a is the a^{th} most likely combination of labels.

Their finding is that their method can provide data which is very similar to real-world data and which therefore works well for general analysis and evaluation of multi-label algorithms. However, it does not allow for generating relevant as well as irrelevant features, and also does not allow explicit control over the correlations between the features or labels.

As far as we could determine, the only other paper looking in depth at synthetic multi-label data generation – but this time outside of a data stream context – is that of Luaces *et al.* (2012b).

In generating multi-label datasets, one of two approaches can be taken. The first is to start by generating labels (if necessary with correlations between the labels), and to then generate feature vectors based on the generated label vectors. Another approach – which is the one followed by Luaces *et al.* (2012b) – is to first generate the feature vectors and then generate the required labels from the generated features in some way. More specifically, they generate feature vectors from a uniform distribution, and then attempt to perform a multi-label classification of the features by using hyperplanes to split the input space into positive and negative regions. Their technique will now be discussed in more detail.

Start with inputs drawn randomly from a uniform distribution $\mathcal{X} = [0,1]^p$ and let \mathbb{X} be the matrix of input instances for the dataset. Let $x_0 \in \mathcal{X} = [0,1]^p$ and $w_0 \in [-1,1]^p$. A hyperplane that passes through x_0 and is perpendicular to w_0 splits the input set \mathbb{X} into 2 subsets, \mathbb{X}^+ and \mathbb{X}^- , where

$$\mathbb{X}^+ = \{x: \langle w_0, x - x_0 \rangle \geq 0\}$$

and

$$\mathbb{X}^- = \{x: \langle w_0, x - x_0 \rangle < 0\}.$$

For linear classifiers, Luaces *et al.* (2012b) construct a first set of randomly generated linear hypotheses, with each one characterised by a collection of hyperplanes h_ℓ , where

$$\{h_\ell = (x_\ell, w_\ell): \ell = 1, \dots, L\}.$$

For non-linear classifiers, they assign relevant labels to regions of the input space defined by the intersection of several hyperplanes that share a common point. In other words, relevant labels are geometrically defined at the interior of pyramids with a certain number of faces. Therefore, in the non-linear case, for a given label ℓ , the hypothesis h_ℓ is defined as

$$\ell \in h_\ell(x) \Leftrightarrow \langle w_k^\ell, x - x_0^\ell \rangle \geq 0 \quad \forall k = 1, \dots, f$$

where f is the number of faces of the pyramid, $x_0^\ell \in \mathcal{X}$ and $w_k^\ell \in [-1,1]^p$.

The problem with this approach is that if the w_k^ℓ values are completely random, the interior of the pyramid may be empty or too small. The authors therefore force the w_k^ℓ 's to form angles within a given range using a Gram-Schmidt procedure (details of which can be found in their paper).

Although this approach of Luaces *et al.* (2012b) generalises well, it does not allow for any explicit control over correlations between labels. Since they also state that the hottest topic in the multi-label learning community is probably “to design new methods able to detect and exploit dependencies among labels”, being able to generate a dataset which allows not only for control over correlations between features but also between labels seems of key importance.

6.3 A simple approach to simulating multi-label data

Our approach is fairly simple, but allows for a good measure of control over aspects such as number of labels, label density, correlations between labels and features as well as allowing for inclusion of relevant and irrelevant features with a view to investigating feature selection strategies.

In short, we generate labels by thresholding observations from a multivariate normal distribution while controlling the number of labels, label densities and the correlations between the labels. We then generate relevant and irrelevant features for each label set, working from the premise that a feature is considered relevant for a label if its distribution when the label is present differs from its distribution when the label is absent. Again, features are generated from a multivariate normal distribution where we control the mean vector as well as the covariance matrix. This technique will now be explained in more detail.

The required parameters are:

- The number of data cases to be generated, N .
- The total number of features p , as well as the number of relevant features p_r ($p_r \leq p$).
- The number of labels K .
- The required label density for each label, given as a vector $\mathbf{D} = [d_1 \ d_2 \ \dots \ d_K]$. For instance, in a case with 3 labels (i.e. $K = 3$) and required densities for the 3 labels of 0.2, 0.3 and 0.4, the density vector would be $[0.2 \ 0.3 \ 0.4]$. This in turn translates to an average density of 0.3 across labels, meaning that the expected label cardinality would be 0.9.
- Σ_Y , a covariance matrix used to control correlations between labels.
- Σ_X , a covariance matrix used to control correlations between features.

Our first objective is to generate a random vector \mathbf{Y} consisting of zeroes and ones, indicating the absence or presence of the different labels. In doing this, we need to take correlations amongst the labels into account. One possibility in this regard is to generate $Y_j = \text{Ind}(W_j > c_j)$. In this expression, $j = 1, 2, \dots, K$ for the different labels while $\mathbf{W}_j = [W_1, W_2, \dots, W_K]$ is from a multivariate normal distribution with mean $\mathbf{0}_{K \times 1}$

and covariance matrix Σ_Y . Furthermore, $c_j = \Phi^{-1}(d_j)$ so that $P(Z > c_j) = d_j$, with $Z \sim N(0,1)$, and where d_j corresponds to the specified label density for the j^{th} label.

The covariance matrix Σ_Y can be used to control correlations between labels, with $\Sigma_Y = I_K$ for the case where no correlations between labels are required, and

$$\Sigma_Y = \begin{bmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{bmatrix}$$

in cases where correlations between labels are required. We only consider the equi-correlated case, with common correlation coefficient ρ , but cases with different correlations between labels would also be possible.

We now consider the generation of the features. An aspect that complicates the data generation process is that cases where no labels are present should be excluded. In other words, the generation of the features is conditional upon

$$\sum_{j=1}^K y_j > 0.$$

With a view to later investigating feature selection strategies, in generating features we want to distinguish between relevant and irrelevant features. To this extent, a feature is considered relevant for a label if its distribution when the label is present differs from its distribution when the label is absent. Consequently, a feature is considered irrelevant if the two distributions (for a label present or absent) are identical.

An \mathbf{X} vector is therefore generated randomly from a multivariate normal distribution, with its mean vector divided into two parts corresponding to relevant and irrelevant features and with a similar partition of its covariance matrix into four submatrices.

Let $\Theta_{p \times K} = [\Theta_1 \quad \Theta_2]$, where $\Theta_{1:p_r \times K}$ refers to the relevant features and $\Theta_{2:(p-p_r) \times K}$ refers to the irrelevant features. In our approach, we draw j vectors of uniform values generated randomly from the interval $(0.49, 0.51)$ and denote these as \mathbf{U}_j (the choice

of (0.49, 0.51) as interval is arbitrary)²⁶. Each of these vectors \mathbf{U}_j will become a column of Θ_1 , after being multiplied by its relevant column number; in other words,

$$\Theta_{1p_r \times K} = [j\mathbf{U}_j]$$

for $j = 1, \dots, K$, p_r the number of relevant features and \mathbf{U}_j as described a $p_r \times 1$ column vector consisting of randomly sampled values from a uniform (0.49, 0.51) distribution. The reason for this specification is to have a progression of relevancy amongst the features as j increases, reflecting growing relevance of the features. The entries in Θ_2 are all zero.

Similarly, Σ_X is written as

$$\Sigma_X = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

where Σ_{11} contains the covariances (and by implication the correlations) of the relevant features only, Σ_{22} contains the covariances of the irrelevant features only, and Σ_{12} (and Σ_{21}) the covariances amongst the relevant and irrelevant features. Using different specifications for Σ_X we can therefore incorporate different covariance structures into the data generation process.

The code for the data simulation process was written in R, a free software environment for statistical computing (R Core Team, 2013), and is given in Appendix A.2.

To illustrate the different possibilities in generating multi-label data using the above method, a few small datasets were created with fixed values for the parameters K , N , p and p_r and varying values for densities \mathbf{D} , covariance matrix for labels Σ_Y and covariance matrix for features Σ_X (with these parameters all as defined on page 150).

The fixed values used were:

- $K = 3$
- $N = 100$

²⁶ While other distributions could certainly also be considered, we chose the uniform distribution as a very easy and economical way of generating values. A short interval (0.49, 0.51) was chosen as a way of avoiding too much variation in feature values.

- $p = 10$
- $p_r = 4$

For densities, three possibilities were investigated: $\mathbf{D} = [0.2 \ 0.3 \ 0.4]$ or $[0.4 \ 0.5 \ 0.6]$ or $[0.7 \ 0.8 \ 0.9]$, giving average densities of 0.3, 0.5 or 0.8 with corresponding label cardinalities 0.9, 1.5 or 2.4.

Σ_Y reflected either correlated (correlations of 0.9) or uncorrelated labels, which means that the following two Σ_Y matrices were used:

$$\Sigma_{Y_{uncorr}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

or

$$\Sigma_{Y_{corr}} = \begin{bmatrix} 1 & 0.9 & 0.9 \\ 0.9 & 1 & 0.9 \\ 0.9 & 0.9 & 1 \end{bmatrix}$$

For Σ_X three different structures were investigated: either no correlations at all, correlations between relevant features only or correlations between all features (correlations of 0.6), leading to one of the following Σ_X formulations:

$$\Sigma_{X_{uncorr}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

or

$$\Sigma_{X_{relcorr}} = \begin{bmatrix} 1 & 0.6 & 0.6 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.6 & 1 & 0.6 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.6 & 0.6 & 1 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.6 & 0.6 & 0.6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

or

$$\Sigma_{X_{relrrcorr}} = \begin{bmatrix} 1 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.6 & 1 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.6 & 1 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.6 & 0.6 & 1 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 1 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 1 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 1 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 1 & 0.6 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 1 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 1 \end{bmatrix}$$

Keeping Σ_Y and Σ_X constant and uncorrelated (in other words, using specifications of $\Sigma_{Y_{uncorr}}$ and $\Sigma_{X_{uncorr}}$), the following sample quantities were calculated for our samples of $N = 100$:

Densities	Label cardinality
[0.2 0.3 0.4]	1.34
[0.4 0.5 0.6]	1.71
[0.7 0.8 0.9]	2.48

It is clear that label cardinality increases as the label densities are increased.

The X -matrices also behave as expected, with the following correlation matrices for the 3 cases (keeping D constant at [0.2 0.3 0.4] and Σ_Y uncorrelated):

No correlations between features ($\Sigma_{X_{uncorr}}$):

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
X_1	1	0.127	0.118	-0.054	-0.085	0.188	-0.040	-0.133	0.059	0.060
X_2	0.127	1	-0.050	0.181	-0.133	-0.013	-0.029	0.086	-0.019	-0.113
X_3	0.118	-0.050	1	0.151	0.035	0.043	-0.109	0.082	0.036	0.191
X_4	-0.054	0.181	0.151	1	-0.149	0.061	-0.044	0.112	0.085	0.065
X_5	-0.085	-0.133	0.035	-0.149	1	-0.058	0.046	-0.093	-0.001	-0.248
X_6	0.188	-0.013	0.043	0.061	-0.058	1	-0.013	0.091	0.011	0.021
X_7	-0.040	-0.029	-0.109	-0.044	0.046	-0.013	1	-0.081	-0.202	-0.069
X_8	-0.133	0.086	0.082	0.112	-0.093	0.091	-0.081	1	0.067	-0.098
X_9	0.059	-0.019	0.036	0.085	-0.001	0.011	-0.202	0.067	1	0.043
X_{10}	0.060	-0.113	0.191	0.065	-0.248	0.021	-0.069	-0.098	0.043	1

Correlations between relevant features only ($\Sigma_{X_{relcorr}}$):

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
X_1	1	0.740	0.742	0.768	-0.055	-0.049	-0.091	0.034	-0.108	-0.065
X_2	0.740	1	0.681	0.751	-0.016	-0.048	-0.259	0.000	-0.207	-0.116
X_3	0.742	0.681	1	0.708	0.056	-0.122	-0.047	-0.058	-0.093	-0.082
X_4	0.768	0.751	0.708	1	0.007	-0.075	-0.096	-0.070	-0.312	-0.094
X_5	-0.055	-0.016	0.056	0.007	1	-0.087	0.008	0.013	-0.084	-0.181
X_6	-0.049	-0.048	-0.122	-0.075	-0.087	1	0.060	0.114	-0.081	0.140
X_7	-0.091	-0.259	-0.047	-0.096	0.008	0.060	1	0.068	0.024	-0.026
X_8	0.034	0.000	-0.058	-0.070	0.013	0.114	0.068	1	0.114	-0.155
X_9	-0.108	-0.207	-0.093	-0.312	-0.084	-0.081	0.024	0.114	1	-0.071
X_{10}	-0.065	-0.116	-0.082	-0.094	-0.181	0.140	-0.026	-0.155	-0.071	1

Correlations between all features ($\Sigma_{X_{relirrcorr}}$):

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
X_1	1	0.544	0.608	0.521	0.539	0.500	0.521	0.564	0.513	0.543
X_2	0.544	1	0.617	0.434	0.549	0.497	0.522	0.541	0.575	0.557
X_3	0.608	0.617	1	0.513	0.651	0.468	0.600	0.572	0.650	0.553
X_4	0.521	0.434	0.513	1	0.474	0.507	0.509	0.558	0.474	0.557
X_5	0.539	0.549	0.651	0.474	1	0.588	0.605	0.617	0.700	0.633
X_6	0.500	0.497	0.468	0.507	0.588	1	0.594	0.647	0.530	0.637
X_7	0.521	0.522	0.600	0.509	0.605	0.594	1	0.587	0.571	0.574
X_8	0.564	0.541	0.572	0.558	0.617	0.647	0.587	1	0.590	0.613
X_9	0.513	0.575	0.650	0.474	0.700	0.530	0.571	0.590	1	0.553
X_{10}	0.543	0.557	0.554	0.557	0.633	0.637	0.574	0.613	0.553	1

However, for the label correlations, at first glance results are somewhat different than expected. Keeping densities fixed at [0.2 0.3 0.4] and features uncorrelated ($\Sigma_{X_{uncorr}}$), label correlations are:

No correlations between labels ($\Sigma_{Y_{uncorr}}$):

	Y_1	Y_2	Y_3
Y_1	1	0.022	-0.332
Y_2	0.022	1	-0.485
Y_3	-0.332	-0.485	1

Correlations between labels ($\Sigma_{Y_{corr}}$):

	Y_1	Y_2	Y_3
Y_1	1	0.524	0.182
Y_2	0.524	1	-0.129
Y_3	0.182	-0.129	1

An obvious explanation for the different than expected correlations is that in generating labels, we discard cases with no labels. This clearly has an effect on the underlying distributions; however, since having cases with no labels attached to them would be pointless (for our purposes, although not impossible in practice), there is no easy way around this. A subject for further study would be to find a way to get the required label correlations even after discarding generated cases with no labels. However, for the purpose of this dissertation we keep up the method as described since it does give correlations between labels (albeit somewhat different correlations from what was expected).

6.4 Summary

In this chapter we highlighted the difficulties involved in generating synthetic multi-label datasets, and discussed some previous approaches to the problem. We then presented a new approach to the problem, which allows for explicit control over many aspects of the data. This new approach will be empirically evaluated in the next chapter.

CHAPTER 7

Results of Simulation Study

7.1 Introduction

In the previous chapter we proposed a new technique for simulating multi-label data, which allows for control over correlations between features as well as correlations between labels. In this chapter the technique is applied in order to simulate multiple multi-label datasets with the aim of investigating the effect of different parameters on classification accuracy. The effect of feature selection is also studied in detail.

The following section (Section 7.2) describes the parameters investigated in the simulation study. In Section 7.3, we discuss the results of the empirical study in detail, considering each parameter separately and also looking at the efficacy of the proposed feature selection method.

7.2 Experimental design

7.2.1 Study parameters

In the empirical study our aim was to investigate the influence of different parameters such as sample size, number of features, correlation between features, number of labels as well as correlation between labels on classification accuracy in a multi-label context. We also investigated the efficacy of feature selection in such a context. To this end, we considered the following factors:

i. Number of features

As explained in Chapter 6, data was generated from a multivariate normal distribution. In the simulation of data, both relevant and irrelevant features were created. In this context, a feature is considered relevant for a label if its distribution when the label is present differs from its distribution when the label is absent. Similarly, a feature is considered irrelevant for a label if its distribution in the presence of the label is no different from its distribution in the absence of the label. This label-specific view of feature relevance can be extended in different ways to define the global (over all labels) relevance of a feature. In fact, it would obviously be possible to distinguish degrees of label relevance, determined by the number of labels. In this study, we generated both relevant and irrelevant features which enabled the evaluation of the efficacy of feature selection. A large number of features (p) was specified, as well as the number of relevant features (p_r). Two scenarios were investigated: $p = 100 / p_r = 20$ and $p = 200 / p_r = 20$, to give 2 different ratios of the number of relevant to the number of irrelevant features. Larger values of p turned out to be impractical given the operating environment since the computations did not finish in a reasonable amount of time. In all of the created datasets we therefore had the scenario where $p \gg p_r$, which is generally a difficult situation for classification algorithms because of the high signal-to-noise ratio.

ii. Number of training cases

The number of training cases was limited to either $N_{train} = 100$ or $N_{train} = 1\,000$. Together with the possible values of p specified in (i) above, it enabled investigation into scenarios where there are many more training points than features, but also the opposite scenario where there are more features than data points (so-called “wide” datasets, which are becoming more commonplace as was briefly touched on in the introduction to Chapter 5).

iii. Number of labels

The number of labels K was taken to be either $K = 3$, $K = 6$ or $K = 12$. The corresponding label density vectors \mathbf{D} were specified as $\mathbf{D} = [0.2\ 0.3\ 0.4]$ for $K = 3$, $\mathbf{D} = [0.2\ 0.3\ 0.4\ 0.2\ 0.3\ 0.4]$ for $K = 6$ and $\mathbf{D} = [0.2\ 0.3\ 0.4\ 0.2\ 0.3\ 0.4\ 0.2\ 0.3\ 0.4\ 0.2\ 0.3\ 0.4]$, for $K = 12$, meaning that although average label density was the same across all generated datasets (0.3), the cardinalities were different.

iv. Correlations between labels

In multi-label problems, correlations between labels are often present and need to be taken into account. Two scenarios were investigated: firstly, data with no correlations between labels (with corresponding covariance matrix denoted by $\Sigma_{Y_{uncorr}}$), and secondly positive correlations between labels of 0.9 (with covariance matrix denoted by $\Sigma_{Y_{corr}}$). It should be kept in mind however, that setting up Σ_Y with correlations of 0.9 does not translate to correlations of 0.9 in the generated dataset (as discussed in Section 6.3 in the previous chapter); in fact, it leads to lower actual correlations between labels, and this was discussed in detail in Chapter 6.

v. Correlations between features

Here 3 scenarios were investigated: no correlations between features (covariance matrix $\Sigma_{X_{uncorr}}$), correlations between relevant features only ($\Sigma_{X_{relcorr}}$) and correlations between all (i.e. relevant and irrelevant) features

($\Sigma_{X_{relirrcorr}}$). In cases of non-zero correlations, these were arbitrarily set at 0.6.

Varying the parameters described above yielded 72 ($2 \times 2 \times 3 \times 2 \times 3$) different configurations to be considered. Table 7.1 shows the parameter values for these 72 configurations; the numbers indicate the number of each configuration as programmed.

Table 7.1: Parameter values for different configurations in Monte Carlo simulations

			Correlations between labels							
			$\Sigma_{Y_{corr}}$			$\Sigma_{Y_{uncorr}}$				
			Correlations between features			Correlations between features				
N_{train}	p	K	$\Sigma_{X_{uncorr}}$	$\Sigma_{X_{relcorr}}$	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{X_{uncorr}}$	$\Sigma_{X_{relcorr}}$	$\Sigma_{X_{relirrcorr}}$		
100	100	3	1	2	3	4	5	6		
		6	13	14	15	16	17	18		
		12	25	26	27	28	29	30		
	200	3	7	8	9	10	11	12		
		6	19	20	21	22	23	24		
		12	31	32	33	34	35	36		
		1000	100	3	37	38	39	40	41	42
				6	49	50	51	52	53	54
				12	61	62	63	64	65	66
200	3		43	44	45	46	47	48		
	6		55	56	57	58	59	60		
	12		67	68	69	70	71	72		

7.2.2 Methodology

For each of the configurations listed in Table 7.1 above, 100 Monte Carlo simulations were performed by generating 100 training and test datasets according to the specified criteria from the appropriate multivariate normal distributions. The process for generating the multi-label data was described in detail in Chapter 6; the R code used

is also replicated in Appendix A.2. The number of training data cases generated was one of the parameters of the study, while the number of test data cases was kept constant at $N_{test} = 1\,000$.

Denote the training data by $D_{train} = \{(\mathbf{x}_i^{tr}, \mathbf{y}_i^{tr}), i = 1, \dots, N_{train}\}$ and the test data by $D_{test} = \{(\mathbf{x}_i^{te}, \mathbf{y}_i^{te}), i = 1, \dots, N_{test}\}$. The generated training data were transformed using the binary relevance method. This created K single label training datasets $D_{single}(j) = \{(\mathbf{x}_i^{tr}, y_{ij}), i = 1, \dots, N_{train}\}, j = 1, \dots, K$, where $y_{ij} = 1$ if label j was present for observation i and $y_{ij} = -1$ otherwise. An SVM was fit to each of these binary relevance datasets, and predictions for the labels of each of the test cases were made. This was done by taking the top t ranking predictions from the fitted SVM for each test data case, where t was predefined to be

$$t = \text{floor}(K * \text{mean}(1 - \mathbf{D})).$$

In fitting the SVM, an RBF kernel was used with hyperparameter $\sigma = \frac{1}{p}$, with p the number of features. The cost parameter C for the SVM was chosen to be $C = 1$, the reason for this is explained in Section 7.2.3. The predicted labels were compared to the true labels of the test dataset, and 6 multi-label evaluation measures were calculated, namely Hamming Loss, Precision, Recall, Accuracy, One-Error and Coverage. Details on the calculation of these can be found in Chapter 4, Section 4.7, but recall that smaller values of Hamming Loss, One-Error and Coverage mean better performance, whereas higher values of Precision, Recall and Accuracy correspond to better performance. The mean of each of these was then calculated over the 100 Monte Carlo simulations.

The above process was repeated with the inclusion of the selection technique proposed in Chapter 5, Section 5.6.2. After feature selection, an SVM was fit to the reduced dataset using the same parameters as before, but this time using the RBF hyperparameter $\sigma = \frac{1}{p_{sel}}$, with p_{sel} the number of selected features. The same multi-label evaluation measures were also calculated, and averaged over the 100 Monte Carlo simulations.

7.2.3 Hyperparameters of the SVM

The first step in the simulation study was to find an optimal value for the cost parameter C for the SVM classifier. For this purpose, a number of datasets were generated with the number of labels kept constant at $K = 3$, and also assuming correlations between labels, but no correlation between features. Different values for the number of features p and the number of relevant features p_r were used together with values for C of $C = 0.1$, $C = 1$, $C = 10$ and $C = 100$. Examination of classification results using the different multi-label evaluation measures indicated that values of $C = 1$ and $C = 0.1$ seemed to give the best results. $C = 1$ was therefore chosen as the value to be used in subsequent simulations, since it is also often proposed as a default value in the literature; for example, the default value for C in the kernlab package's implementation of SVMs in R (Karatzoglou *et al*, 2004) is 1. Table 7.2 below shows the different error measures for different values of C for $p = 100$ and $p_r = 20$. The bold entries indicate the best result over the different choices of C for each error measure. The C values for $p = 1\ 000$ and $p_r = 100$ showed the same general trend. Detailed results can be found in Appendix B.1.

Table 7.2: Error rates for different choices of cost parameter C for SVM

	ERROR MEASURE					
	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
$C = 0.1$	0.2801	0.8118	0.8326	0.6674	0.0706	0.0869
$C = 1$	0.2801	0.8124	0.8320	0.6678	0.0702	0.0876
$C = 10$	0.3024	0.7949	0.8124	0.6462	0.0749	0.1098
$C = 100$	0.3016	0.7950	0.8132	0.6470	0.0743	0.1091

7.3 Results

7.3.1 Scope

In this section we will analyse the obtained results to establish the following:

1. Generally:
 - a. What is the impact of the size of the training dataset (N_{train})?
 - b. What is the effect of the number of features (p)?
 - c. What is the effect of the ratio between training data and number of features $\left(\frac{N_{train}}{p}\right)$?
 - d. What is the effect of label cardinality?
 - e. What is the effect of label correlations?
 - f. What is the effect of correlations between features?

2. Regarding feature selection:
 - a. Overall, how effective is feature selection?
 - b. How many features are selected? How many of these are relevant and how many irrelevant?
 - c. What is the effect of the quantity $\frac{p_r}{p}$ on feature selection?

The detailed results per configuration are given in Appendix B.2 and B.3; here, we will only discuss the results in summary.

7.3.2 Size of training data

Values of $N_{train} = 100$ and $N_{train} = 1\ 000$ were used. The table below shows the mean for each of the error measures, across all values of p , K , Σ_Y and Σ_X (there were no apparent interaction effects of N_{train} with these parameters):

Table 7.3: Average error rates for different choices of N_{train}

	Hamming Loss	Precision	Recall	Accuracy	One- Error	Coverage
$N_{train} = 100$	0.3900	0.5763	0.8242	0.4941	0.2794	0.6784
$N_{train} = 1000$	0.3761	0.5867	0.8430	0.5062	0.2461	0.6022

As could be expected, with more training data to work with, slightly better classification results can be obtained.

7.3.3 Number of features

The following table shows the average error rates for $p = 100$ and $p = 200$. Averages were taken over all values of N_{train} , K , Σ_Y and Σ_X , and again there were no apparent interaction effects between p and these parameters.

Table 7.4: Average error rates for different choices of p

	Hamming Loss	Precision	Recall	Accuracy	One- Error	Coverage
$p = 100$	0.3822	0.5821	0.8351	0.5008	0.2589	0.6335
$p = 200$	0.3840	0.5809	0.8321	0.4995	0.2666	0.6471

Classification results are fairly similar for the two values of p . This is actually an interesting result since in the simulation of data, the number of relevant features for $p = 100$ and $p = 200$ was kept constant at 20. For the results above, no feature selection was performed, which means that classification results are almost just as good when adding an extra 100 “noise” features to the data. This is possibly a result of the classifier used, as SVMs are known to be quite robust to noise. Another classifier might not have performed as well with the addition of so many noise features. An interesting objective for further research would be to evaluate other classifiers as well to see if this result will be replicated.

7.3.4 Ratio between size of training data and number of features

In terms of the ratio between the size of the training data and the number of features, we have four different configurations:

<i>Ntrain</i>	<i>p</i>	<i>Ntrain/p</i>
100	200	0.5
100	100	1
1000	200	5
1000	100	10

Table 7.5 below shows the average error rates for the different values of $Ntrain / p$. Averages were taken over all values of K , Σ_Y and Σ_X .

Table 7.5: Average error rates for different values of $Ntrain/p$

	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
$\frac{Ntrain}{p} = 0.5$	0.3913	0.5755	0.8223	0.4931	0.2850	0.6880
$\frac{Ntrain}{p} = 1$	0.3888	0.5772	0.8261	0.4950	0.2737	0.6687
$\frac{Ntrain}{p} = 5$	0.3767	0.5864	0.8419	0.5059	0.2481	0.6061
$\frac{Ntrain}{p} = 10$	0.3755	0.5871	0.8442	0.5065	0.2442	0.5983

Higher values of $Ntrain/p$ lead to better classification results, which again highlights the difficulties involved in working with wide datasets. Reducing p through feature selection is therefore an important avenue to explore (results of feature selection will be discussed in Section 7.3.8).

7.3.5 Number of labels

One would generally expect a larger number of labels to lead to less accurate classification performance, as it makes the problem more complex. This is backed up by the figures in the following table (Table 7.6), which shows the values for the

different multi-label evaluation measures for different values of K . Averages were taken over all values of N_{train} , p , Σ_Y and Σ_X , and there were no apparent interactions between K and any of these parameters.

Table 7.6: Average error rates for different values of K

	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
$K = 3$	0.3329	0.6804	0.8314	0.5920	0.1965	0.2576
$K = 6$	0.3879	0.5649	0.8461	0.4851	0.2593	0.5519
$K = 12$	0.4285	0.4994	0.8234	0.4233	0.3324	1.1113

For all error measures with the exception of Recall, performance declined as the number of labels increased. The degradation in performance was especially pronounced for Coverage. Recall, on the other hand, stayed approximately constant. Results are possibly somewhat biased by the fact that we use prior knowledge of the number of labels when determining the predicted labels; in real world cases, the true number of labels will often not be known and this will negatively impact classification results.

7.3.6 Label correlations

Correlations between labels appear to lead to substantially better classification results, as shown in Table 7.7(a) below. This is an unexpected result, given that the classification method used was based on a binary relevance transformation of the data, which technically does *not* take label correlations into account.

Table 7.7(a): Average error rates for different values of Σ_Y

	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
$\Sigma_{Y_{corr}}$	0.3263	0.7195	0.8398	0.5860	0.1253	0.3093
$\Sigma_{Y_{uncorr}}$	0.4399	0.4436	0.8274	0.4143	0.4002	0.9713

For further investigation, one additional simulation run was performed, with labels generated using a covariance matrix Σ_Y with negative correlations of -0.9. Other parameters were set at $K = 12$, $N_{train} = 100$, $p = 100$ (with corresponding $p_r = 20$) and no correlations between features in the case of the Σ_X parameter. The results of this additional simulation run are presented in Table 7.7(b) below ($\Sigma_{Y_{corr-0.9}}$), together with the corresponding results for uncorrelated labels ($\Sigma_{Y_{uncorr}}$) and labels with strong positive correlations ($\Sigma_{Y_{corr+0.9}}$).

Table 7.7(b): Average error rates for different values of Σ_Y

	Hamming Loss	Precision	Recall	Accuracy	One- Error	Coverage
$\Sigma_{Y_{corr+0.9}}$	0.3670	0.6389	0.8455	0.5150	0.2063	0.6268
$\Sigma_{Y_{uncorr}}$	0.4915	0.3597	0.7986	0.3308	0.4738	1.6177
$\Sigma_{Y_{corr-0.9}}$	0.4821	0.3960	0.7723	0.3608	0.4608	1.3871

It appears that the better classification results in the presence of label correlations only hold for cases where the label correlations are strongly positive. In the case of strongly negative correlations, classification results are not that much different from the case where no label correlations are present. In Chapter 6 it was explained that in the data simulation process, actual correlations realised after data generation are different from the specified correlations in the covariance matrices. While at first glance this does not present a problem for the simulation process, it does turn out to impact on the classification results in this case. Further investigation revealed that the actual realised correlations for the case where label correlations were specified to be large negative ($\Sigma_{Y_{corr-0.9}}$), were actually more similar to the uncorrelated case than the case where strong positive correlations were specified. As explained in Chapter 6, one cause of this is the fact that in the simulation process we discard cases where no labels were generated. In addition, it should be kept in mind that throughout we are working with a fairly low average label density of 0.3. Increasing these densities will probably also mean that in the simulation process, we will get closer to the intended label correlations when simulating data. This is something that remains to be investigated in further research. For now, we were satisfied that the simulation process produces satisfactory synthetic datasets which enable multi-label methods to be objectively compared.

As for the unexpected result of better classification with (positive) label correlations than without – given the fact that the binary relevance method was applied – it should be kept in mind that although the actual classification process takes place within a binary relevance loop and therefore does not take label correlations into account, the data generation process is not independent of label correlations. Features (that is, the X -matrix) are generated taking label correlations into account, since we specifically want to create relevant features that have a different distribution when a label is present compared to when the label is absent. In the binary relevance loops in the classification process, similar X -matrices are used for each of the repetitions in the loop, and these X -matrices do therefore depend to some extent on the label correlations. The results are therefore not as unexpected as they may have seemed to be at first glance.

7.3.7 Correlations between features

Correlations between features do not have a substantial effect on classification accuracy, as the figures in Table 7.8 show. Classification seemed to be slightly worse when there were correlations between relevant features only compared to when there were no correlations between features, or correlations between relevant and irrelevant features, but these differences were not substantial enough to draw any meaningful inferences.

Table 7.8: Average error rates for different values of Σ_X

	Hamming Loss	Precision	Recall	Accuracy	One- Error	Coverage
$\Sigma_{X_{uncorr}}$	0.3781	0.5854	0.8408	0.5045	0.2507	0.6139
$\Sigma_{X_{relcorr}}$	0.3943	0.5732	0.8186	0.4903	0.2907	0.7028
$\Sigma_{X_{relirrcorr}}$	0.3769	0.5860	0.8415	0.5056	0.2469	0.6041

7.3.8 Overall efficacy of feature selection

The following 2 figures (Figure 7.1(a) and Figure 7.1(b)) show Hamming Loss and Precision for the full set of features as well as the selected set of features across the full set of 72 simulation scenarios. Only Hamming Loss and Precision are shown, since Recall, Accuracy, One-Error and Coverage show the same general trend. The figures clearly show that there is almost no difference in terms of classification results between using the full set of features compared to using the selected set only. Given the fact that, depending on the configuration, there is a 51% – 88% reduction in the number of features used (see Section 7.3.9), feature selection is clearly very effective.

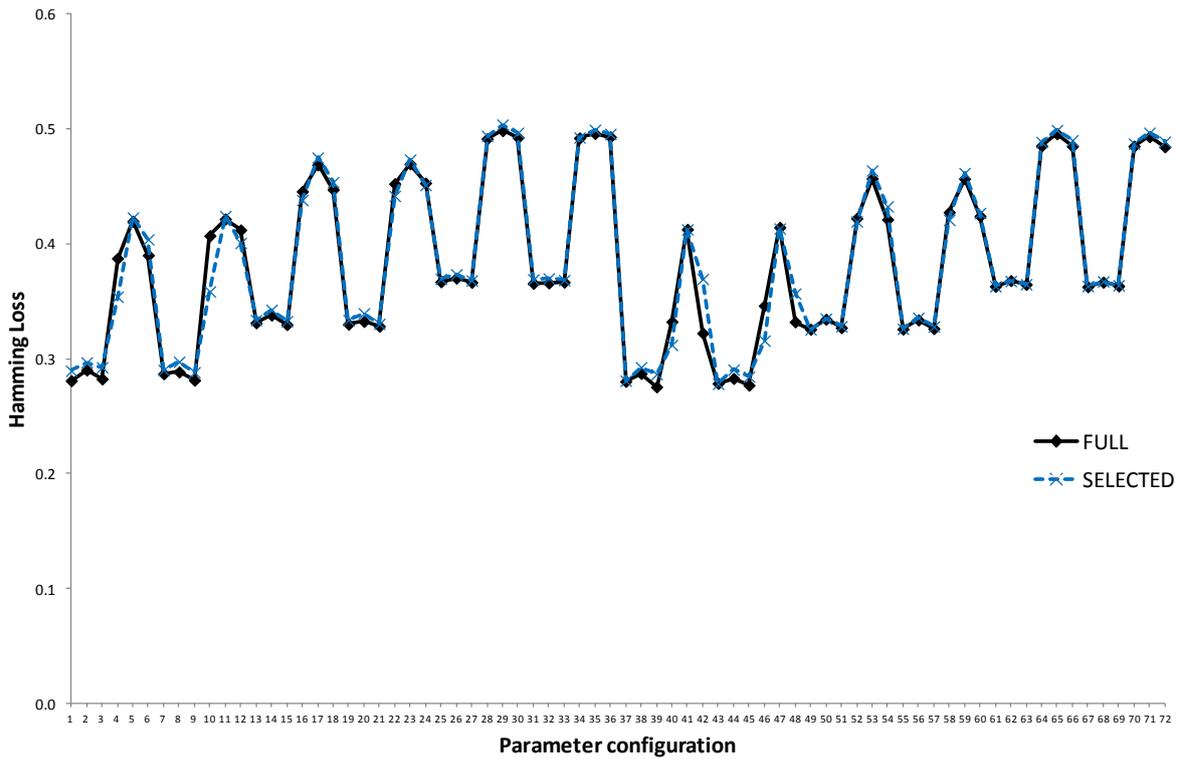


Figure 7.1(a): Hamming Loss for different parameter configurations for full feature set as well as selected feature set

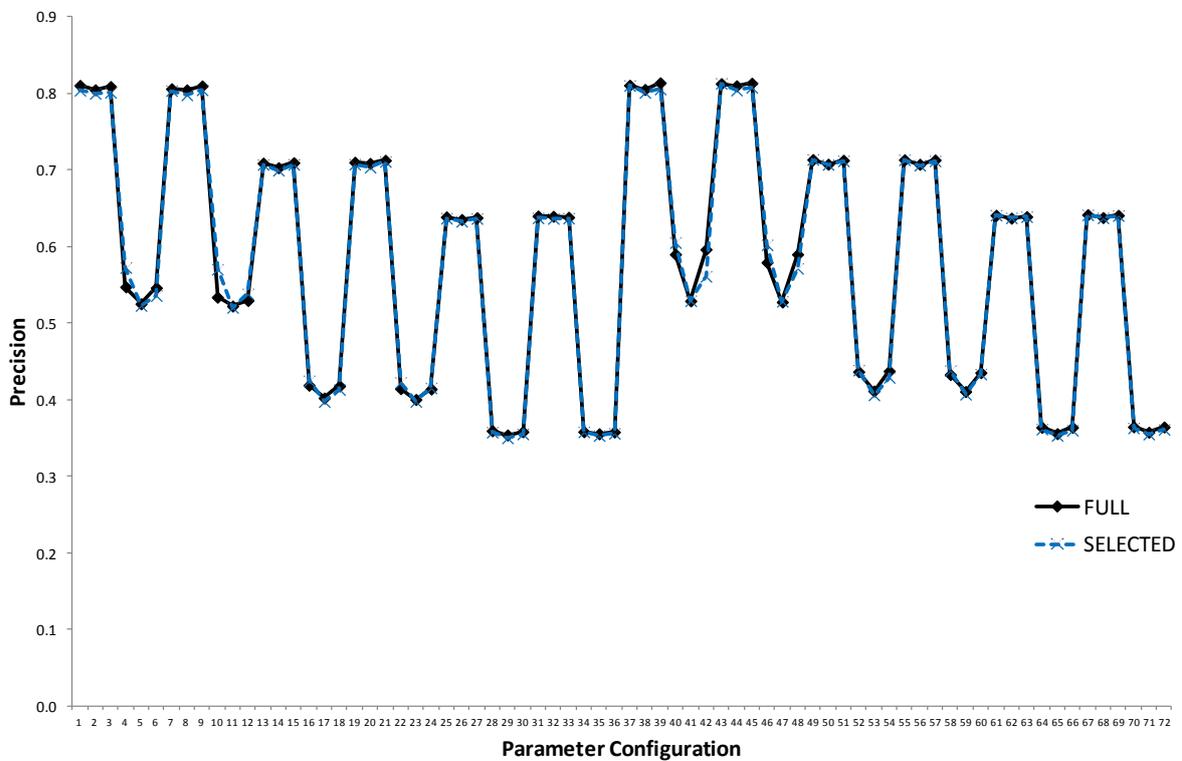


Figure 7.1(b): Precision for different parameter configurations for full feature set as well as selected feature set

Both graphs show peaks and troughs occurring in groups of three; these peaks and troughs correspond to better classification performance for the configurations where there are correlations between labels compared to configurations where there are no correlations between labels.

7.3.9 Number of features selected

As outlined in Section 7.2, due to the way data was simulated it meant that there were always 20 relevant features in each dataset, with either $p = 100$ or $p = 200$ features overall (depending on the configuration).

On average, for the configurations with $p = 100$, 33 features were selected by the feature selection method, while for configurations with $p = 200$, 50 features were selected on average. The proposed feature selection method is therefore clearly effective in substantially reducing the number of features in the dataset (without sacrificing classification accuracy; see Section 7.3.8). For both values of p , on average 19 of the 20 relevant features were selected while for $p = 100$, 14 irrelevant features were selected on average and for $p = 200$, 31 irrelevant features were selected on average. Therefore, not only is our proposed feature selection method effective in reducing the number of features without sacrificing classification accuracy, it is also very effective in selecting the *relevant* features. The correlation structure of the features (in other words, whether features were uncorrelated with each other, whether only relevant features were correlated or whether both relevant and irrelevant features were correlated) had no effect on the number of features selected.

Figure 7.2(a) and Figure 7.2(b) (on the next two pages) show the average number of relevant and irrelevant features selected for the different parameter configurations; Figure 7.2(a) shows configurations for which $p = 100$ while Figure 7.2(b) shows configurations for which $p = 200$. The line in the middle of each graph divides the graph into two areas corresponding to the 2 different values of N_{train} , while the different shaded blocks indicate different values of K . The circles identify configurations with correlations between labels.

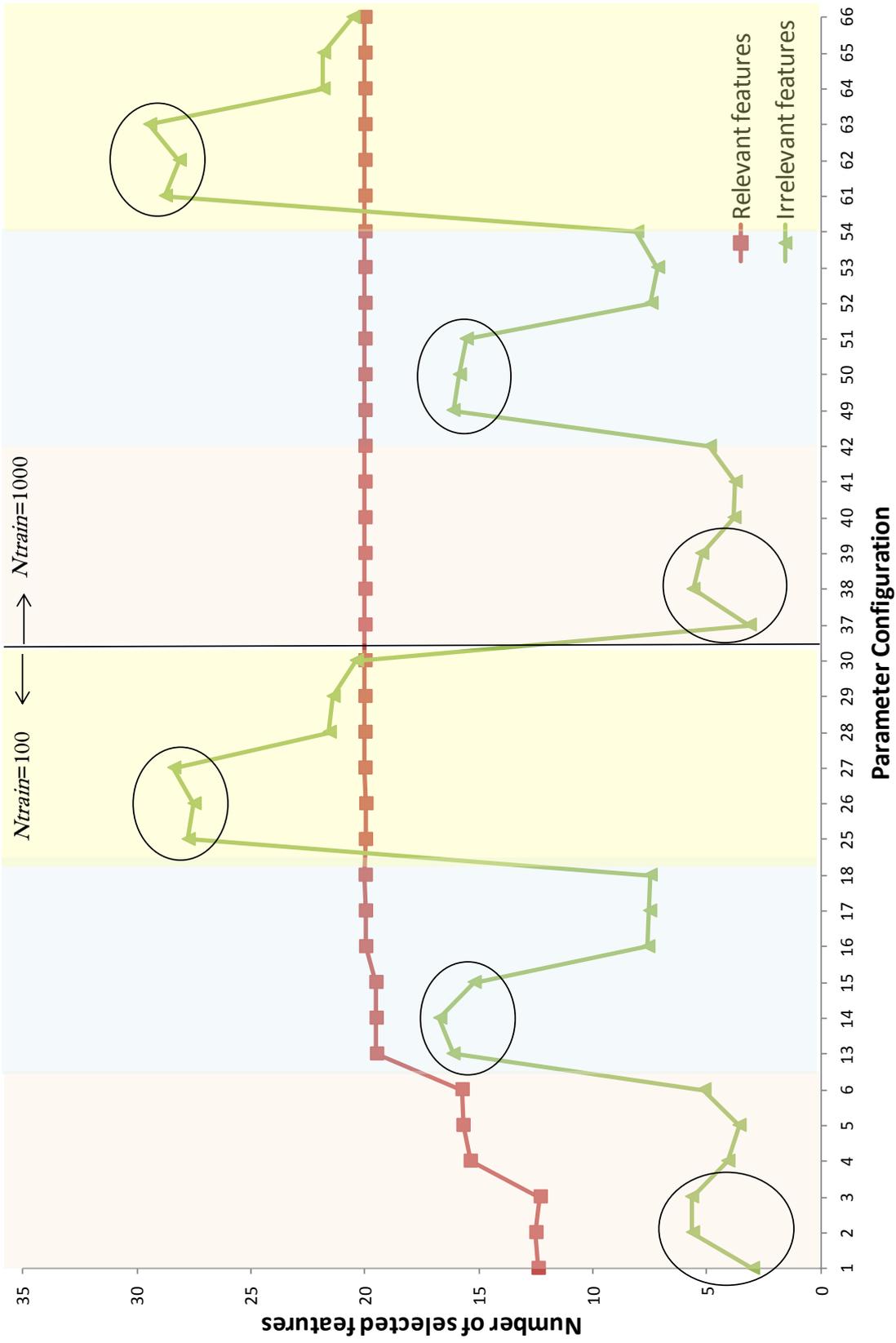


Figure 7.2(a): Number of relevant and irrelevant features selected for configurations where $p=100$

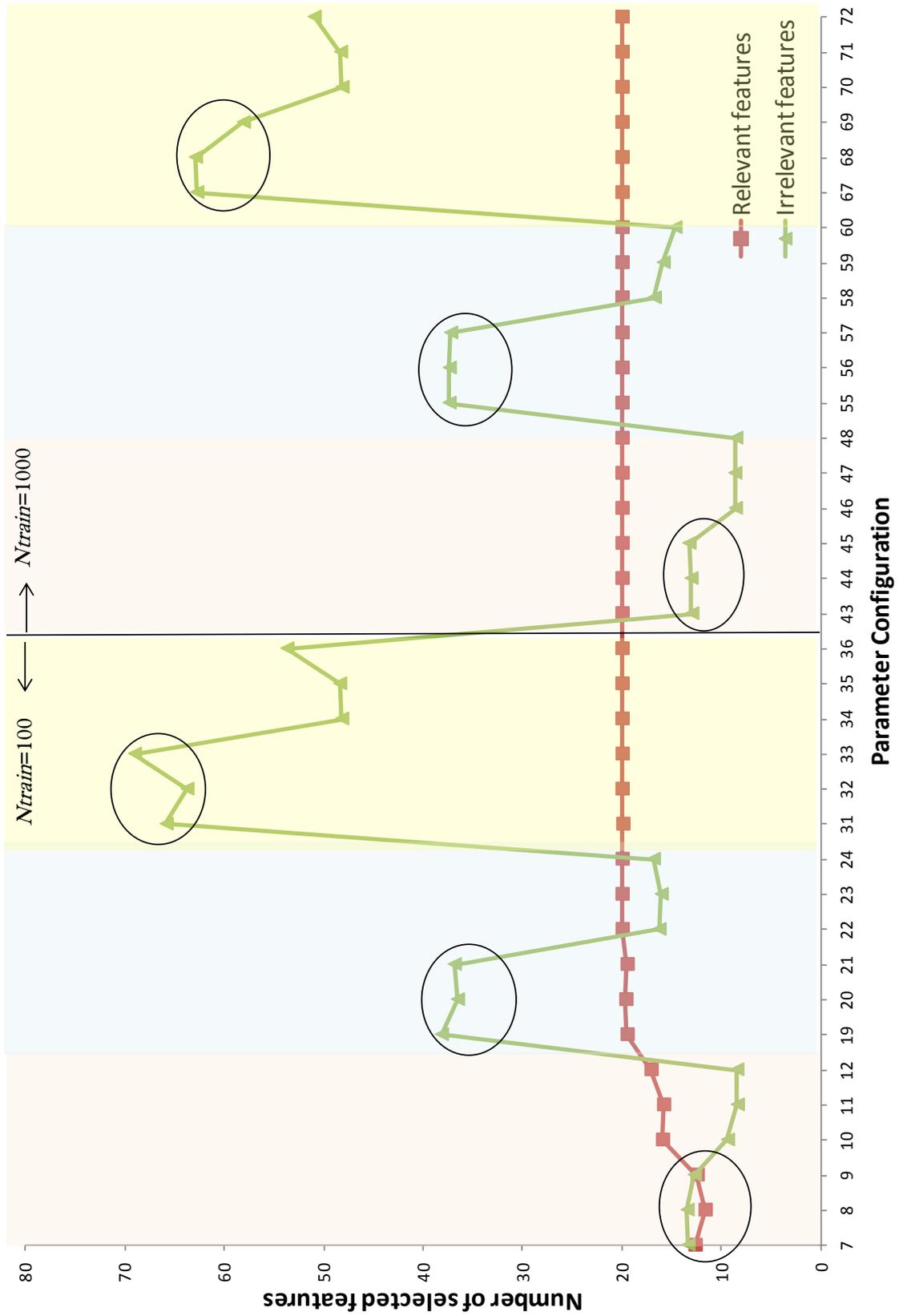


Figure 7.2(b): Number of relevant and irrelevant features selected for configurations where $p=200$

The following inferences can be drawn from these two figures:

- Except for configurations with $K = 3$ and $N_{train} = 100$, in most instances all relevant features are selected by the proposed feature selection method.
- The number of features selected increases as K increases. This implies that more irrelevant features are selected as the number of labels increases. An explanation for this phenomenon is still being sought.
- More irrelevant features are selected in configurations where there are correlations between labels than in configurations where labels are uncorrelated. There therefore appears to be interaction between the selection process and label correlations; this is somewhat unexpected, and requires further investigation, but falls outside of the scope of this thesis.

7.3.10 General remarks

The following additional points can be made about the results in this chapter:

1. Recall does not appear to be a good measure to use in our scenarios, since it does not really demonstrate the differences between the classification accuracies for different parameter configurations. This observation also emphasises the importance of choosing the correct error measure when conducting comparative multi-label classification studies, since the choice of error measure can have a significant impact on the interpretation of results.
2. The effect of the number of labels and especially the correlation structure of the labels appeared to be more complex than anticipated at the outset of this study. Increasing the number of labels decreases classification accuracy, but this result was not unexpected. However, the impact of label correlations (both in terms of classification accuracy as well as feature selection) lead to some unexpected results and there is a lot of scope for further research in this regard. One of the first steps in our subsequent research will be to refine the data simulation process in order to be able to more accurately realise the intended correlation structures in label matrices. We would also like to extend the simulation study to investigate the effect of more label correlation structures. This would also entail investigating different label densities, as these go hand in hand with the label correlations. Another step for further research would be to

investigate variations of the binary relevance algorithm, such as 2BR and BR+ (as discussed in Section 4.4.1), which are extensions of the binary relevance algorithm to incorporate label dependencies.

3. Feature selection results were truly promising, and although straightforward and relatively easy to implement, our proposed technique performed very well in being able to correctly identify relevant features. A next step could be to improve the efficacy of the technique even further by attempting to reduce the number of irrelevant features selected. In this regard, we would first experiment with the feature selection parameter α (as defined in Section 5.6.2). We also included features if they were relevant for at least one label; another step would be to experiment with “stricter” measures in this regard, for example to only include features if they are relevant for more than one label, or for, say, at least half of the labels.

7.4 Summary

In this chapter, we discussed the results of an empirical study based on the feature selection method proposed in Chapter 5, and the proposed method of simulating multi-label data discussed in Chapter 6. The effect of different parameter settings were considered, and some interesting conclusions were reached regarding the effect of these parameters on classification accuracy and feature selection.

CHAPTER 8

Application to Music Data

8.1 Introduction

As mentioned in Chapters 2 and 3, a substantial research obstacle in the field of music information retrieval is the absence of ground-truth datasets for comparing results obtained from different algorithms. There are many reasons for this (and these were discussed at length in Chapters 2 and 3), but perhaps the most important issue is the problem surrounding copyright and intellectual property, meaning that music data cannot be freely distributed.

An attempt to address this issue has been made with the annual MIREX challenges. The Music Information Retrieval Evaluation eXchange²⁷ (MIREX) is an annual community-based evaluation campaign for MIR algorithms, and has been run annually since 2005 in conjunction with the International Society for Music Information Retrieval (ISMIR). It consists of many individual sub-tasks, and tasks for

²⁷ http://www.music-ir.org/mirex/wiki/MIREX_HOME (accessed 1 July 2013)

the 2012 MIREX are listed in Table 8.1. Participants in MIREX submit their algorithms to be evaluated against a centrally held database of music collections.

Table 8.1: MIREX 2012 tasks

Task	Nr. of datasets	Nr. of submissions
Audio Classification		
- <i>Genre classification</i>	1	16
- <i>Latin genre classification</i>	1	15
- <i>Mood classification</i>	1	20
- <i>Classical composer identification</i>	1	15
Audio Tag Classification	2	9
Music Similarity	1	10
Symbolic Music Similarity	1	6
Onset Detection	1	10
Key Detection	1	6
Real-time Audio to Score Alignment	1	3
Query by Singing / Humming	3	5
Melody Extraction	6	5
Multiple F0 Estimation	1	7
Multiple F0 Tracking	2	9
Chord Estimation	2	11
Query by Tapping	3	2
Beat Tracking	3	20
Structural Segmentation	4	9
Tempo Estimation	1	4

As yet, there is no specific instrument recognition task as part of MIREX, although there is a degree of overlap between instrument recognition and the multiple F0 estimation task in MIREX.

There are also a few useful datasets that are publicly available, and these were discussed in Chapter 3, Section 3.4. While using one or more of these databases would have been an option for this study, the required pre-processing of audio as well as feature extraction would have been a very time-consuming task, and would have meant a lot of additional work outside the scope of this study. A decision was

therefore made to use a dataset that was utilised for a music instrument recognition challenge which formed part of the 2011 ISMIS conference.

The ISMIS challenge will be briefly described in Section 8.2, followed by a description of the training and test data in Section 8.3. In Section 8.4 some characteristics of the data will be illustrated by means of descriptive statistics, graphs and plots. Section 8.5 will start with an explanation of the methodology followed in our empirical study, followed by a detailed discussion of results obtained.

8.2 ISMIS contest data

Competition platforms are becoming increasingly popular as a way of solving data mining and predictive problems. Organisations can post a dataset and description of a problem online on a competition platform, and data scientists from all over the world can compete – either individually or in teams – to come up with the best solution. A well-publicised early competition of this kind was the Netflix Prize²⁸ which had a grand prize of \$1 million and ran for 3 years, with the goal of substantially improving the accuracy of predictions about a user’s film ratings based on his / her previous film ratings. The most widely known platform currently used for such competitions is Kaggle²⁹, which proclaims itself to be the world’s largest community of data scientists and has a number of data mining competitions running at any given time.

Such an open data mining contest was organised in conjunction with the 19th International Symposium on Methodologies for Intelligent Systems (ISMIS 2011). The contest consisted of two independent tasks, one of which was the automatic recognition of two instruments playing together in a given sample. The challenge was run on the open TunedIT Challenges³⁰ platform, and was an online interactive competition.

The competition attracted large interest in the data mining and MIR communities, and 292 teams (with 357 members) registered for the contest, with 150 of them

²⁸ <http://www.netflixprize.com> (accessed 18 July 2013)

²⁹ <http://www.kaggle.com> (accessed 18 July 2013)

³⁰ <http://tunedit.org/challenges> (accessed 1 July 2013)

actively participating, and submitting over 12 000 solutions in total. The contest ran from 10 January 2011 to 21 March 2011.

The winner of the instrument recognition contest was Eleftherios Spyromitros-Xioufis from the Aristotle University of Thessaloniki, Greece. Spyromitros-Xioufis achieved an error 63% lower than baseline.

Since the dataset was publicly available, was fit for the purposes of this study and also had the added convenience of having features pre-extracted, it was decided to use this dataset for the empirical work in this study – even though there may be certain drawbacks to the data.

8.3 Definition of the data

8.3.1 Training data

According to a report on the ISMIS contest (Kostek *et al.*, 2011), the original data was taken from the McGill University Master Samples database, as well as additional samples recorded in the KDD Lab at the University of North Carolina, Charlotte, USA. The goal of the contest was instrument recognition in the polyphonic case, although it was limited to two instruments playing together at a time.

There were two training datasets: one large, containing single instrument data samples, and a smaller set containing mixtures of pairs of instruments. The single instrument training dataset consisted of 114 914 samples of 19 different instruments. The mixed instrument training dataset consisted of 5 422 samples of 21 different instruments playing in pairs. In total there were 32 distinct instruments across the two training sets, with only eight instruments appearing in both datasets. However, there was a different level of taxonomy between the two training sets, as some instruments which appeared in the mixed instrument training set (for example B-Flat Trumpet and C Trumpet) appeared at an instrument family level in the single instrument training set (i.e. Trumpet). As a pre-processing step for our study, the instruments in the mixed training set were therefore relabelled to instrument family

level as well – a matter of convenience for the purposes of this study, although this would not have been possible for contest participants. In our approach, mixture data was also handled as single instrument data by duplicating each mixture observation, labelled once with each instrument respectively. For example, if a mixed instrument training case was labelled as “Piano + Guitar”, it was replaced by two training cases with exactly the same feature values but labelled as “Piano” in the first case, and labelled as “Guitar” in the other. Since the binary relevance approach was used to analyse the data, this seems to be quite acceptable. Tables 8.2 and 8.3 show the instrument distribution in the two training datasets before and after pre-processing. In the case of the mixed instruments training data, the column percentages will sum to 200%, since each training case has two labels. For example, the first entry of Table 8.2 shows that there are 634 training cases for which Accordion is one of the two labels, which means that 11.7% of the 5 422 mixed instrument training cases have Accordion as one of its labels.

Table 8.2: Original training datasets

	Single instrument training set		Mixed pairs training set	
	<i>Nr. of samples</i>	<i>%</i>	<i>Nr. of samples</i>	<i>%</i>
Accordion	1 460	1.3	634	11.7
Acoustic Bass	0	0.0	417	7.7
Alto Saxophone	0	0.0	337	6.2
Baritone Saxophone	0	0.0	530	9.8
Bassoon	5 763	5.0	0	0.0
Bass Saxophone	0	0.0	503	9.3
B flat Clarinet	0	0.0	567	10.5
B flat Trumpet	0	0.0	412	7.6
Cello	4 964	4.3	332	6.1
Clarinet	9 492	8.3	0	0.0
C Trumpet	0	0.0	1 033	19.1
Double Bass	3 849	3.3	634	11.7
Electric Guitar	0	0.0	590	10.9
English Horn	1 672	1.5	0	0.0
Flute	7 408	6.4	0	0.0
French Horn	2 482	2.2	0	0.0
Guitar	34 723	30.2	0	0.0
Marimba	0	0.0	590	10.9
Oboe	1 643	1.4	332	6.1
Piano	6 144	5.3	417	7.7
Piccolo	2 874	2.5	0	0.0
Saxophone	4 388	3.8	0	0.0
Soprano Saxophone	0	0.0	329	6.1
Synth Bass	918	0.8	0	0.0
Tenor Saxophone	0	0.0	934	17.2
Tenor Trombone	0	0.0	666	12.3
Trombone	4 503	3.9	0	0.0
Trumpet	11 152	9.7	0	0.0
Tuba	3 463	3.0	522	9.6
Vibraphone	0	0.0	249	4.6
Viola	3 006	2.6	567	10.5
Violin	5 010	4.4	249	4.6

Table 8.3: Training datasets after pre-processing
(aggregation to instrument family level)

	Single instrument training set		Mixed pairs training set	
	<i>Nr. of samples</i>	<i>%</i>	<i>Nr. of samples</i>	<i>%</i>
Accordion	1 460	1.3	634	11.7
Acoustic Bass	0	0.0	417	7.7
Bassoon	5 763	5.0	0	0.0
Cello	4 964	4.3	332	6.1
Clarinet	9 492	8.3	567	10.5
Double Bass	3 849	3.3	634	11.7
Electric Guitar	0	0.0	590	10.9
English Horn	1 672	1.5	0	0.0
Flute	7 408	6.4	0	0.0
French Horn	2 482	2.2	0	0.0
Guitar	34 723	30.2	0	0.0
Marimba	0	0.0	590	10.9
Oboe	1 643	1.4	332	6.1
Piano	6 144	5.3	417	7.7
Piccolo	2 874	2.5	0	0.0
Saxophone	4 388	3.8	2 633	48.6
Synth Bass	918	0.8	0	0.0
Trombone	4 503	3.9	666	12.3
Trumpet	11 152	9.7	1 445	26.7
Tuba	3 463	3.0	522	9.6
Vibraphone	0	0.0	249	4.6
Viola	3 006	2.6	567	10.5
Violin	5 010	4.4	249	4.6

It should be noted that Electric Guitar and Marimba data samples do not appear in the single instrument training set, while in the mixture training set they only appear with each other, and not with any other instrument. This means that it will be impossible to train an algorithm to correctly identify these instruments based on the training data provided.

8.3.2 Test data

The test set contained only mixture data, and the contest organisers revealed that the test and training mixture sets contained different pairs of instruments; in other words, the pairs of instruments playing together in the test data do not appear together in the training data. Furthermore, the organisers also stated that not all instruments from the training data must also appear in the test set, and that there may be instruments from the test set that only appear in the single instruments training set. There were 14 662 test samples.

In the context of the ISMIS contest, the distribution of the test data was not known (although some participants submitted “dummy” predictions during the course of the competition to get an indication of the distribution of instruments in the test dataset). However, the distribution of the test data is shown in Table 8.4 below, to provide some indication as to how the different instruments are represented in the test data. Percentages sum to 200% once again, as explained before. (Please note though that no knowledge of the test data was used in training the algorithms discussed in this chapter.)

Table 8.4: Test dataset
(aggregation to instrument family level)

	<i>Nr. of samples</i>	<i>%</i>
Accordion	2 427	16.6
Cello	694	4.7
Clarinet	3 661	25.0
Double Bass	1 338	9.1
Electric Guitar	4 222	28.8
French Horn	425	2.9
Marimba	377	2.6
Oboe	3 247	22.1
Piano	635	4.3
Saxophone	3 721	25.4
Synth Bass	635	4.3
Trombone	1 248	8.5
Trumpet	2 048	14.0
Vibraphone	594	4.1
Viola	815	5.6
Violin	3 237	22.1

There were only 16 instruments present in the test dataset, compared to 23 in the training sets.

8.3.3 Features

There were 123 pre-computed audio features in the training and test datasets. Audio features were described and defined in Chapter 2, Section 2.7, so here the features will only be listed. The features provided were:

- **Flatness coefficients:** 33 flatness coefficients (*BandsCoef1-33*) as well as a sum (*bandsCoefSum*) (see page 37)
- **MFCC coefficients:** 13 MFCC coefficients (*MFCC1-13*) (pages 34-35)
- **Harmonic peaks:** 28 harmonic peaks (*HamoPk1-28*) (page 38)

- **Spectrum projection coefficients:** 33 projection coefficients (*Prj1-33*) as well as the minimum and maximum values (*prjmin*, *prjmax*), sum (*prjsum*), distribution (*prjdis*) and standard deviation (*prjstd*) (page 37)
- **Spectral centroid measures:** MPEG-7 Harmonic Spectral Centroid (*SpecCentroid*) and MPEG-7 Audio Spectrum Centroid (*LogSpecCentroid*) (pages 31-32)
- **Spectral spread measures:** MPEG-7 Harmonic Spectral Spread (*SpecSpread*) and MPEG-7 Audio Spectrum Spread (*LogSpecSpread*) (page 33)
- **Flux** (page 37)
- **Rolloff** (page 36)
- **Zero Crossing Rate** (*ZeroCrossings*) (pages 35-36)
- **Energy** (page 35)
- **Log Attack Time** (*LogAttackTime*) (pages 38-39)
- **Temporal centroid** (*temporalCentroid*) (page 31)

Some additional information (such as pitch) was also provided for the single instrument training set, but this was ignored for the purposes of this study.

The Flux feature was deliberately discarded in this study, since it contained extreme values which caused problems for the classification algorithms.

8.4 Data characteristics

The aim of this section is to illustrate, with the help of plots and graphs, that the dataset under consideration showed some inherent complexities which implied that feature selection as well as classification of instruments would not be a trivial task. This is illustrated in three parts:

1. Firstly, only single instrument data was considered. Furthermore, only data for a single pitch was considered in order for instruments to be more objectively comparable.

2. Secondly, data for two single instruments (Accordion and Double Bass) were considered together with the mixture data for these two instruments.
3. Lastly, an attempt at dimension reduction was made through principal component analysis (PCA).

8.4.1 Single instrument, single pitch

The single instrument training dataset consisted of 114 914 samples representing 19 different instruments. This data also represented different pitch values for the instruments. Since pitch affects feature values, a single pitch (A) was isolated to enable better direct comparisons between features measured for different instruments. Averages of the different feature values were calculated for each instrument and then plotted by feature and instrument. Features were split into groups to ensure more legible plots could be drawn. Boxplots were also drawn for some features. These plots can be found on the following pages (Figures 8.1 to 8.10).

The following are examples of observations from these plots:

- Some features seem to discriminate better between instruments than others. For instance, the 11th MFCC does not appear to separate well between Cello, Clarinet and Double Bass; however, the 3rd MFCC appears to separate these 3 instruments fairly well (Figures 8.2(a) and 8.2(b)). Similarly, there is no clear difference between Trombone and Trumpet in terms of the 4th harmonic peak, while there is a clear difference in terms of the 15th harmonic peak, where the Trombone appears to have the value zero in most instances (Figures 8.8(a) and 8.8(b)). A third example is French Horn and Flute in terms of Energy (Figure 8.10(c)) and Rolloff (Figure 8.10(b)).
- Some instruments appear to have similar profiles in terms of some or all features. For instance, the Violin and Viola have very similar profiles in terms of flatness coefficients (Figure 8.3(a)), while their MFCC profiles show more distinct differences (Figure 8.1(a)). Similarly, the Piccolo and English Horn have very similar harmonic peak profiles (Figure 8.7(b)), yet they clearly differ in terms of MFCCs (Figure 8.1(b)). This implies that when feature selection is

performed, it might be sensible to do selection at an instrument level in order to capture such differences.

- Projection coefficients appear to have fairly limited usefulness for separating instruments – the exception being Synth Bass (which is the only instrument with a distinctly different average value for the second projection coefficient, see Figure 8.5(d)) and string instruments (where the projection coefficients appear to show fairly good discriminatory power, see Figure 8.5(a)). (The first projection coefficient was excluded from Figures 8.5(a)-(d) since it had an average value of close to negative 1 for all instruments; the first projection coefficient is plotted in Figure 8.6(c)).
- It should be kept in mind that these plots are for single instruments only, and the fact that some features appear to display good discriminative power in terms of individual instruments does not necessarily imply that they will be able to discriminate well in the case of mixtures of instruments.
- Even though these plots illustrate the profiles for single instruments and not mixtures, there is clearly a strong case to be made for feature selection in instrument recognition problems.

Figure 8.1(a)-(d): MFCCs – averages by single instrument (pitch A)

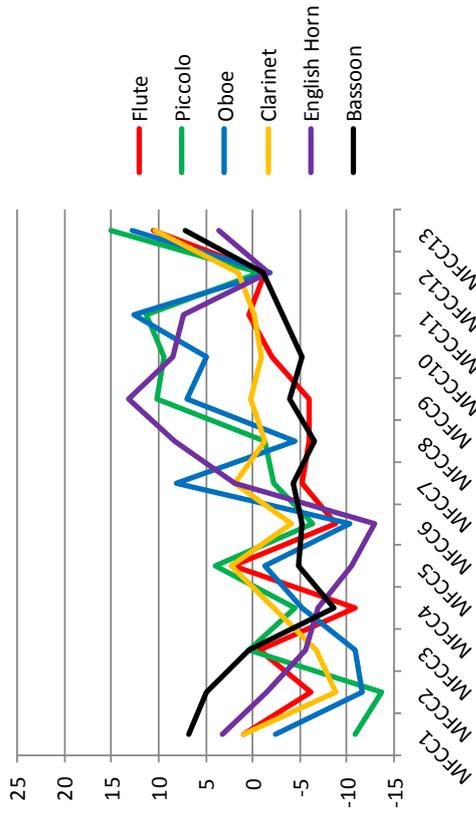


Figure 8.1(b): MFCC averages for woodwinds

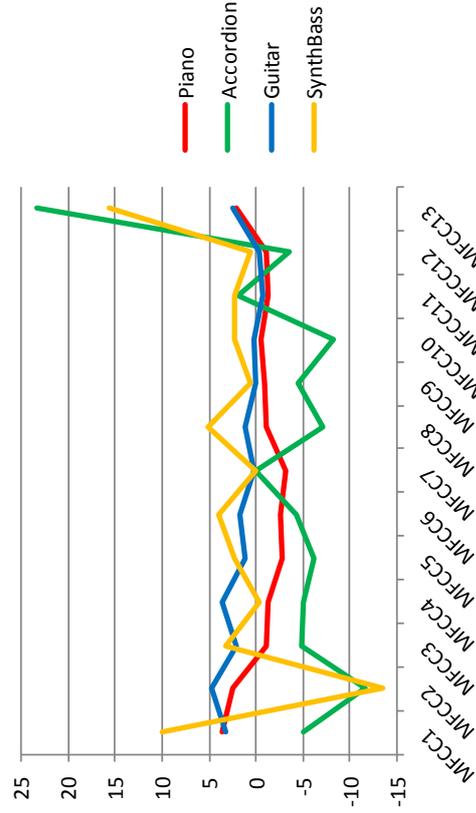


Figure 8.1(d): MFCC averages for other instruments

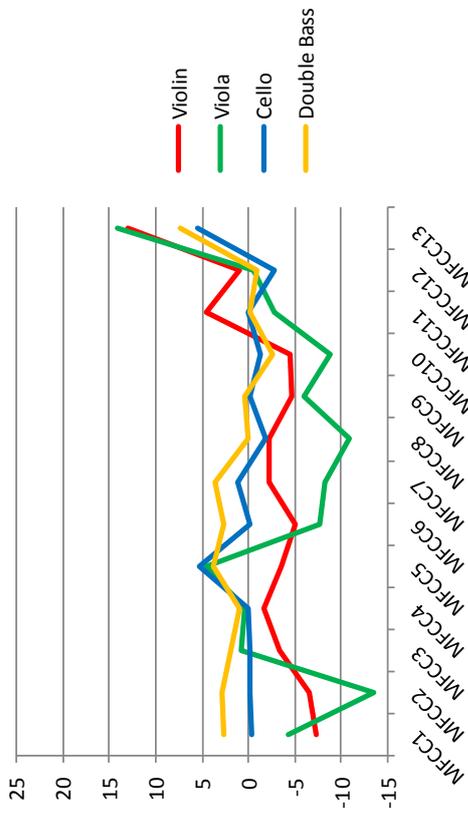


Figure 8.1(a): MFCC averages for strings

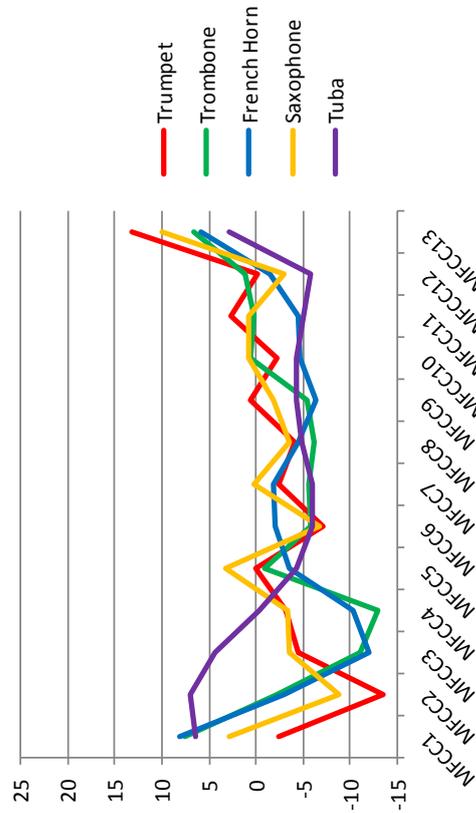


Figure 8.1(c): MFCC averages for brass

Figure 8.2(a): Boxplot per instrument for MFCC3

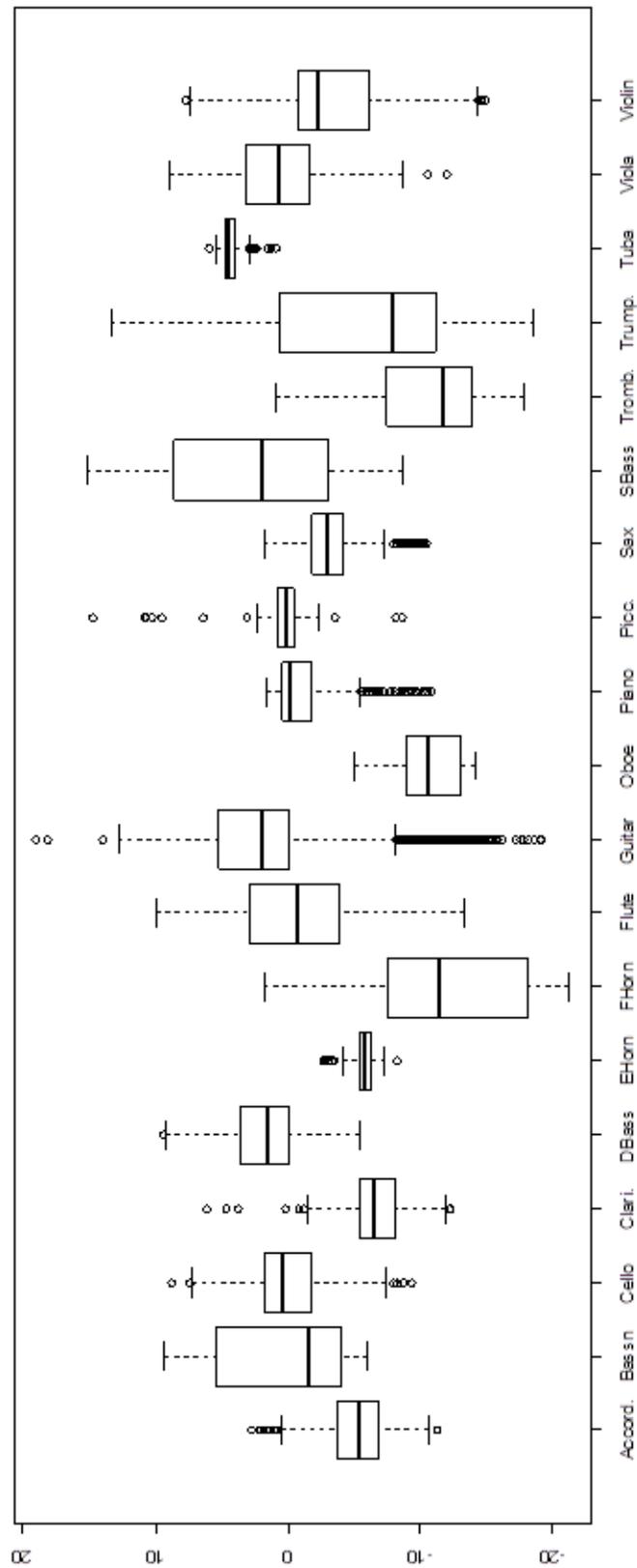


Figure 8.2(b): Boxplot per instrument for MFCC11

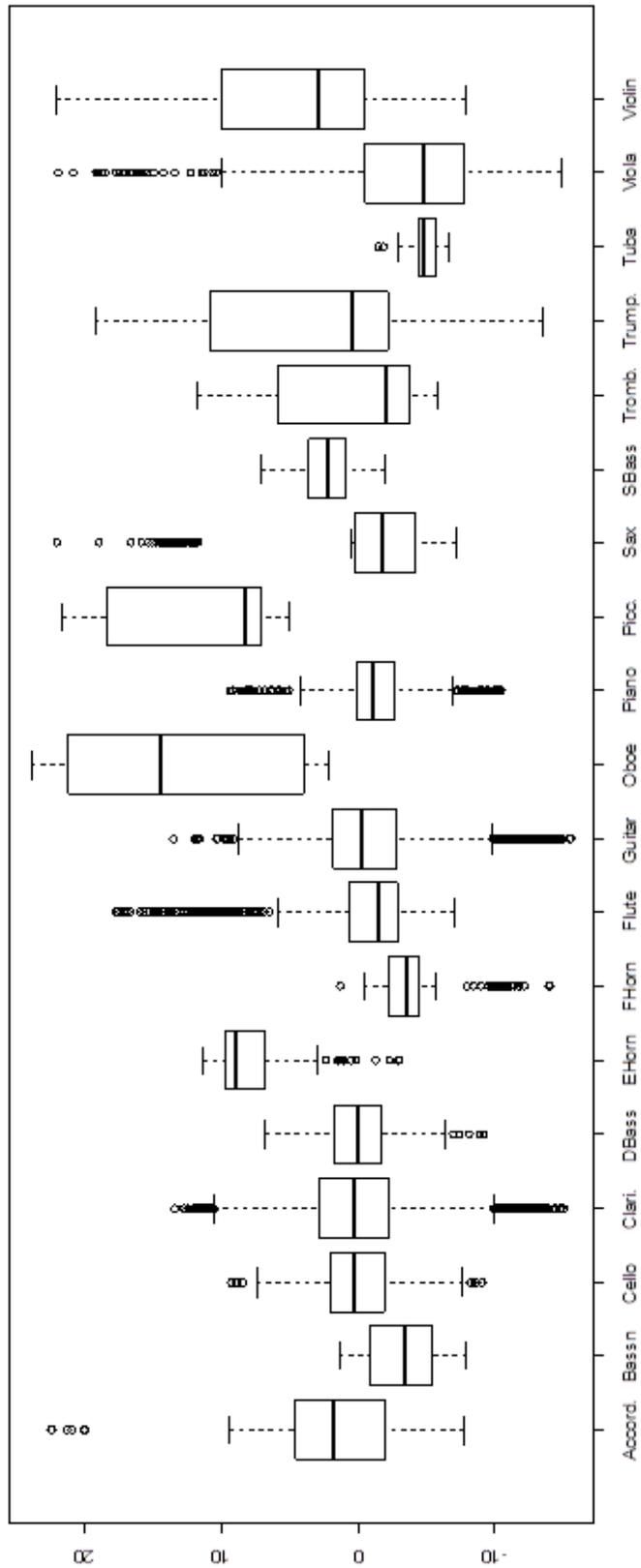


Figure 8.3(a)-(d): Flatness coefficients – averages by single instrument (pitch A)

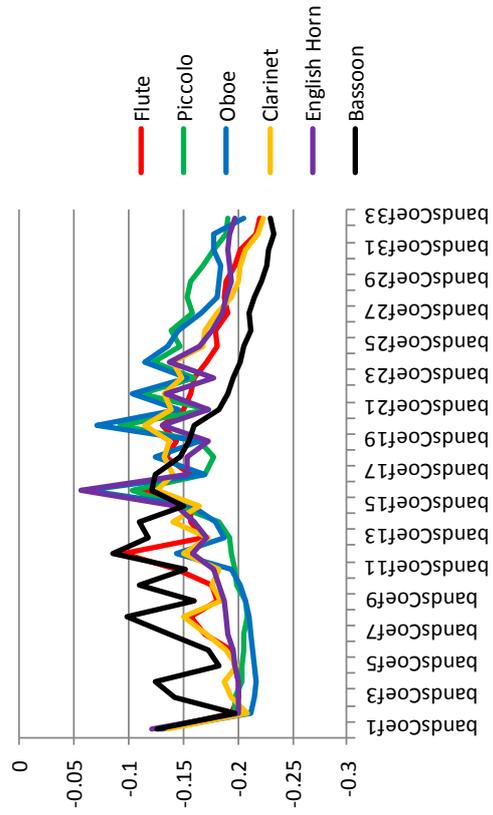


Figure 8.3(b): Flatness coefficient averages for woodwinds

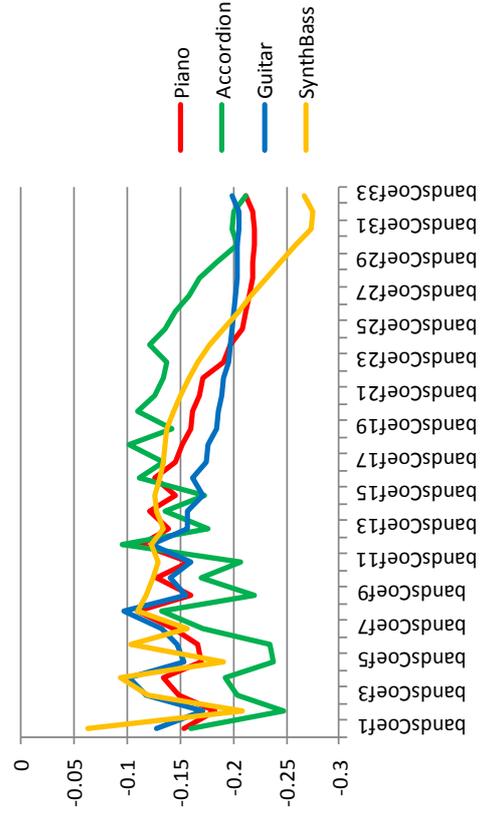


Figure 8.3(d): Flatness coefficient averages for other instruments

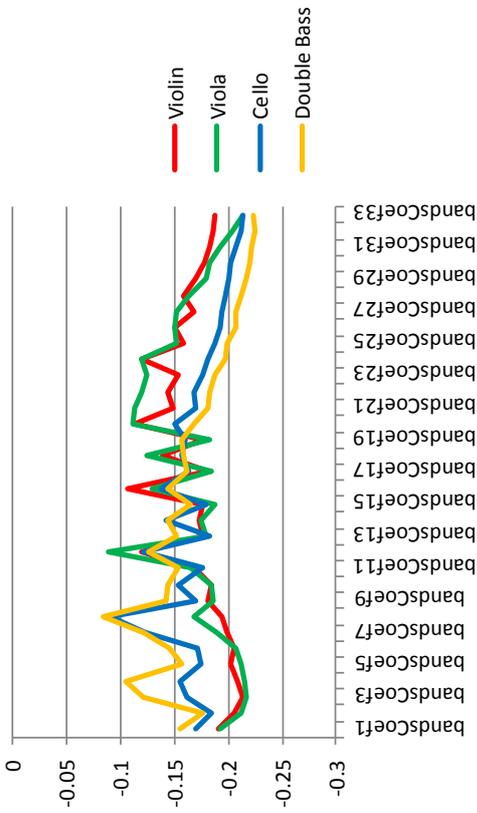


Figure 8.3(a): Flatness coefficient averages for strings

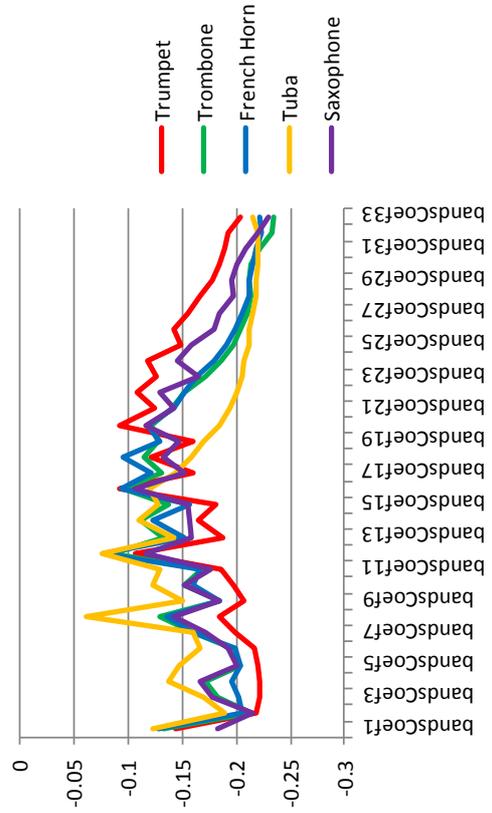


Figure 8.3(c): Flatness coefficient averages for brass

Figure 8.4(a): Boxplot per instrument for 25th flatness coefficient (bandsCoef25)

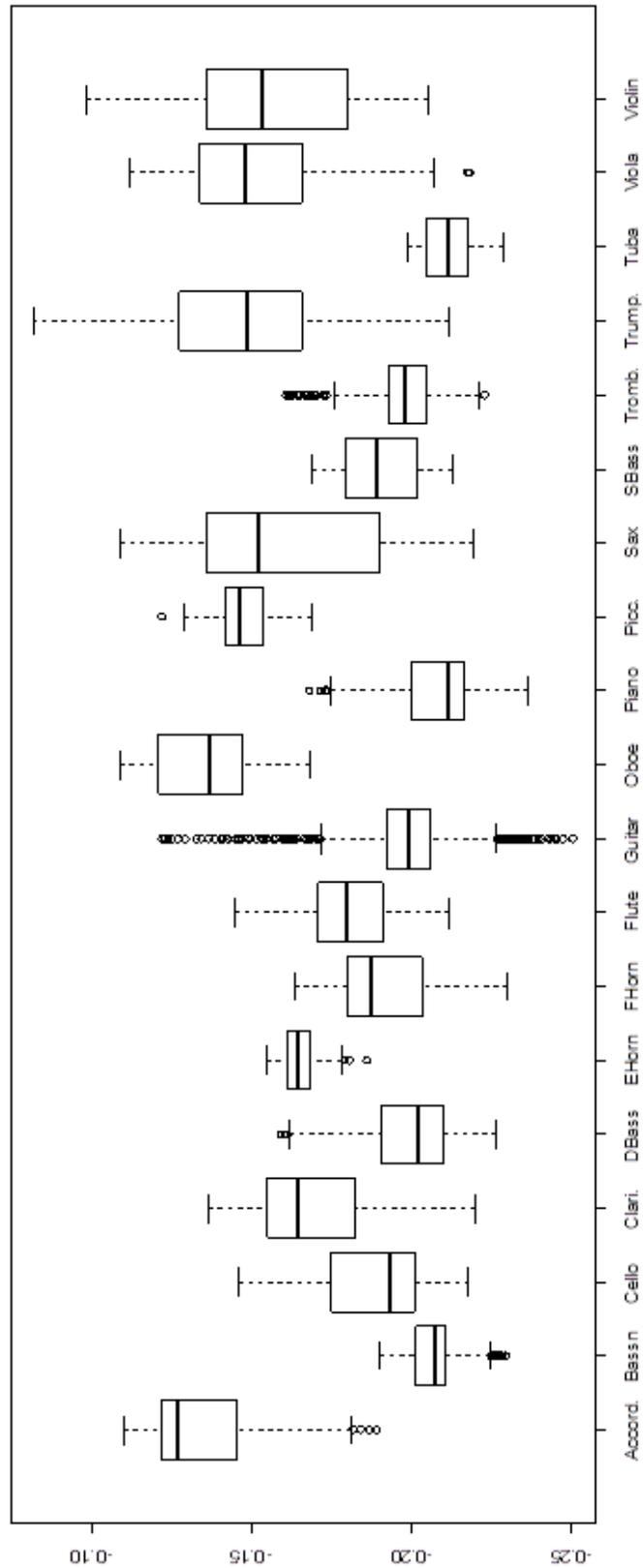


Figure 8.4(b): Boxplot per instrument for 10th flatness coefficient (bandsCoef10)

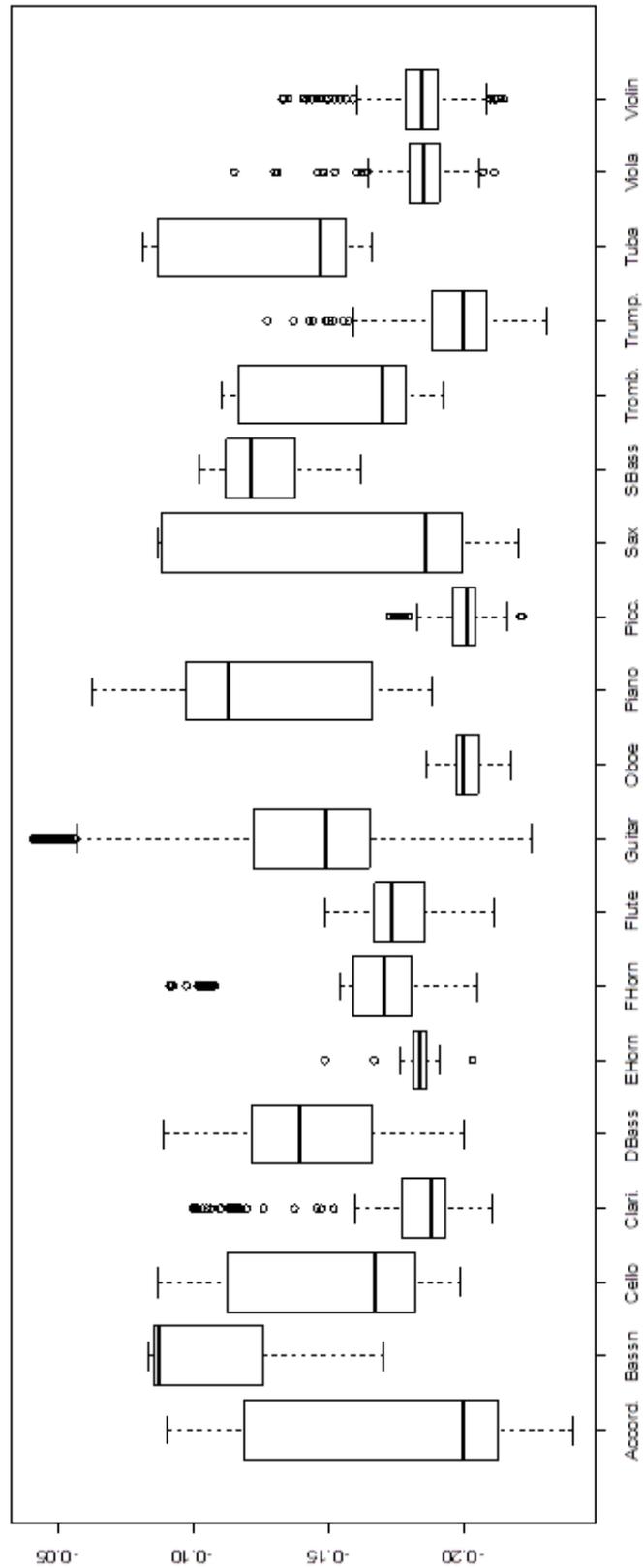


Figure 8.5(a)-(d): Projection coefficients (excluding first one) – averages by single instrument (pitch A)

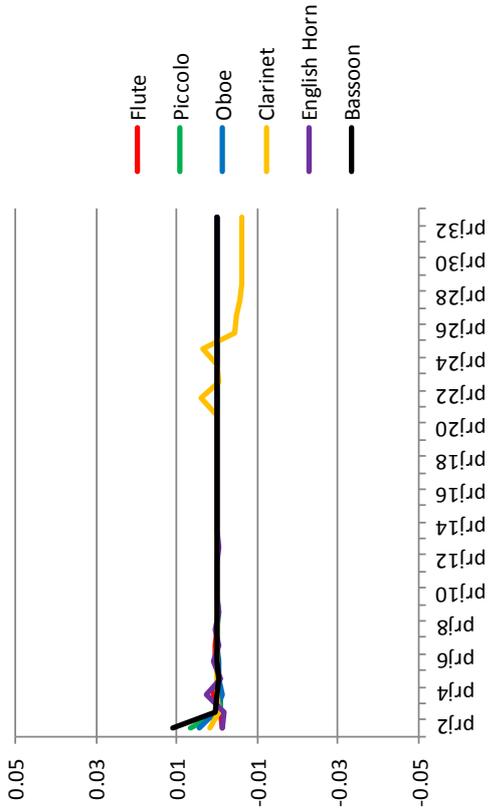


Figure 8.5(b): Projection coefficient averages for woodwinds

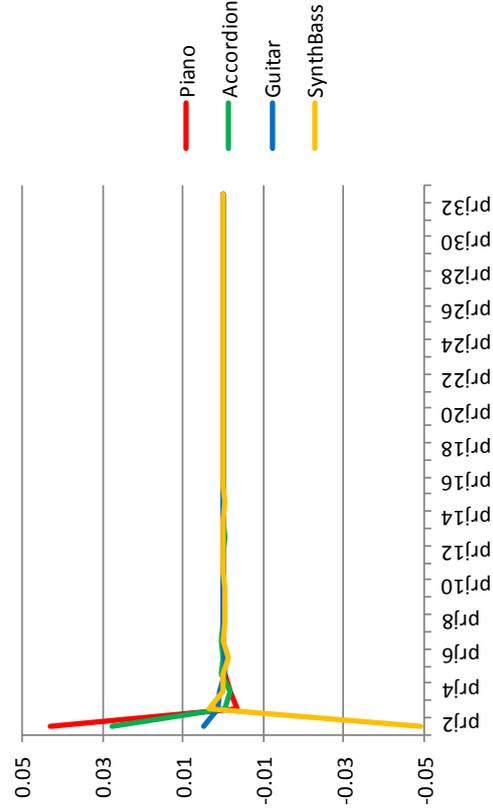


Figure 8.5(d): Projection coefficient averages for other instruments

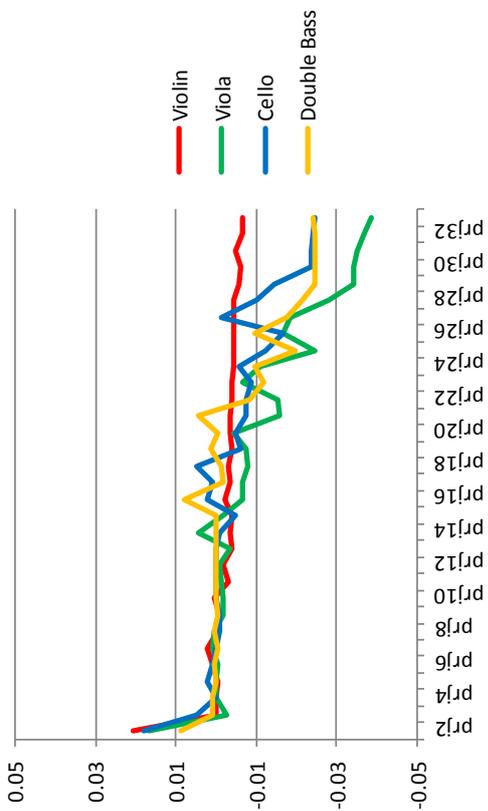


Figure 8.5(a): Projection coefficient averages for strings

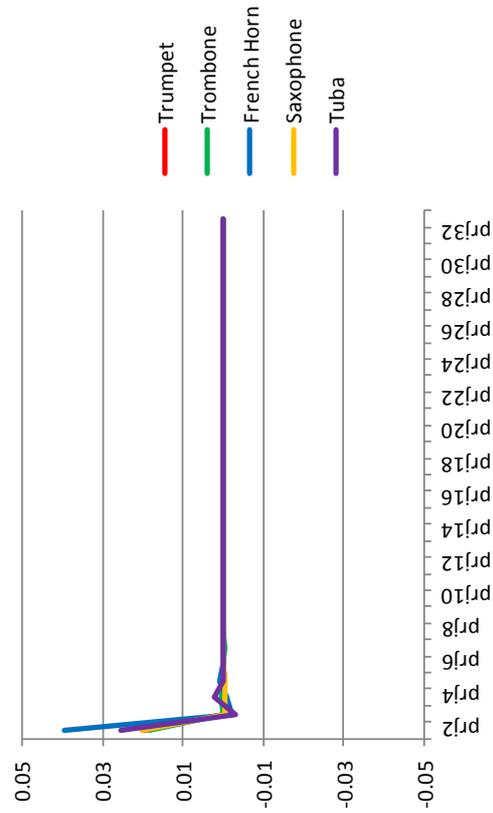


Figure 8.5(c): Projection coefficient averages for brass

Figure 8.6(a): Boxplot per instrument for 2nd projection coefficient (prj2)

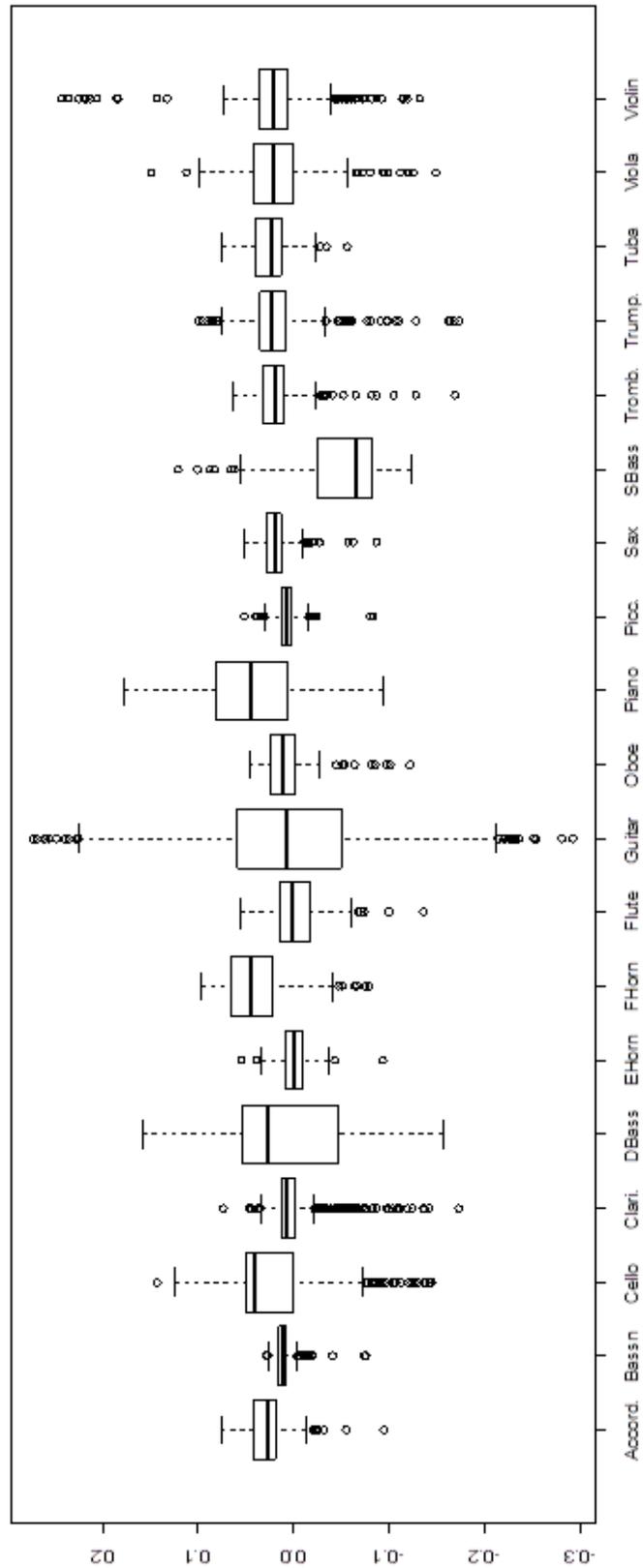


Figure 8.6(b): Boxplot per instrument for 8th projection coefficient (*prj8*)

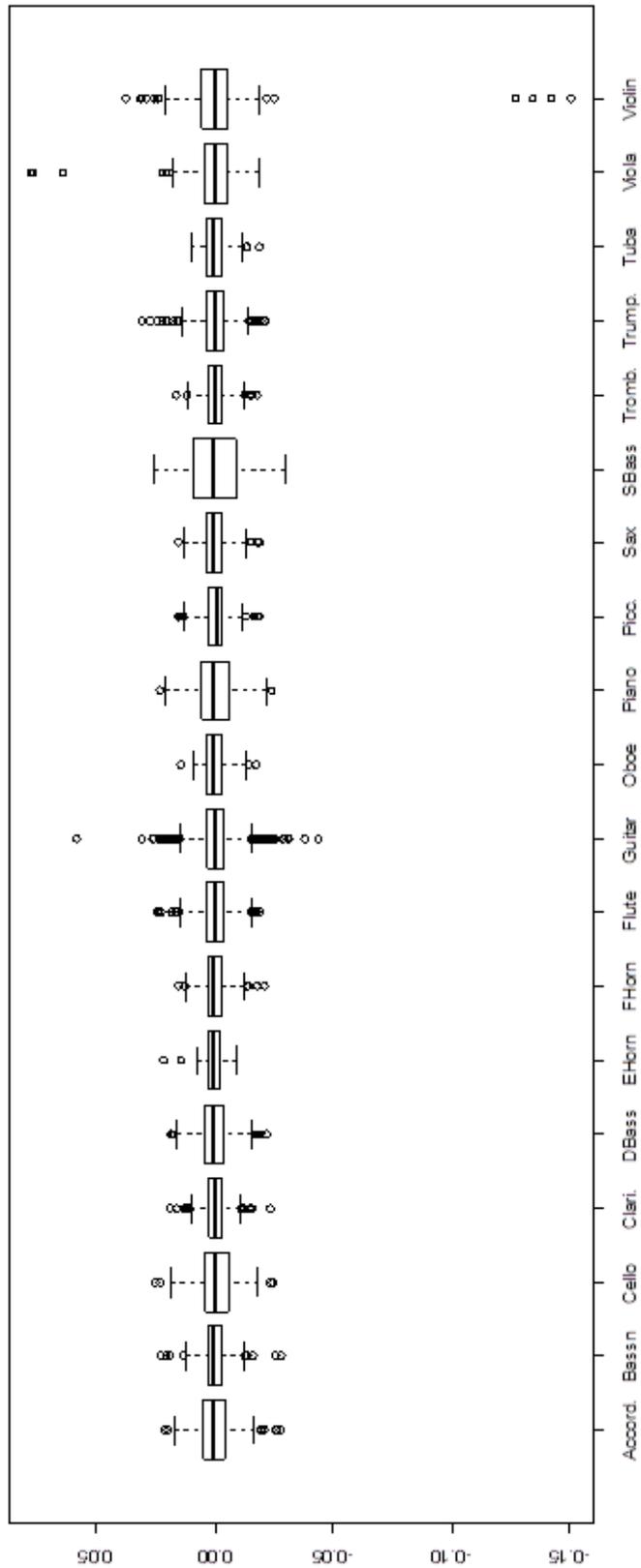


Figure 8.6(c): Boxplot per instrument for 1st projection coefficient (*pr1*)

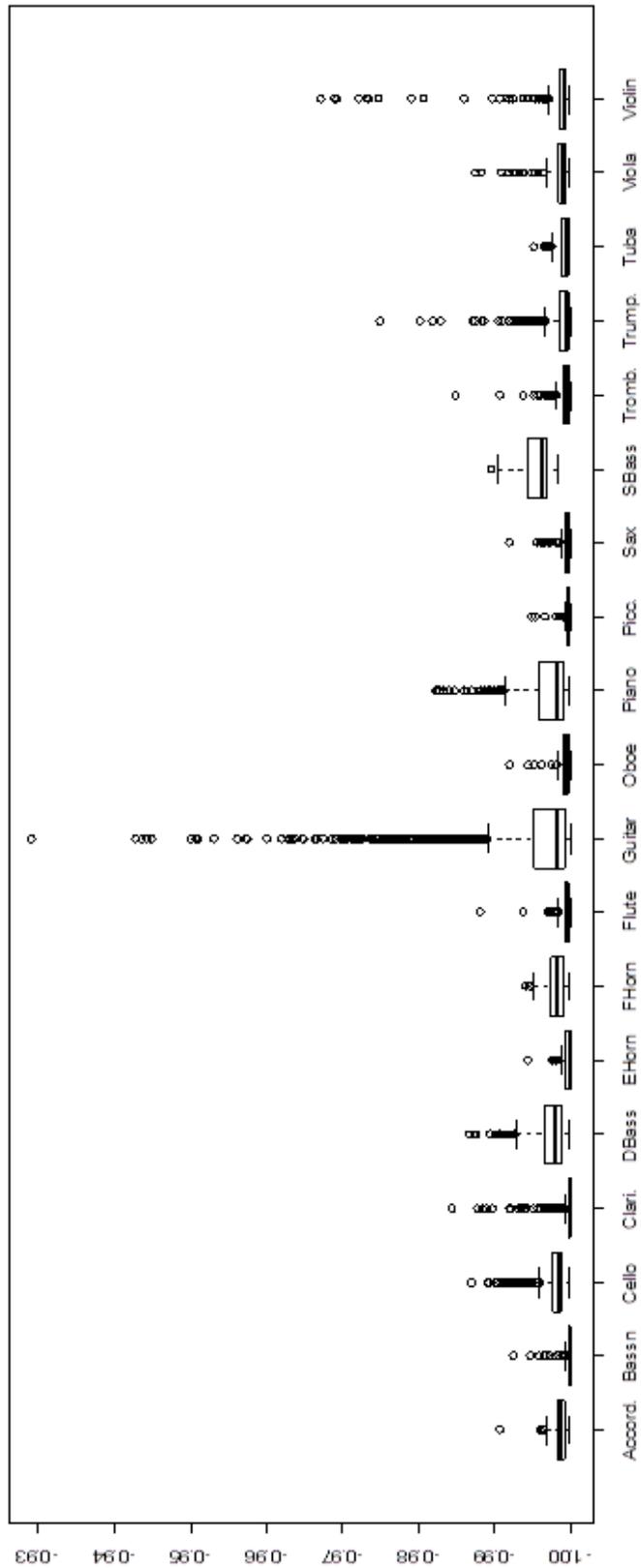


Figure 8.7(a)-(d): Harmonic peaks – averages by single instrument (pitch A)

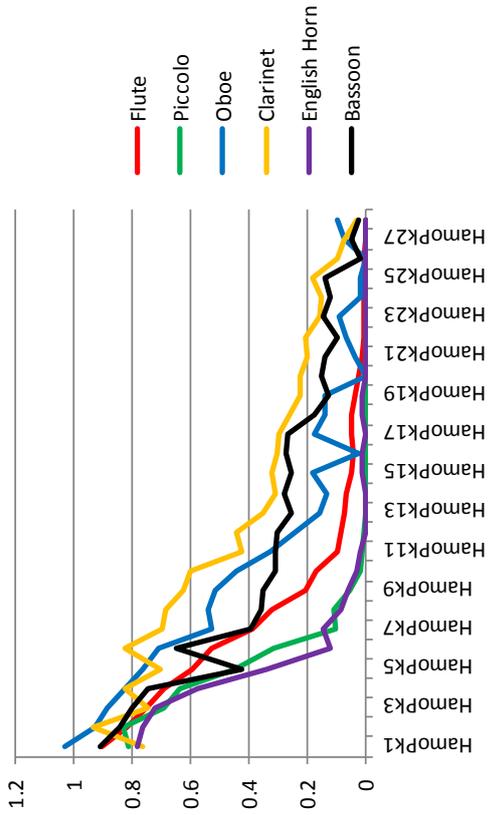


Figure 8.7(b): Harmonic peak averages for woodwinds

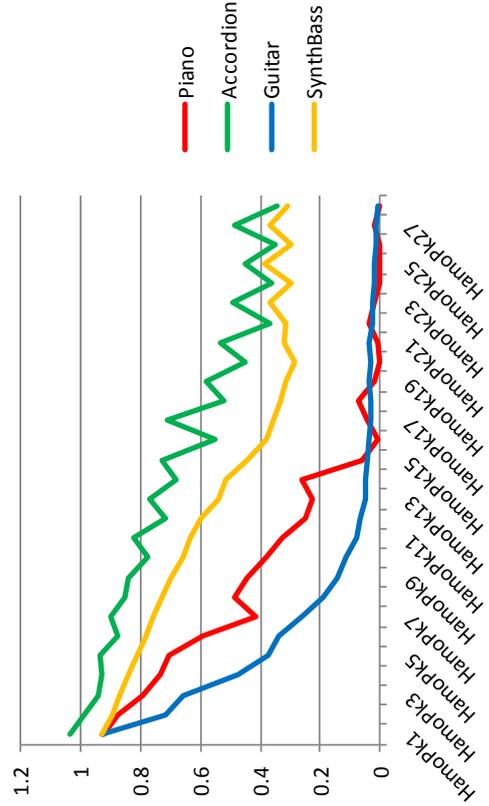


Figure 8.7(d): Harmonic peak averages for other instruments

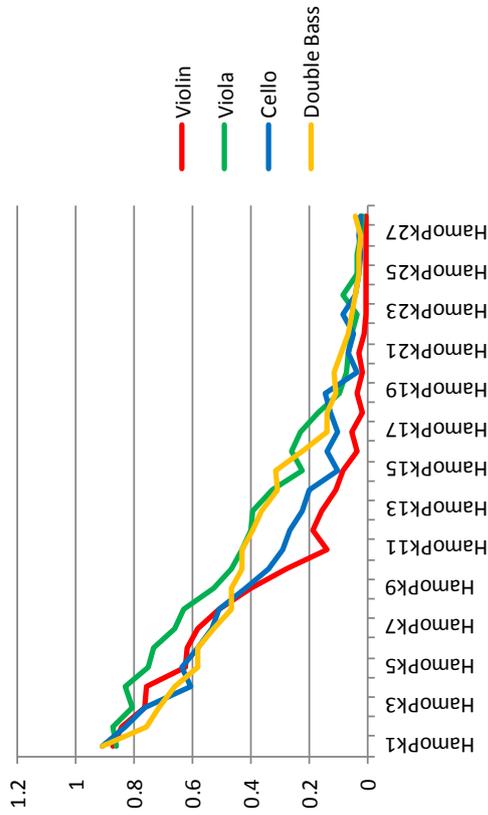


Figure 8.7(a): Harmonic peak averages for strings

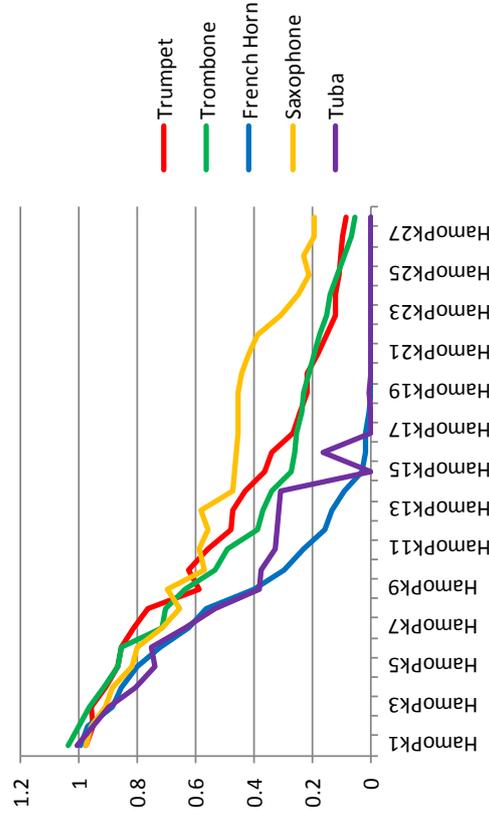


Figure 8.7(c): Harmonic peak averages for brass

Figure 8.8(a): Boxplot per instrument for 15th harmonic peak (HamoPk15)

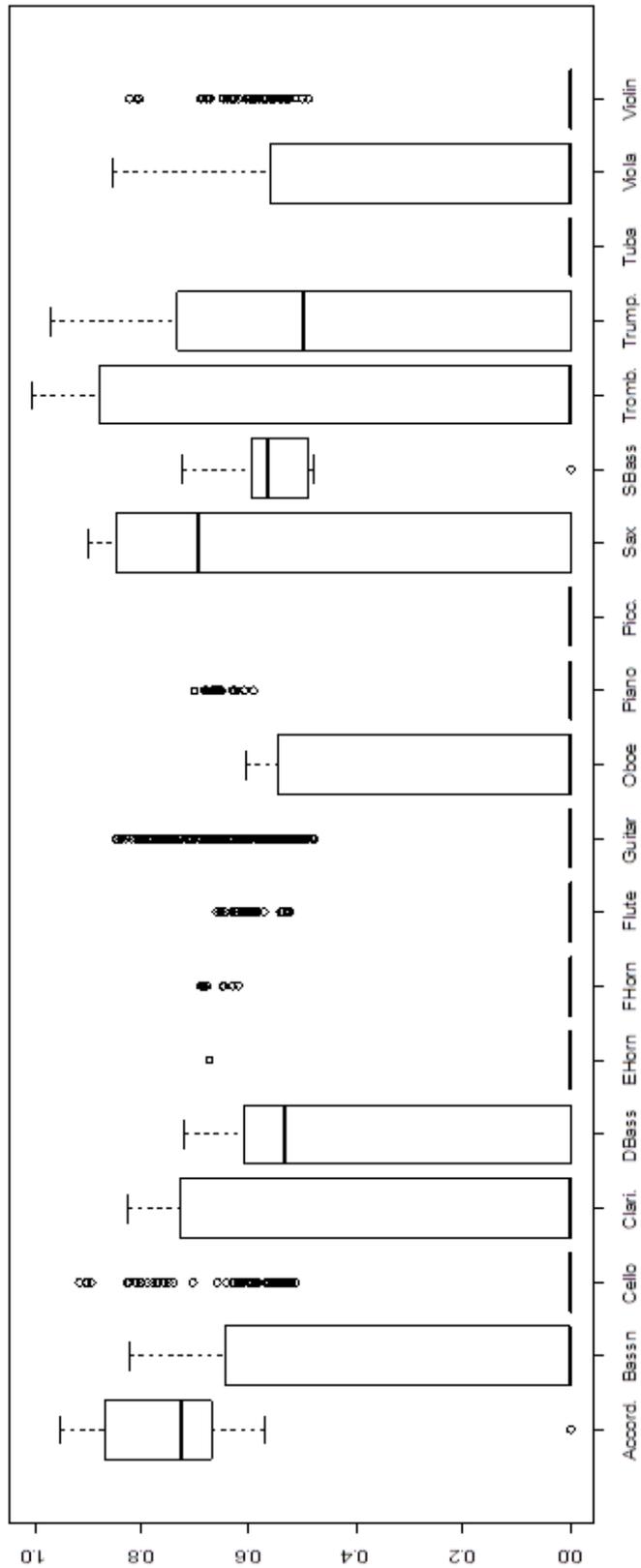


Figure 8.8(b): Boxplot per instrument for 4th harmonic peak (HamoPk4)

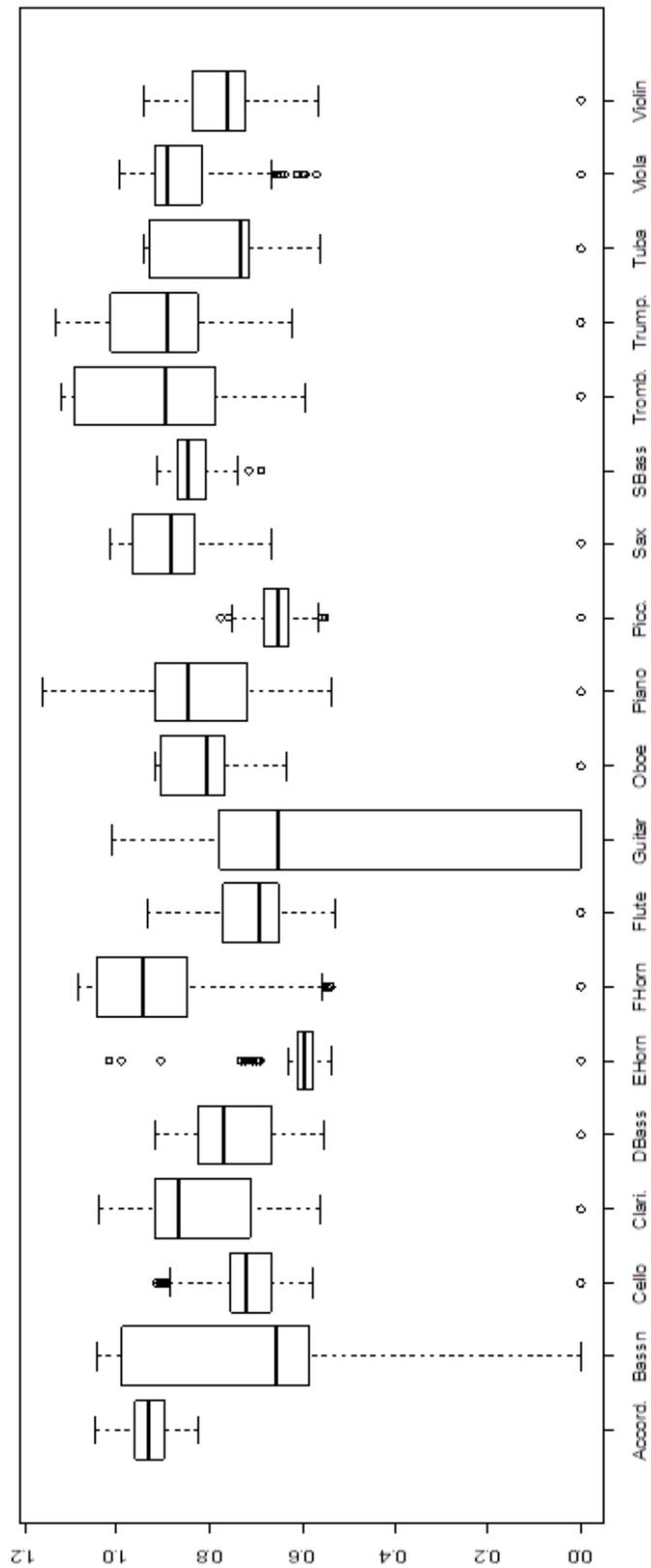


Figure 8.9(a): Other features – averages by single instrument (pitch A)

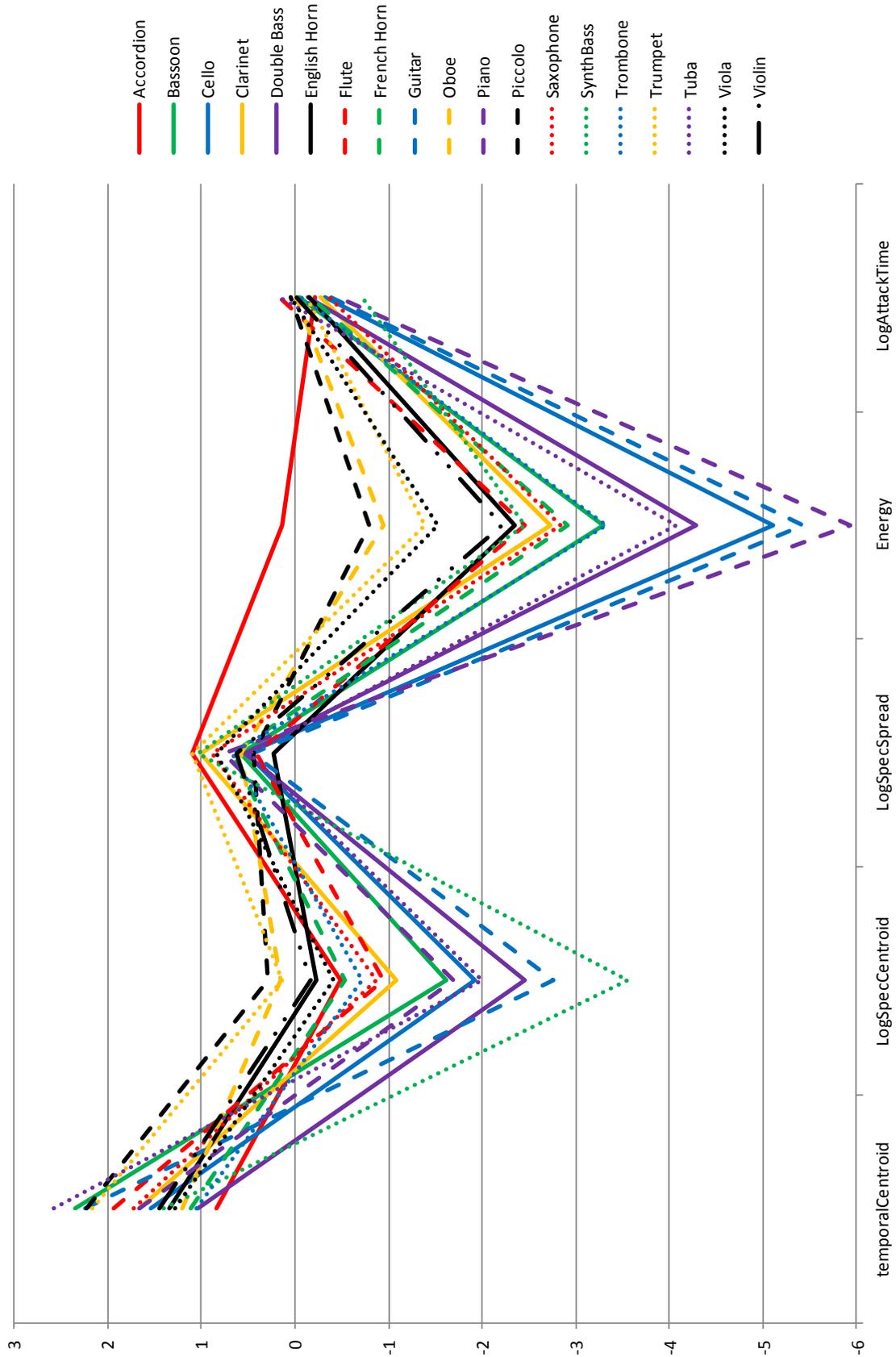


Figure 8.9(b): Other features – averages by single instrument (pitch A)

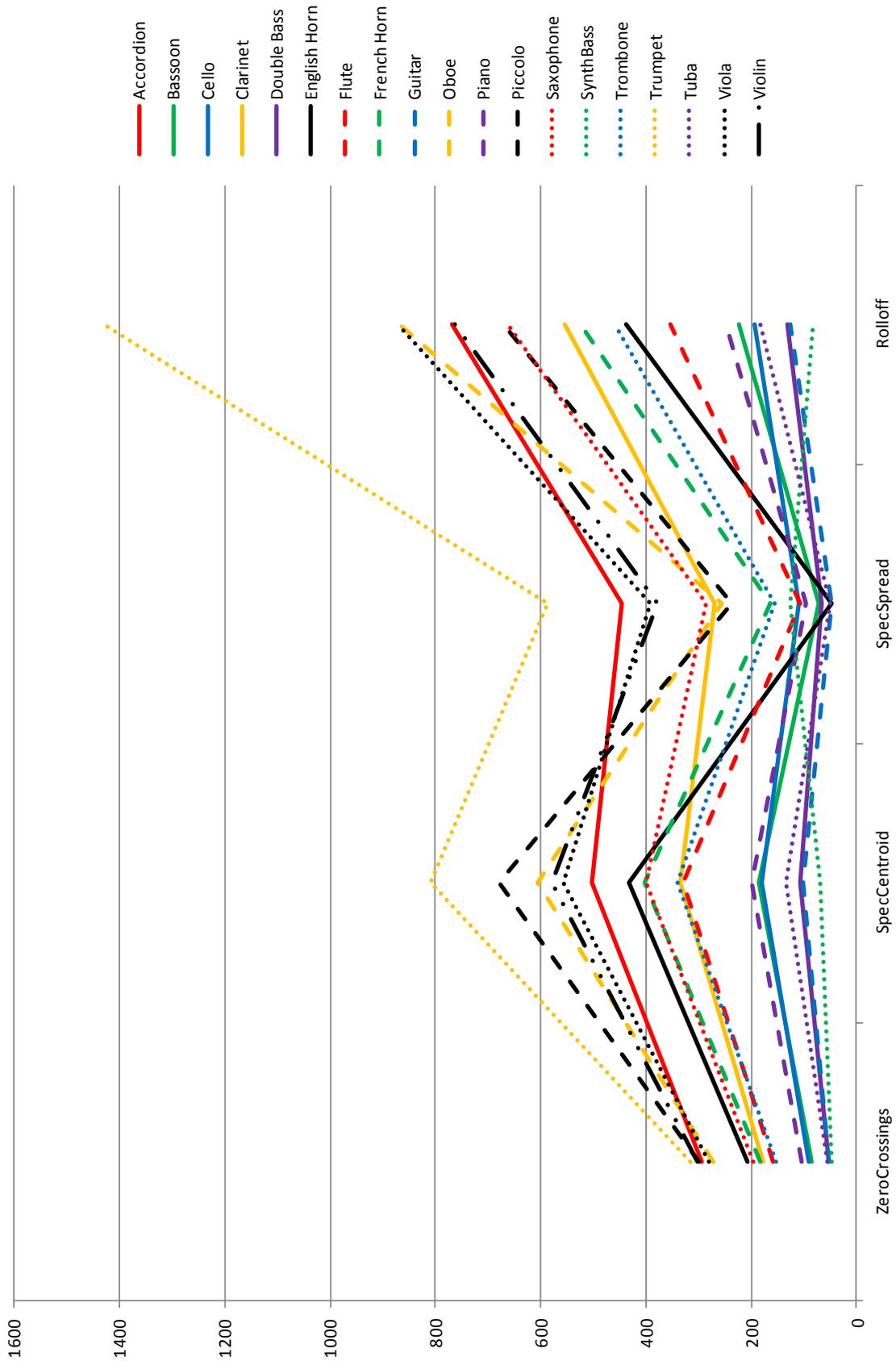


Figure 8.10(a): Boxplot per instrument for spectral spread (*SpecSpread*)

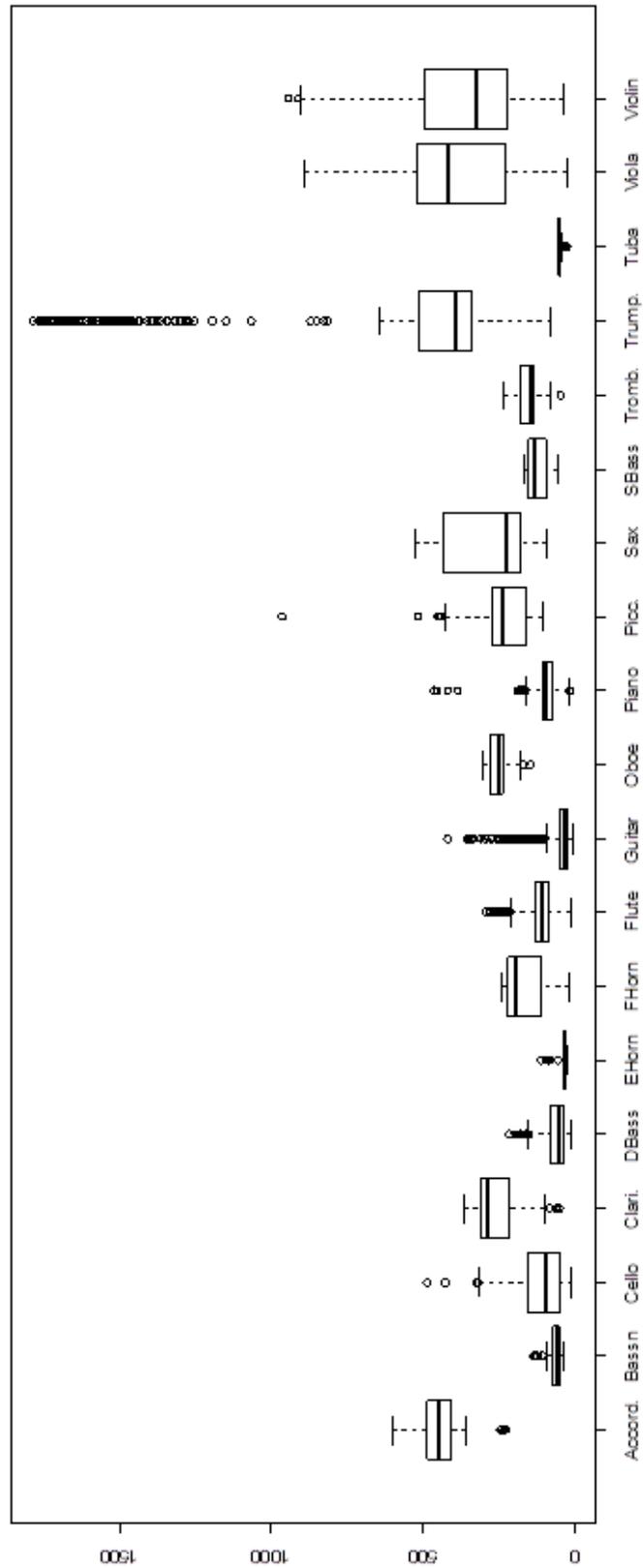


Figure 8.10(b): Boxplot per instrument for rolloff

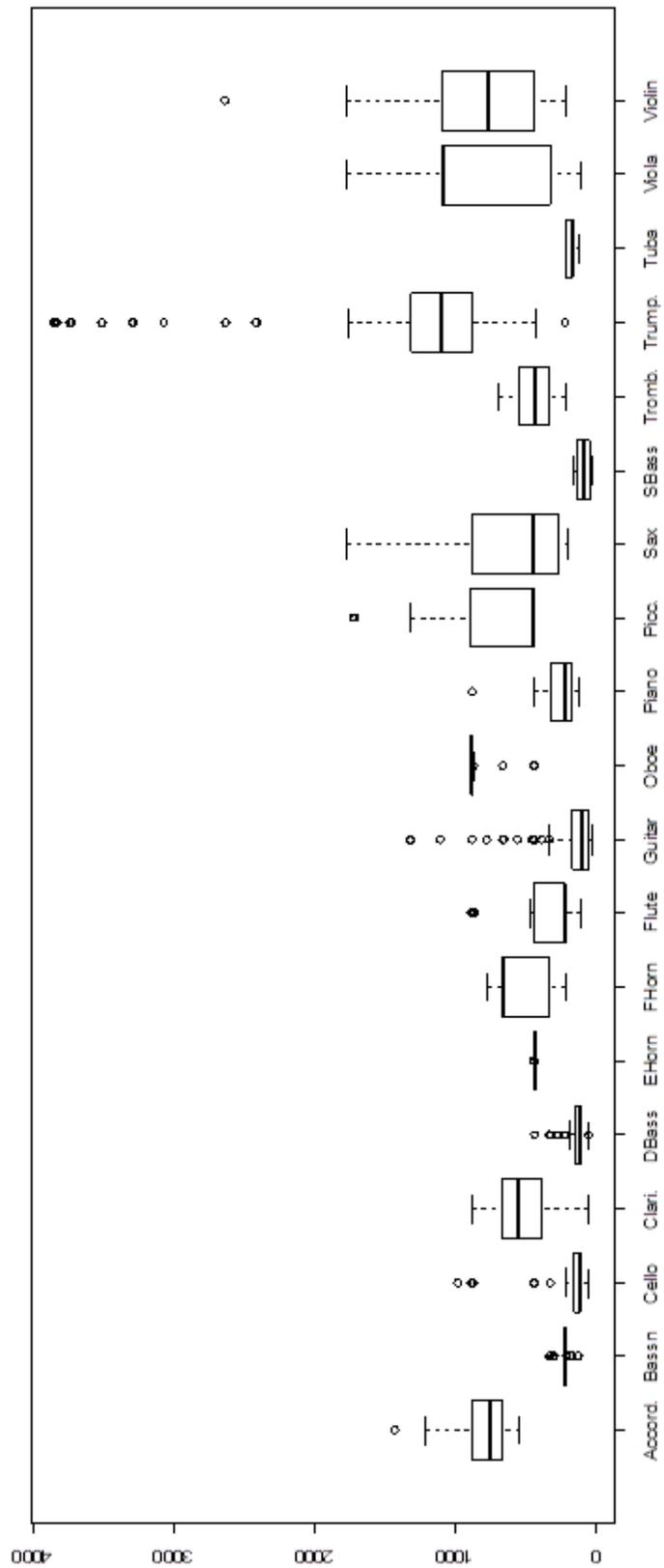
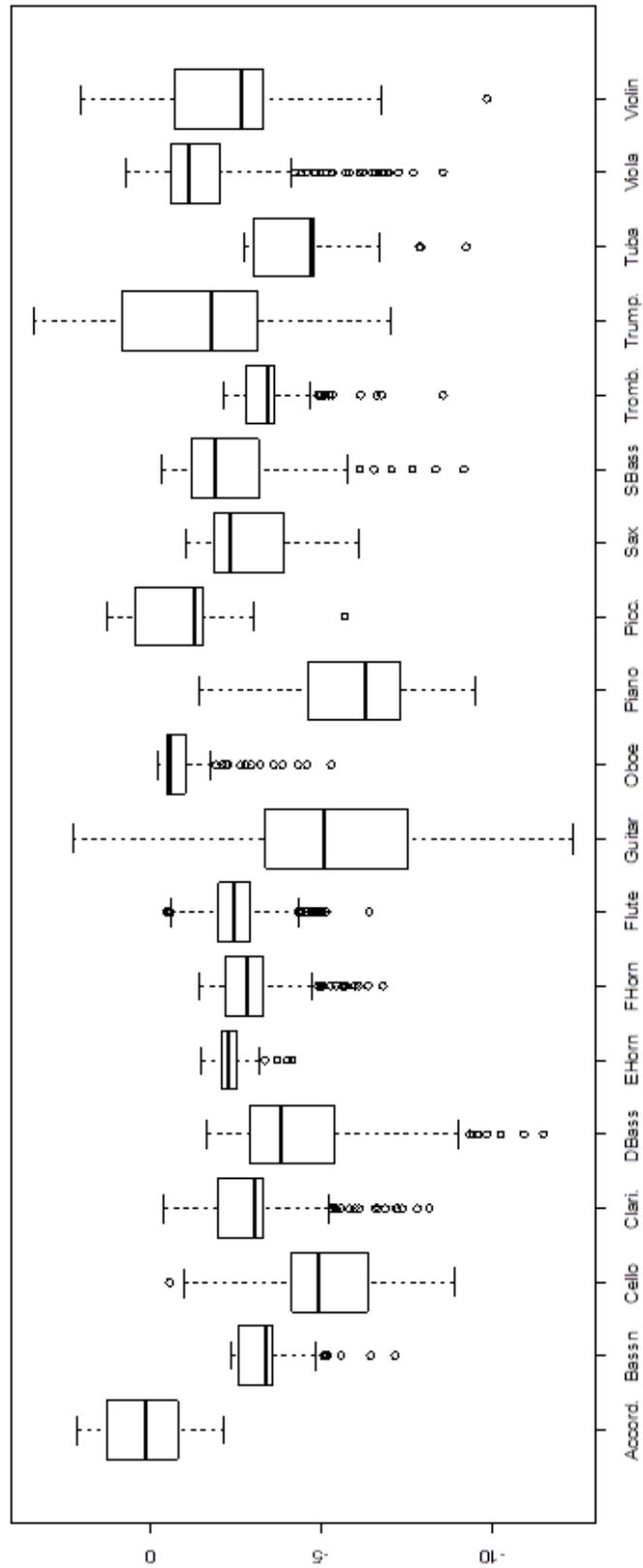


Figure 8.10(c): Boxplot per instrument for energy



8.4.2 Mixture pairs with single instruments

To illustrate the effect of mixing two instruments, average values per feature were considered for the Accordion and Double Bass playing solo, as well as for these two instruments playing together. (Pitch was not taken into account in this instance, as pitch data was not available for the mixture data; presumably because the two instruments need not necessarily be playing the same pitch in the sample).

The average values per feature are shown in Table 8.5 below (* indicates a very small value), and some plots are also drawn in Figures 8.11 to 8.13.

It is clear that there are some complex interactions between instruments in the mixture data and that separating two instruments playing together is not an easy task.

Table 8.5: Averages by instrument for accordion, double bass and mixture of the two

	Accordion	Double Bass	Mixture
MFCCs			
MFCC1	-5.515	2.121	-3.970
MFCC2	-14.011	1.196	-14.167
MFCC3	-2.185	0.945	-0.476
MFCC4	-5.059	-0.254	-6.673
MFCC5	-5.914	2.494	-0.712
MFCC6	-7.014	1.275	-6.576
MFCC7	0.004	2.002	3.624
MFCC8	-7.398	-0.588	-6.182
MFCC9	-2.358	-0.340	-2.974
MFCC10	-4.327	-2.907	-10.200
MFCC11	2.399	-0.308	3.374
MFCC12	-1.828	-0.544	-0.148
MFCC13	23.160	7.578	29.895
Flatness coefficients			
bandsCoef1	-0.155	-0.152	-0.187
bandsCoef2	-0.235	-0.159	-0.263
bandsCoef3	-0.223	-0.150	-0.255
bandsCoef4	-0.223	-0.142	-0.152
bandsCoef5	-0.222	-0.142	-0.119

	Accordion	Double Bass	Mixture
bandsCoef6	-0.213	-0.141	-0.250
bandsCoef7	-0.194	-0.139	-0.248
bandsCoef8	-0.194	-0.135	-0.190
bandsCoef9	-0.186	-0.134	-0.081
bandsCoef10	-0.171	-0.134	-0.234
bandsCoef11	-0.160	-0.137	-0.108
bandsCoef12	-0.153	-0.146	-0.232
bandsCoef13	-0.147	-0.143	-0.101
bandsCoef14	-0.138	-0.143	-0.125
bandsCoef15	-0.131	-0.150	-0.088
bandsCoef16	-0.137	-0.147	-0.158
bandsCoef17	-0.132	-0.156	-0.116
bandsCoef18	-0.133	-0.154	-0.095
bandsCoef19	-0.119	-0.156	-0.086
bandsCoef20	-0.127	-0.163	-0.120
bandsCoef21	-0.129	-0.175	-0.118
bandsCoef22	-0.126	-0.179	-0.142
bandsCoef23	-0.127	-0.182	-0.120
bandsCoef24	-0.122	-0.193	-0.113
bandsCoef25	-0.122	-0.195	-0.131
bandsCoef26	-0.137	-0.204	-0.158
bandsCoef27	-0.153	-0.205	-0.154
bandsCoef28	-0.165	-0.210	-0.161
bandsCoef29	-0.179	-0.215	-0.197
bandsCoef30	-0.196	-0.219	-0.210
bandsCoef31	-0.195	-0.222	-0.219
bandsCoef32	-0.204	-0.225	-0.229
bandsCoef33	-0.215	-0.224	-0.250
Projection coefficients			
prj1	-0.999	-0.993	-0.997
prj2	0.022	0.009	0.036
prj3	*	*	-0.004
prj4 - 33	*	*	*
Harmonic peaks			
HamoPk1	0.997	0.915	1.067
HamoPk2	0.968	0.851	1.005

	Accordion	Double Bass	Mixture
HamoPk3	0.936	0.788	1.009
HamoPk4	0.914	0.732	0.948
HamoPk5	0.933	0.685	0.999
HamoPk6	0.875	0.648	0.848
HamoPk7	0.865	0.605	0.899
HamoPk8	0.852	0.566	0.810
HamoPk9	0.837	0.530	0.864
HamoPk10	0.797	0.478	0.759
HamoPk11	0.775	0.470	0.796
HamoPk12	0.705	0.405	0.719
HamoPk13	0.718	0.382	0.691
HamoPk14	0.657	0.355	0.688
HamoPk15	0.664	0.312	0.672
HamoPk16	0.590	0.286	0.643
HamoPk17	0.617	0.248	0.672
HamoPk18	0.541	0.223	0.607
HamoPk19	0.558	0.201	0.636
HamoPk20	0.488	0.186	0.602
HamoPk21	0.533	0.152	0.640
HamoPk22	0.451	0.127	0.587
HamoPk23	0.489	0.110	0.617
HamoPk24	0.416	0.098	0.574
HamoPk25	0.468	0.073	0.646
HamoPk26	0.364	0.079	0.614
HamoPk27	0.414	0.068	0.602
HamoPk28	0.342	0.056	0.615
Other features			
SpectralCentroid	545.434	129.051	543.885
LogSpectralCentroid	-0.409	-2.350	-0.425
TemporalCentroid	0.842	1.013	1.137
SpectralSpread	510.771	89.769	508.256
LogSpectralSpread	1.097	0.594	1.246
Energy	0.498	-4.547	1.538
ZeroCrossing	302.839	70.103	252.199
RollOff	909.523	168.446	825.640
LogAttackTime	-0.250	-0.108	-0.222

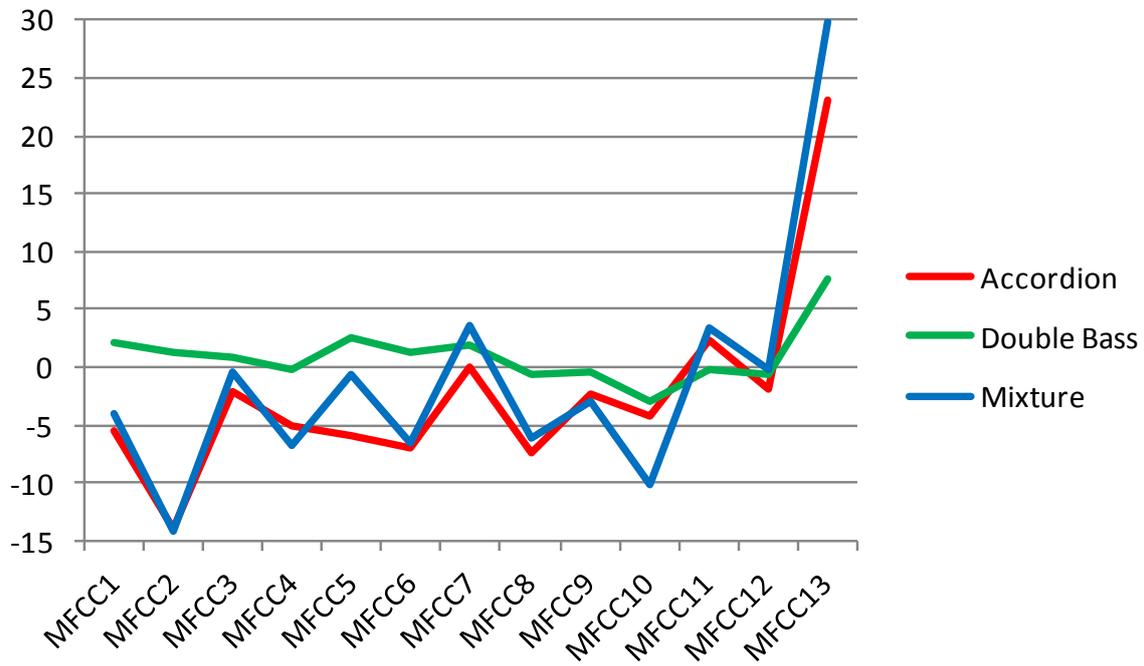


Figure 8.11: MFCCs for single instruments and mixture – average values

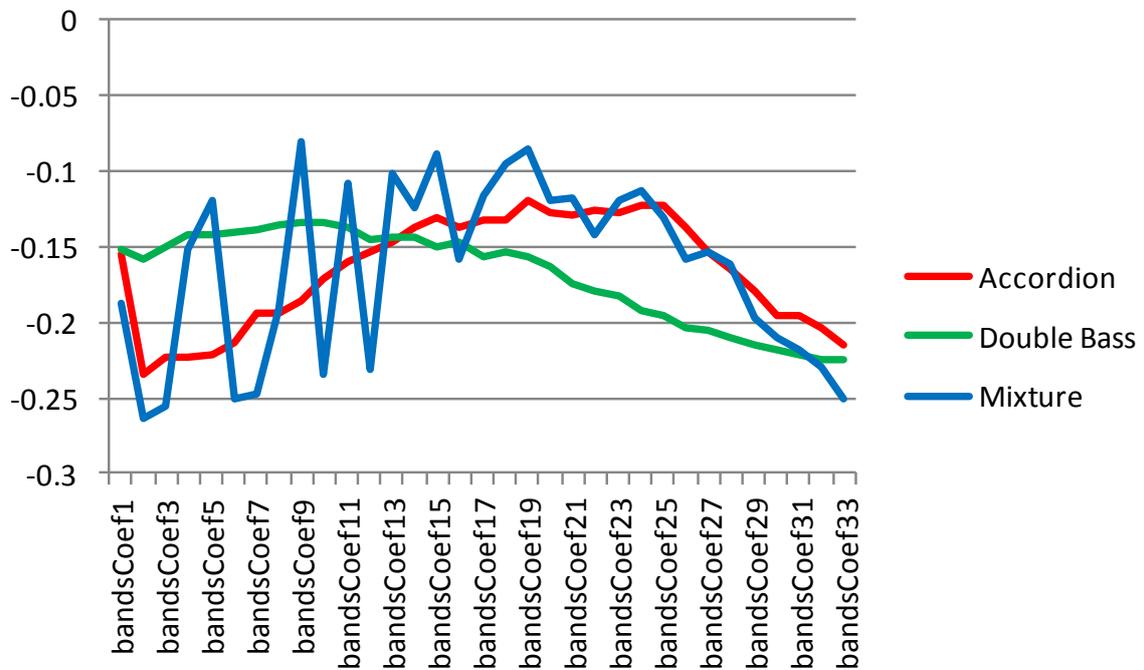


Figure 8.12: Flatness coefficients for single instruments and mixture – average values

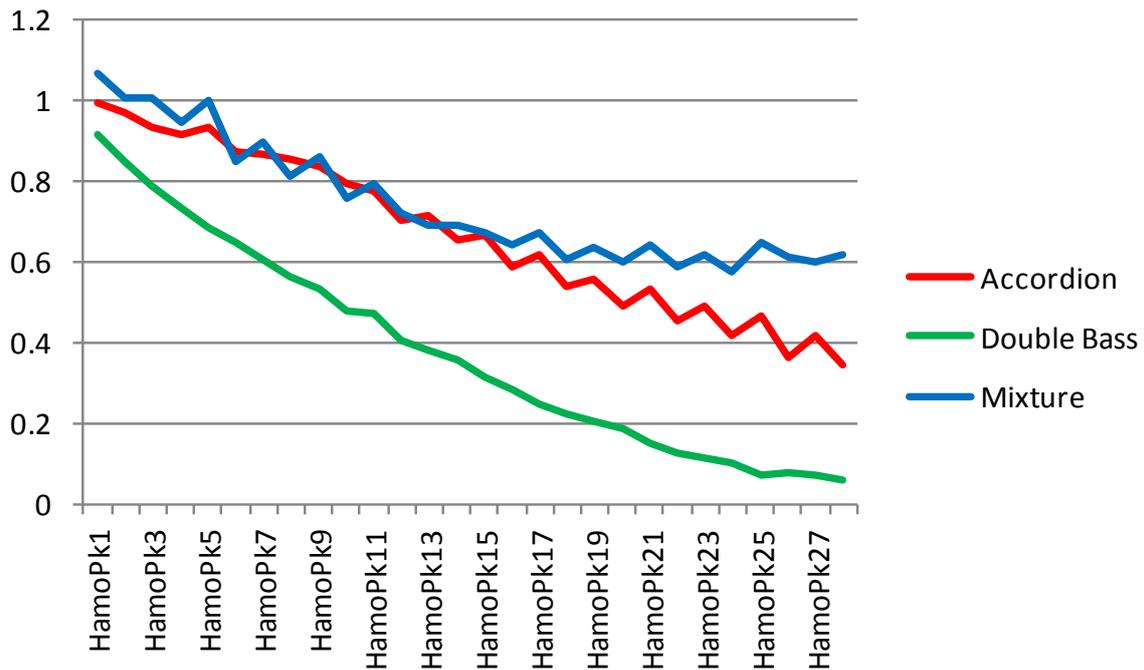


Figure 8.13: Harmonic peaks for single instruments and mixture – average values

In terms of MFCCs, the mixture profile more closely matches the Accordion than the Double Bass (Figure 8.11), while for flatness coefficients the mixture has a discernibly more erratic profile than the single instruments do (Figure 8.12). For harmonic peaks, the mixture again resembles the shape of the Accordion profile, although at a higher level (Figure 8.13). These examples reinforce the finding by Little and Pardo (2008), Wiczorkowska and Kubera (2010) and Spyromitros-Xioufis *et al.* (2011) that where the goal is to identify instruments in mixtures, it is better to train from mixture data rather than from isolated instruments.

8.4.3 Dimension reduction

The dataset under consideration had very high dimensionality (after data pre-processing, there were 125 758 observations and 122 features), so in order to better visualise the data, Principal Component Analysis (PCA) was performed to reduce the data to two dimensions so that the data could be plotted on two axes.

Only the single instrument data was considered, and a random sample was taken of the data to avoid clutter on the plots. In the random sampling process, the Guitar data was also deliberately undersampled to avoid it dominating the plots. Only MFCCs were considered for this exercise.

A simple PCA was run in R (data was centered but not scaled) and the first two dimensions represented about 45% of the variance in the data.

Figure 8.14 shows the resulting plot for all 13 MFCCs and all 19 single instruments. No clear separation between instruments is apparent.

In Figure 8.15 the PCA was repeated, but this time the instrument data was aggregated by the first level of the widely used Hornbostel-Sachs musical instrument categorisation system (see Chapter 3, Section 3.5.5 for details; in this instance the 19 instruments were divided into 3 categories, namely chordophones, aerophones and electrophones). Again no clear separation between categories was apparent.

Figure 8.16 shows the same PCA results, but this time using the second level Hornbostel-Sachs categories; we still have large overlaps between categories.

This demonstrates that two dimensions are not enough to represent the data, as is apparent from the fairly low 45% of explained variance. Three dimensions explain 59% of the variance, and four dimensions 69%. Bear in mind though that this is just the explained variance in terms of the 13 MFCCs and not the entire set of features.

Figure 8.14: PCA in 2 dimensions of sample of single instrument data (MFCCs only)

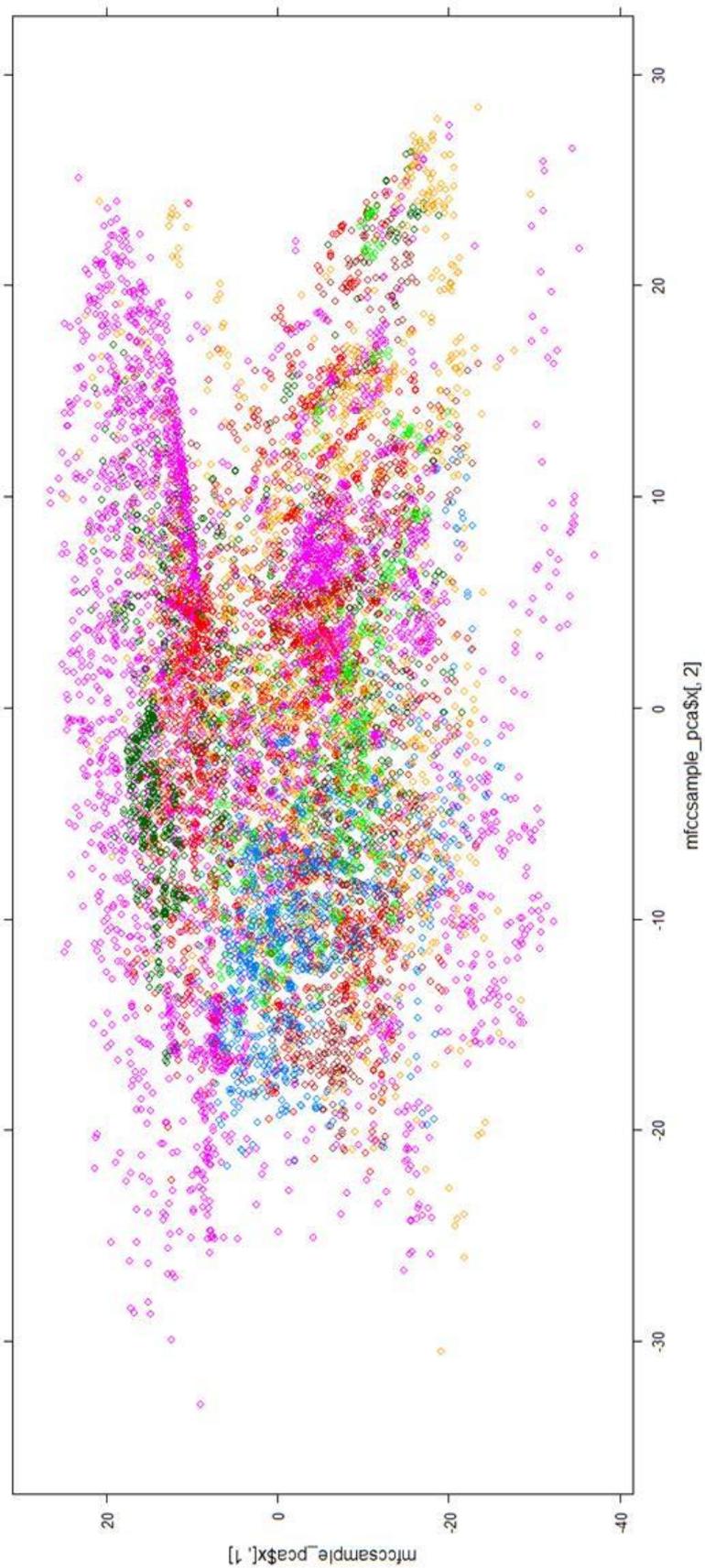


Figure 8.15: PCA in 2 dimensions of sample of single instrument data by first level Hornbostel-Sachs (using MFCCs only)

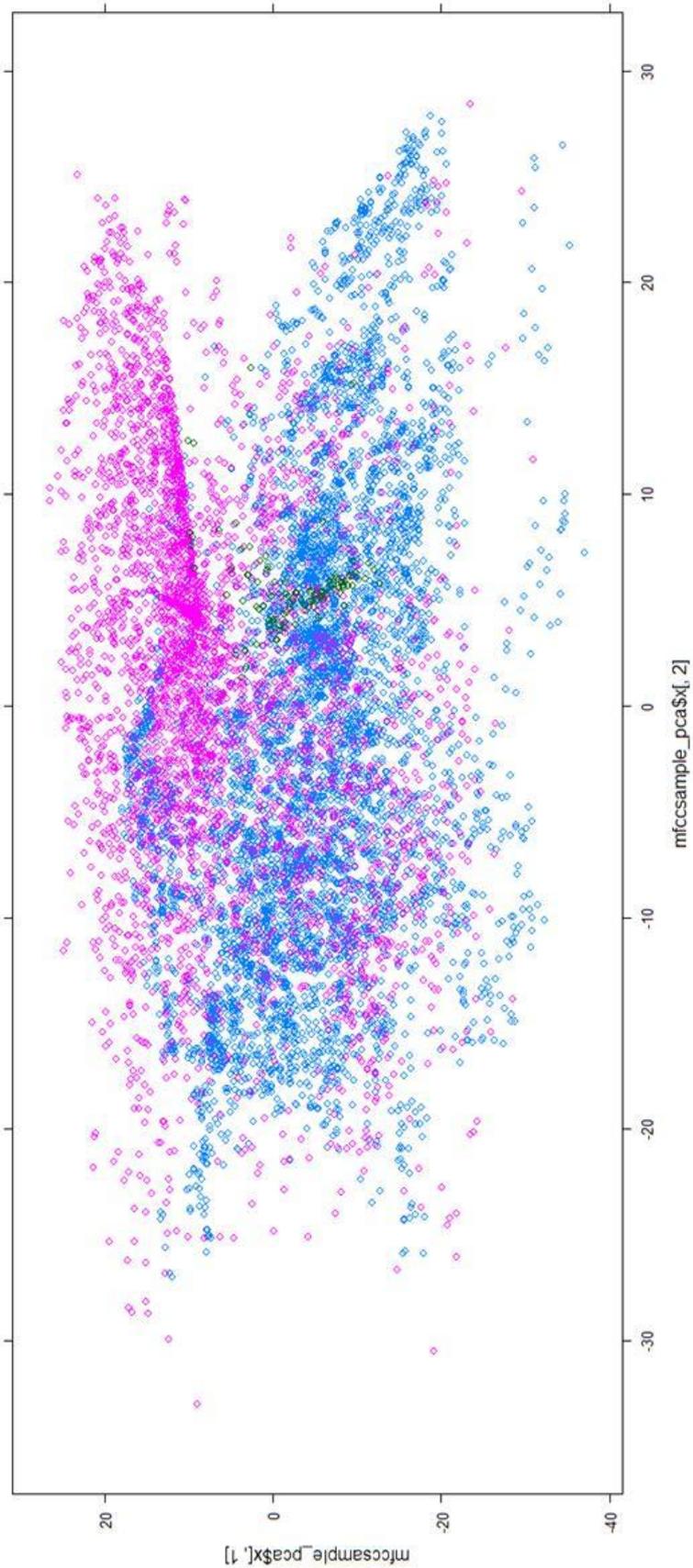
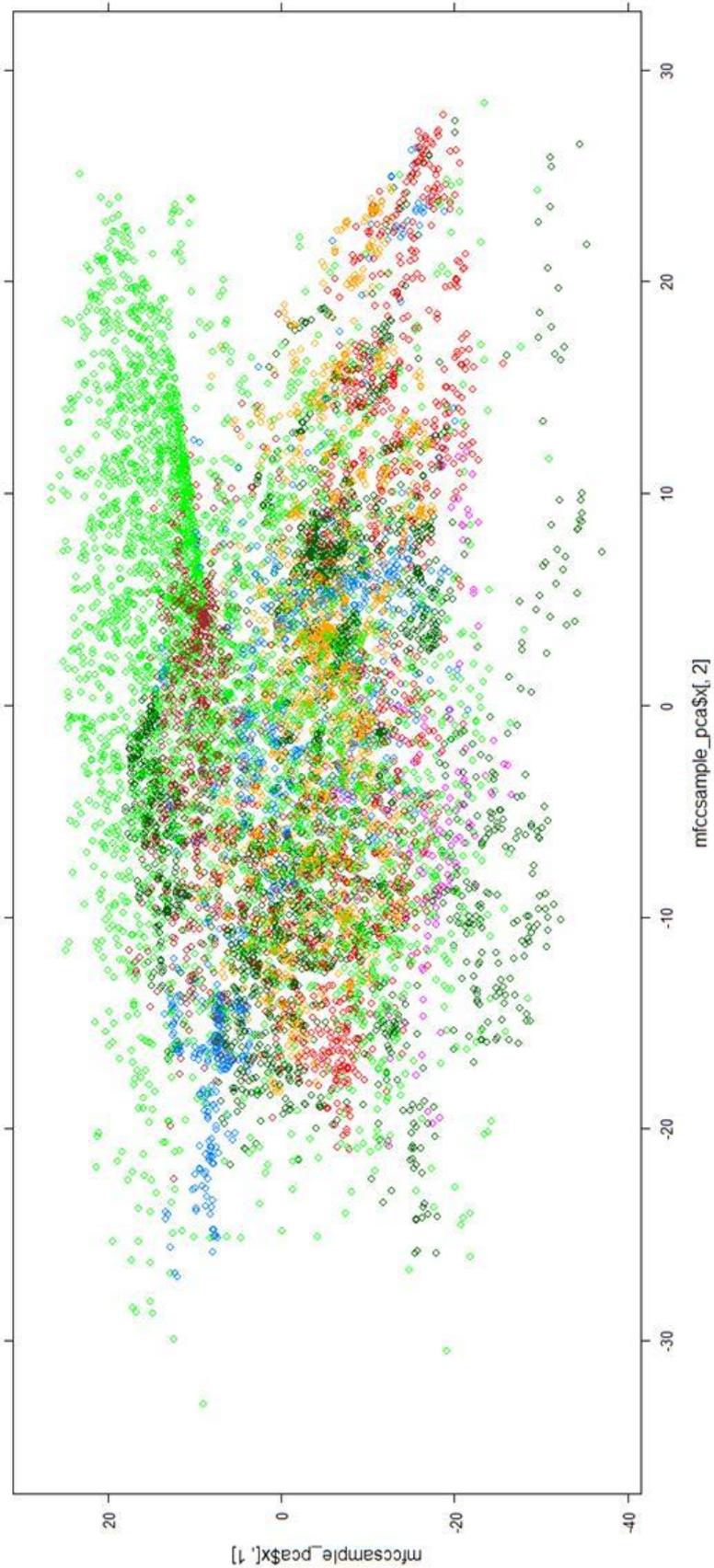


Figure 8.16: PCA in 2 dimensions of sample of single instrument data by second level Hornbostel-Sachs (using MFCCs only)



8.5 Empirical results

8.5.1 Methodology

The datasets described above were used to empirically evaluate the proposed feature selection method as discussed in Chapter 5. To this end, the binary relevance multi-label transformation was applied and two classifiers – SVM and kNN – were trained, both on the full set of features as well as on a selected set of features. The results were then evaluated using some of the multi-label measures defined in Chapter 4, Section 4.7. The methodology followed will now be discussed in more detail.

Empirical evaluation was done using the statistical computing software package R. Bootstrap samples for the feature selection step were drawn from the two training datasets (single instruments as well as mixture data); the number of bootstrap samples (referred to as *Btrain* in the R code given in Appendix A.5) was set to 50 for experimenting with different hyperparameter values, and set to 20 for evaluating feature selection. Within each bootstrap sample, 750 observations were drawn from the mixture dataset, and 570 from the single instrument data. The sample from the single instrument data was stratified so that 30 observations were drawn for each different instrument, to ensure that all instruments are represented in the sample data (30 observations \times 19 instruments = 570 observations from the single instrument dataset). Each bootstrap sample therefore consisted of 1320 observations.

Feature selection was subsequently performed using the technique proposed in Chapter 5, Section 5.6.2. We took $\alpha = 0.9$ as the significance level for identifying irrelevant features and $B = 100$ as the number of repetitions to determine critical values. Different values for α and B were not evaluated in this study, since the chosen values seemed intuitively reasonable. However, experimenting with different values for α and B could be the subject of further research. The only further parameter that required specification was the feature cut-off point U . This is the number of labels for which a feature needs to be considered relevant in order to be selected. Different values for U were evaluated, and details are presented in Section 8.5.4.

SVM and kNN classifiers were then constructed, for the full feature set as well as for the selected feature set. This was performed separately for each label k , $1 \leq k \leq 23$. For fitting the SVM classifiers, the data was scaled. The radial basis function kernel was used for each SVM, with hyperparameter $\sigma = \frac{1}{p}$, where p was the number of features in the dataset. Different cost parameters C were evaluated; see Section 8.5.3 for details. For the kNN classifiers, a further sample was drawn from the data in such a way that the number of positive examples in each binary relevance dataset equalled the number of negative examples. This was necessary to avoid running into problems with unbalanced data in the kNN algorithm. After sampling, the kNN classifiers were constructed (with data scaled and centred) using a weighted kNN implementation (which uses kernel functions to weight neighbours according to their distance) with Euclidean distance measure and Gaussian kernel. Different values for the number of neighbours k were experimented with; details can be found in Section 8.5.3

After fitting the classifier models, posterior probabilities were calculated for each label in every test case; these probabilities were then ranked and the two top ranking labels were taken as the predicted label set for each test case. These were compared to the actual labels for each test case and five multi-label evaluation measures were thereafter calculated to evaluate the model performance. The five measures calculated were Hamming Loss, Precision (which is the same as Recall for this dataset, since by design there are only 2 true labels and 2 predicted labels), Accuracy, One-Error and Coverage (the last two being ranking-based measures and the first three being example-based measures).

The R programs used in this analysis can be found in Appendix A. Appendix A.1 contains the main R program, while the subroutines called for sampling, feature selection and evaluation measures are given in Appendix A.2, Appendix A.3 and Appendix A.4 respectively.

8.5.2 Overall accuracy

Several values for the SVM cost parameter C , the number of neighbours for kNN k , and the feature cut-off point U were evaluated, and models with and without feature

selection were also compared. Details will be presented in the following sections, but over the course of all of these experiments, the best performing model (evaluated in terms of example-based measures) was an SVM, incorporating feature selection, and fitted with $C = 1$ and $U = 11$. For this model, the following results were obtained:

Hamming Loss	0.0803
Precision	0.5384
Accuracy	0.4257

In terms of ranking-based measures, the best result for One-Error was obtained by an SVM incorporating feature selection and using $C = 100$ and $U = 12$ (One-Error = 0.7822). For Coverage, the best result was a kNN model with $k = 5$ and $U = 11$ (Coverage of 6.5043).

Unfortunately, these results cannot be compared to the results obtained in the ISMIS contest, since the data was pre-processed for our study, and in the contest additional knowledge such as the prior distribution of the test data was incorporated into the models. In addition, the way in which recognition performance was calculated for the contest, differed from the standard multi-label evaluation measures. Comparison to other studies are also not possible, since these generally dealt with other datasets, different numbers of instruments and different numbers of features. In particular, a summary by Barbedo (2011) of previous polyphonic instrument recognition studies showed that most dealt with fewer instruments and also fewer features. A search of published papers on instrument recognition did not yield even a single paper in which conditions were similar enough for results to be compared. This affirms the need for benchmark datasets for instrument recognition studies. Until some way of objectively evaluating proposed instrument recognition algorithms is established, advances in the field will be very difficult.

Nevertheless, as a very crude benchmark, Srinivasan *et al.* (2002) evaluated the instrument recognition capabilities of conservatory students and found that in a test involving the recognition of 27 isolated instrument tones, the average recognition rate was 55.7%. In this context, the obtained Precision of 53.8% for the recognition of 23 possible instruments playing in duets seems like a fair result.

8.5.3 Hyperparameter choice

Different values for C (the SVM cost parameter) and k (the number of neighbours in kNN) were evaluated to find the optimum choice of parameter for the two classifiers. Only the full dataset was considered for this purpose. Table 8.6 shows the error rates for different values of C for the SVM, with corresponding boxplots in Figure 8.17(a) and Figure 8.17(b). Table 8.7 shows the error rates for different values of k for kNN, with corresponding boxplots in Figure 8.18(a) and Figure 8.18(b). In both Table 8.6 and Table 8.7, bold entries indicate the best obtained result for each error measure.

Table 8.6: Error rates for different choices of cost parameter C for SVM

	ERROR MEASURE				
	Hamming Loss	Precision	Accuracy	One-Error	Coverage
$C = 0.00001$	0.1287	0.2600	0.1848	0.9201	6.8924
$C = 0.1$	0.0836	0.5193	0.4059	0.7948	6.7148
$C = 1$	0.0840	0.5170	0.4043	0.7951	6.7466
$C = 10$	0.0837	0.5186	0.4068	0.7845	6.6544
$C = 100$	0.0843	0.5155	0.4032	0.7857	6.6612
$C = 10000$	0.0839	0.5176	0.4055	0.7872	6.6599

Table 8.7: Error rates for different choices of number of neighbours k for kNN

	ERROR MEASURE				
	Hamming Loss	Precision	Accuracy	One-Error	Coverage
$k = 1$	0.1009	0.4200	0.2892	0.8012	6.5579
$k = 3$	0.1013	0.4176	0.2885	0.8063	6.5306
$k = 5$	0.1015	0.4166	0.2878	0.8066	6.5043
$k = 7$	0.1049	0.4200	0.2897	0.8020	6.5517
$k = 11$	0.1012	0.4183	0.2882	0.8063	6.5360
$k = 17$	0.1010	0.4190	0.2901	0.8080	6.5570
$k = 25$	0.1004	0.4228	0.2916	0.8049	6.5447

Except for extremely small values of C ($C = 0.00001$), the choice of C does not have a substantial impact on classification performance. Within the range considered, the choice of k also has little to no effect on the error measures.

Figure 8.17(a): Boxplot showing Hamming Loss for different values of C

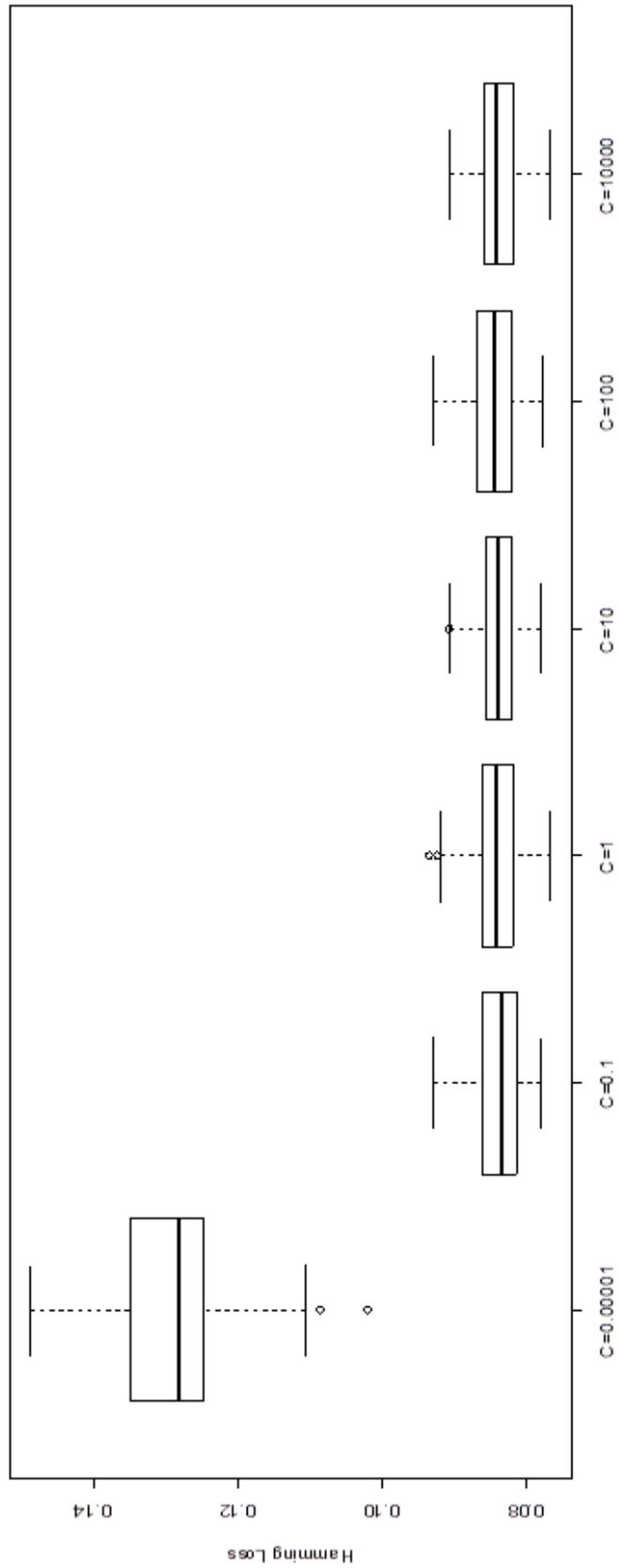


Figure 8.17(b): Boxplot showing Precision for different values of C

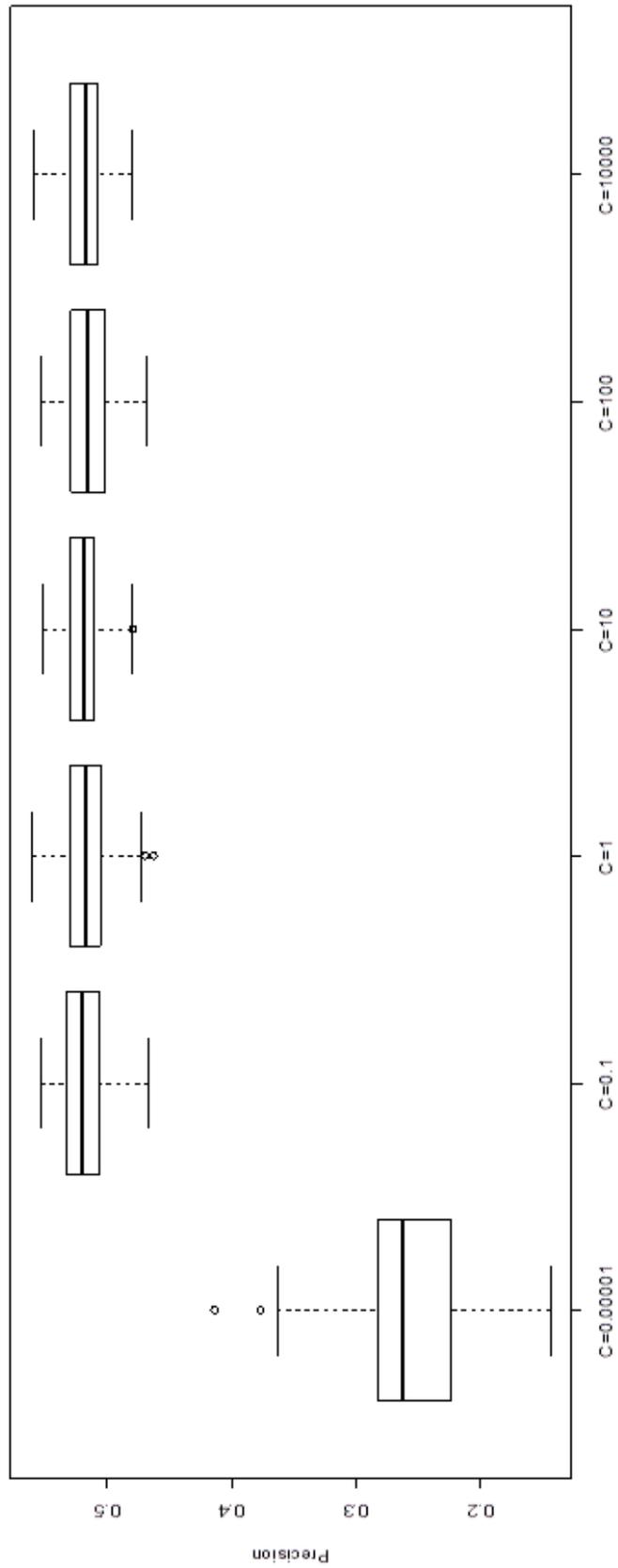


Figure 8.18(a): Boxplot showing Hamming Loss for different values of k

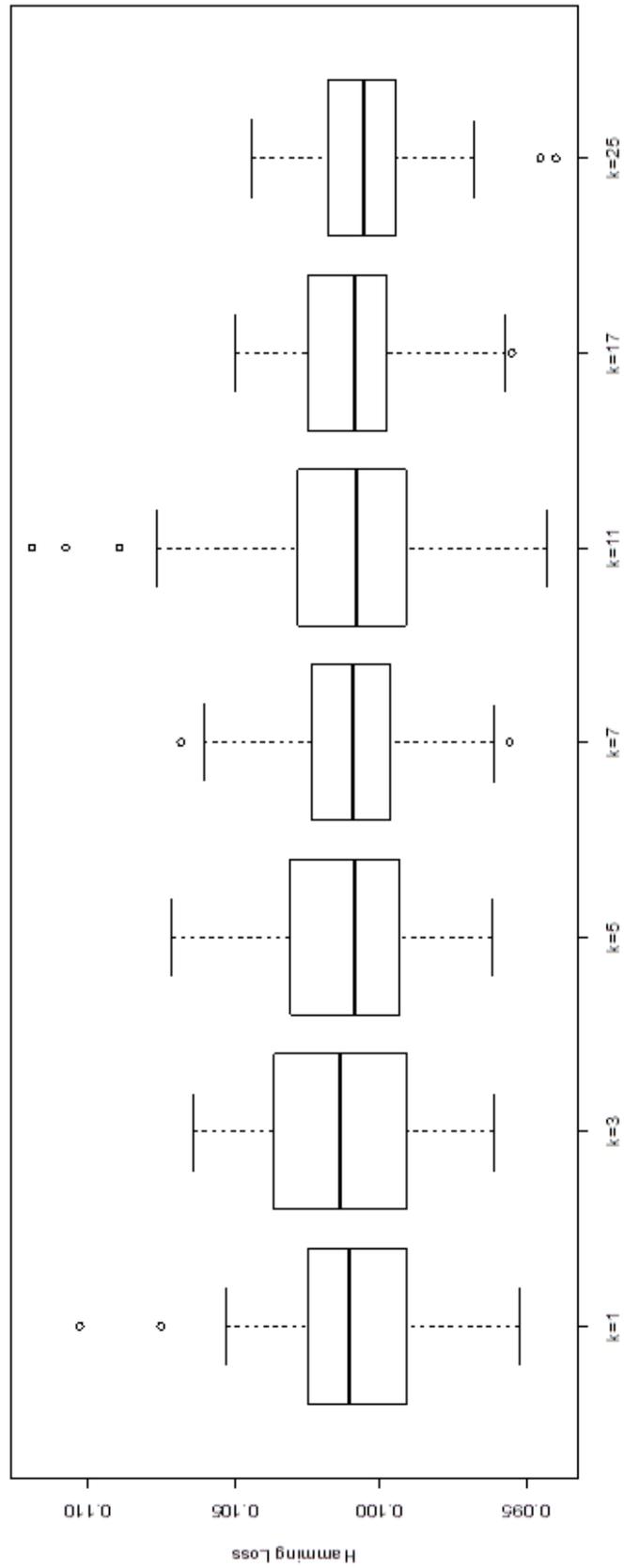
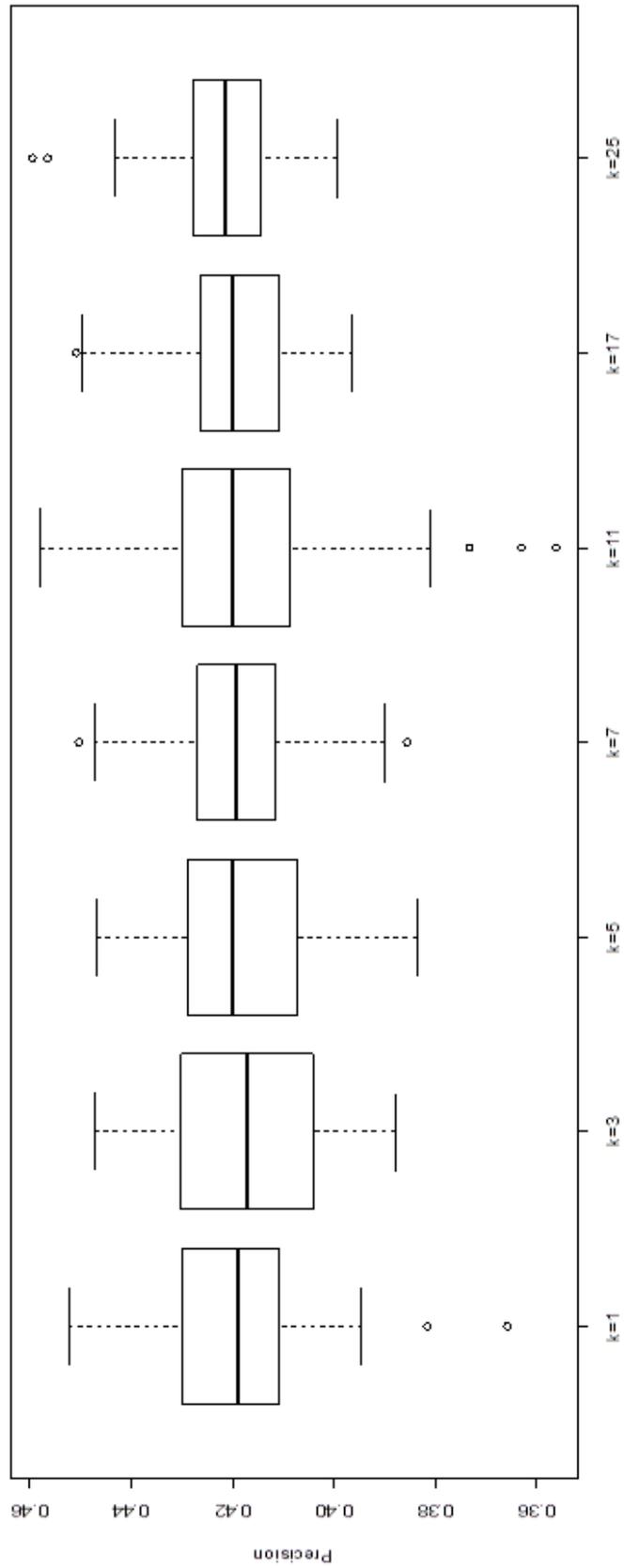


Figure 8.18(b): Boxplot showing Precision for different values of k



Overall therefore, it appears that the choice of hyperparameter has very little effect on the classification results. Therefore, to evaluate feature selection results (discussed in the next section), values of $C = 1$ and $k = 1$ were chosen for the sake of simplicity: $C = 1$ is often proposed as a default value in the literature (for example, the default value for C in the `ksvm` package's implementation of SVMs in R is 1), and while $k = 1$ did marginally worse than a value of, say, $k = 25$, the smaller value lead to a much quicker implementation.

It is difficult to speculate about the reasons for the negligible effect of the model hyperparameters on results. Firstly, it should be pointed out that an exhaustive grid-search was not performed, so there may yet be an optimal value of C and / or k which remains to be discovered. These values were also not considered in conjunction with other model parameters – for instance, the sigma parameter of the RBF kernel in the SVM model was kept constant, as was the distance measure in the kNN model. In addition, it has been shown (Gunn, 1998; Ben-Hur and Weston, 2010) that C may have a range of optimal values, depending on the decision boundary desired, so the results for the SVMs are not entirely unexpected.

8.5.4 Feature selection

The main purpose of this empirical study was to evaluate the effect of feature selection and, in particular, the proposed feature selection technique.

Firstly, as has been mentioned in Section 8.5.2 above, the best performing model overall was one incorporating feature selection. On this basis alone, feature selection seems to be useful for this dataset.

Overall, except in cases where a very strict feature cut-off point U was used for selection (specifically, values of around $U = 18$ or stricter), models incorporating feature selection fared better than corresponding models using the full feature set. A cut-off point of 18 implies that a feature needs to be considered relevant for at least 18 (out of the 23) labels in order to be considered relevant. The following tables (Table 8.8(a)-(e)) compare the results for feature selection versus the full dataset

corresponding to different parameters. In these tables, highlighted entries indicate the better result between the full and selected feature sets.

Table 8.8(a): Hamming Loss for full vs. selected feature sets for different classifiers and feature cut-off points

Feature cut-off point U	Classifier	Full dataset	Selected feature set
18	SVM	0.0836	0.0928
	kNN	0.1018	0.1028
12	SVM	0.0844	0.0829
	kNN	0.1010	0.0969
6	SVM	0.0848	0.0833
	kNN	0.1013	0.0980

Table 8.8(b): Precision for full vs. selected feature sets for different classifiers and feature cut-off points

Feature cut-off point U	Classifier	Full dataset	Selected feature set
18	SVM	0.5194	0.4661
	kNN	0.4148	0.4089
12	SVM	0.5146	0.5235
	kNN	0.4194	0.4429
6	SVM	0.5126	0.5210
	kNN	0.4176	0.4363

Table 8.8(c): Accuracy for full vs. selected feature sets for different classifiers and feature cut-off points

Feature cut-off point U	Classifier	Full dataset	Selected feature set
18	SVM	0.4078	0.3507
	kNN	0.2859	0.2805
12	SVM	0.4026	0.4095
	kNN	0.2888	0.3044
6	SVM	0.3991	0.4061
	kNN	0.2877	0.2996

Table 8.8(d): One-Error for full vs. selected feature sets for different classifiers and feature cut-off points

Feature cut-off point U	Classifier	Full dataset	Selected feature set
18	SVM	0.7878	0.7905
	kNN	0.8063	0.7960
12	SVM	0.7839	0.7822
	kNN	0.8027	0.8062
6	SVM	0.7853	0.7851
	kNN	0.8086	0.8175

Table 8.8(e): Coverage for full vs. selected feature sets for different classifiers and feature cut-off points

Feature cut-off point U	Classifier	Full dataset	Selected feature set
18	SVM	6.6452	6.7007
	kNN	6.5202	6.6525
12	SVM	6.6608	6.6801
	kNN	6.5459	6.5691
6	SVM	6.6777	6.6331
	kNN	6.5264	6.5489

These tables show the following:

- For high values of U (corresponding to very few features being selected; see Figure 8.21(a) further on in this section), performance is better using the full set of features rather than the selected set of features.
- Although the superiority of the selected feature set over the full feature set is not substantial in most instances, it should be kept in mind that even a marginally better result using a fewer number of features will lead to more efficient classifiers.
- For ranking-based measures, the results are not as clear cut. In some instances, the selected feature set delivers better results while in other (especially in the case of kNN), the full dataset delivers better results. This implies that if the aim is to optimise a ranking-based measure, feature selection – and indeed, SVMs – might not necessarily deliver superior results. There is no apparent explanation for this observation.

With the usefulness of feature selection (for most instances) demonstrated above, the next aim was to further investigate the impact of the feature selection cut-off point U . Recall that this point U refers to the number of labels for which a feature needs to be deemed relevant (according to the selection measure applied) for it to be included in the set of selected features. Since the number of labels (K) in this study is 23, values ranging from 0 to 23 were investigated, with 0 corresponding to the full feature set and 23 being the very strict criterion that in order to be selected, a feature should be deemed relevant for all labels. In practice, experiments showed that for values of $U > 20$, the selection method was not feasible since in some instances no relevant features could be found. Tables 8.9(a) and 8.9(b) show error measures for different values of U for SVM and kNN respectively.

Table 8.9(a): Error rates for different feature cut-off points: SVM

	ERROR MEASURE				
	Hamming Loss	Precision	Accuracy	One-Error	Coverage
$U=0$	0.0836	0.5194	0.4079	0.7940	6.7440
1	0.0830	0.5225	0.4102	0.7954	6.7663
2	0.0820	0.5288	0.4161	0.7943	6.8177
3	0.0817	0.5304	0.4191	0.7970	6.8345
4	0.0817	0.5301	0.4163	0.7949	6.8354
5	0.0820	0.5283	0.4156	0.7949	6.8166
6	0.0818	0.5294	0.4155	0.7948	6.7903
7	0.0816	0.5306	0.4171	0.7969	6.7773
8	0.0816	0.5306	0.4166	0.7953	6.7480
9	0.0805	0.5369	0.4234	0.7952	6.8070
10	0.0825	0.5258	0.4117	0.7939	6.7808
11	0.0803	0.5384	0.4257	0.7942	6.7718
12	0.0814	0.5317	0.4181	0.7952	6.8017
13	0.0830	0.5225	0.4082	0.7968	6.8047
14	0.0816	0.5310	0.4165	0.7906	6.8148
15	0.0847	0.5130	0.3982	0.7927	6.8467
16	0.0854	0.5087	0.3940	0.8005	6.8441
17	0.0885	0.4912	0.3752	0.7960	6.8489
18	0.0946	0.4559	0.3386	0.7946	6.8298
19	0.0987	0.4324	0.3149	0.7861	6.8289
20	0.1089	0.3736	0.2643	0.8042	6.8891

Table 8.9(b): Error rates for different feature cut-off points: kNN

	ERROR MEASURE				
	Hamming Loss	Precision	Accuracy	One-Error	Coverage
0	0.1022	0.4121	0.2838	0.8097	6.5259
1	0.0999	0.4256	0.2924	0.7988	6.5884
2	0.0987	0.4326	0.2979	0.8042	6.5577
3	0.0979	0.4370	0.3009	0.8080	6.5756
4	0.0976	0.4389	0.3024	0.8197	6.5332
5	0.0982	0.4356	0.2993	0.8094	6.5418
6	0.0986	0.4332	0.2979	0.8119	6.5342
7	0.0987	0.4322	0.2972	0.8094	6.5488
8	0.0971	0.4415	0.3043	0.8113	6.5427
9	0.0967	0.4439	0.3061	0.8094	6.5562
10	0.0975	0.4395	0.3017	0.8163	6.5551
11	0.0977	0.4384	0.3009	0.8212	6.5611
12	0.0982	0.4354	0.3004	0.8074	6.5505
13	0.0982	0.4355	0.2991	0.8086	6.5803
14	0.0976	0.4388	0.3016	0.8160	6.5926
15	0.0982	0.4355	0.3004	0.8151	6.5546
16	0.1001	0.4245	0.2928	0.8002	6.6031
17	0.1007	0.4213	0.2894	0.7891	6.6888
18	0.1023	0.4116	0.2825	0.7980	6.6971
19	0.1038	0.4030	0.2780	0.8007	6.7415
20	0.1137	0.3461	0.2372	0.7973	6.6275

The values for Hamming Loss and Precision in Tables 8.9(a) and (b) above are plotted in the following four figures (Figures 8.19(a)-(b) and 8.20(a)-(b)).

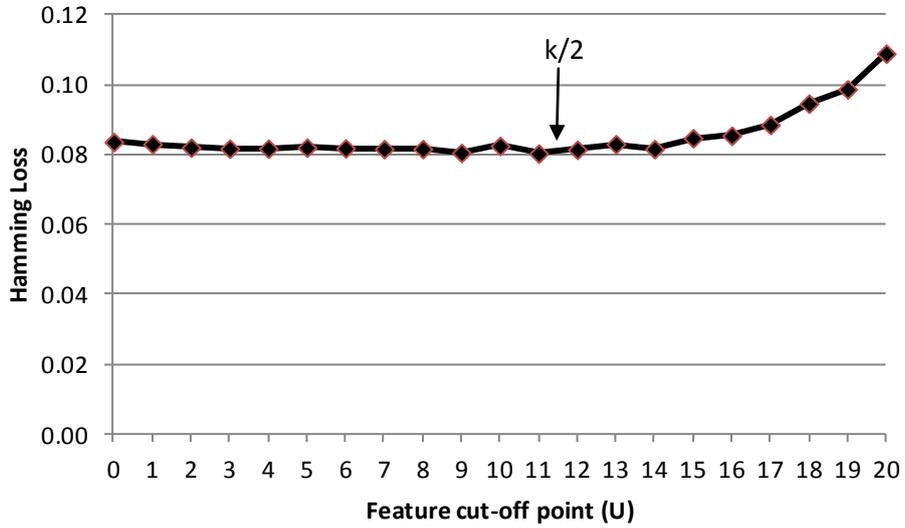


Figure 8.19(a): SVM: Hamming Loss for different feature cut-off points

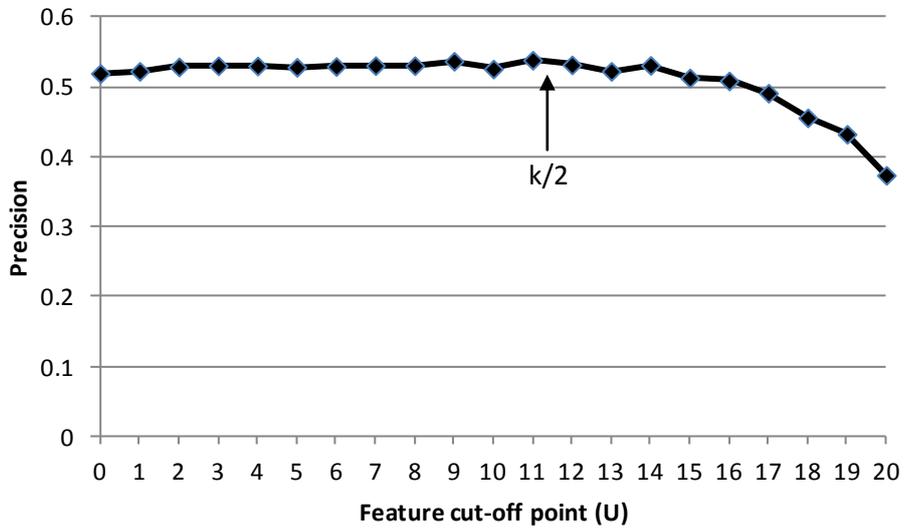


Figure 8.19(b): SVM: Precision for different feature cut-off points

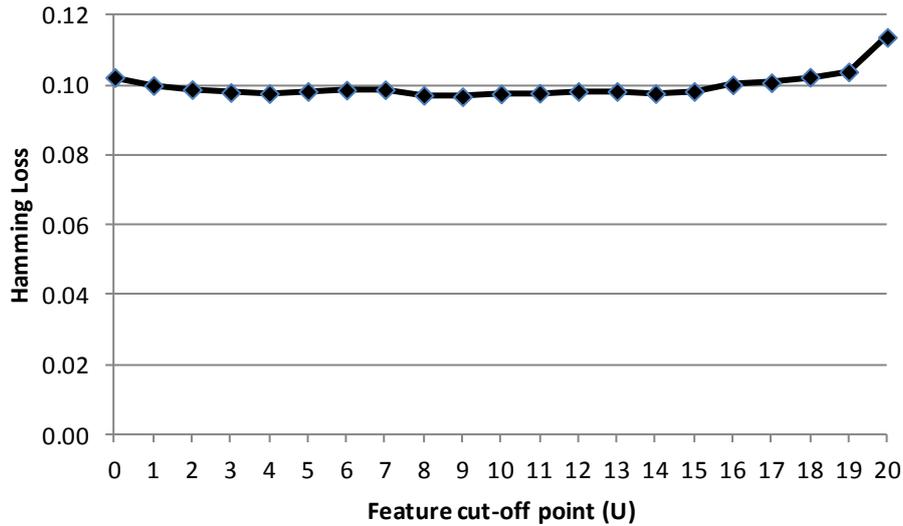


Figure 8.20(a): kNN: Hamming Loss for different feature cut-off points

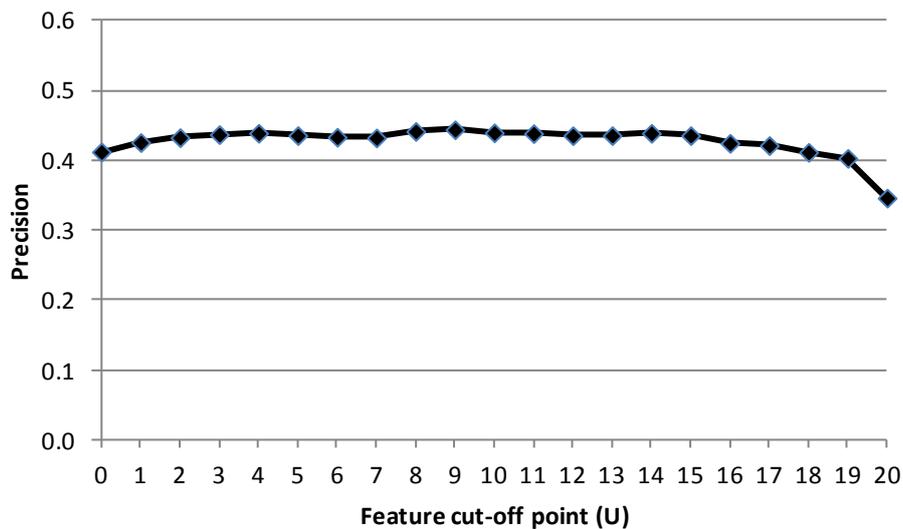


Figure 8.20(b): kNN: Precision for different feature cut-off points

For this dataset, there appears to be a cut-off point U in the vicinity of $\frac{K}{2}$ which is optimal for feature selection. Up to this point, the exclusion of features has no substantial impact on test error rates, which means that classification can be done using fewer features and therefore more efficiently, without negatively impacting results. Excluding further features beyond this point though, has a detrimental impact on error rates. The increase in error rates is gradual at first, with significant increase only occurring from a cut-off point of $U = 18$ (or about $\frac{3K}{4}$ for this dataset). For this

dataset therefore, a natural feature selection rule appears to be to include a feature if it is deemed relevant (in terms of the feature selection method applied) for about half of the labels. If even stricter feature selection is required, the rule can be amended to only include features which are relevant for at least three quarters of labels at only limited cost in terms of classification error.

The following graph (Figure 8.21) shows the number of features chosen on average over the different training iterations for each feature cut-off point U , together with the corresponding Precision values from the SVM.

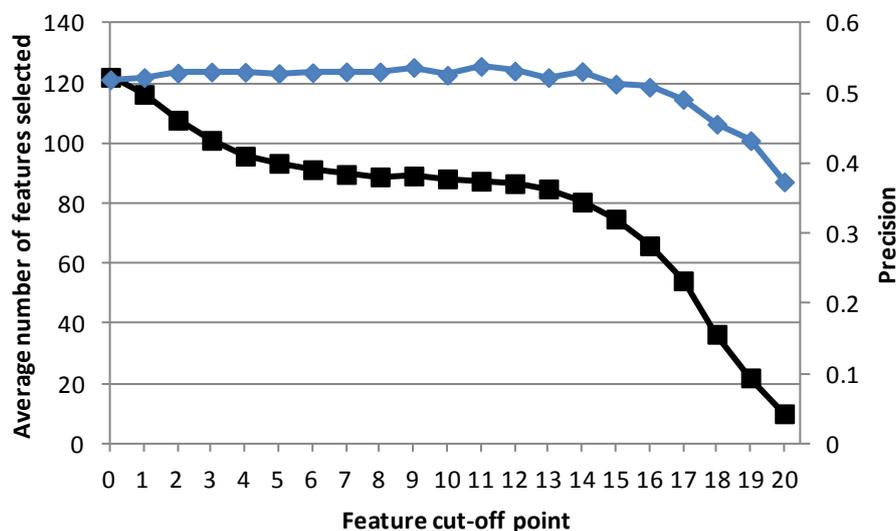


Figure 8.21: Average number of features selected (in black) for different feature cut-off points, plotted with SVM Precision (in blue) for each cut-off point

The number of selected features tapers off at around 90. The decline in Precision co-occurs with the steep decline in the number of features selected. Figure 8.21 also clearly shows that more or less the same performance can be obtained with around 90 features as with 122 – another strong argument for feature selection.

8.5.5 Choice of classifier

The aim of this study was not to evaluate different classifiers for use with music data. However, in all of the different scenarios evaluated in terms of choice of parameter

and feature selection, the best performance overall was achieved by an SVM on all evaluation measures with the exception of Coverage (see Section 8.5.2). As Table 8.10 below shows, especially for Precision and Accuracy, the superiority of the SVM is apparent.

Table 8.10: Error rates for best performing SVM and kNN respectively

	ERROR MEASURE				
	Hamming Loss	Precision	Accuracy	One-Error	Coverage
SVM	0.0803	0.5384	0.4257	0.7822	6.6221
kNN	0.0962	0.4469	0.3074	0.7891	6.5043

For Hamming Loss, Precision and Accuracy the best performance was obtained by an SVM with a cost parameter $C = 1$ and using feature selection with a selection cut-off point of 11. The best One-Error was obtained by an SVM with a cost parameter $C = 100$ and using feature selection with a selection cut-off point of 12. The best Coverage was obtained by a kNN model with $k = 5$ and using the full set of features.

Results in Section 8.5.4 also suggested that kNN fares better on ranking-based measures. Coupled with the fact that kNN is generally faster to implement than SVM, this suggests that although the SVM appears to be superior for this dataset, kNN could be a suitable choice for situations where implementation time is important (for example in a real-time classification scenario) and where more emphasis should be placed on ranking-based measures.

8.5.6 Feature importance

A crude measure of feature importance can be derived by calculating how frequently a feature is selected during the different evaluation runs. This is calculated for each individual feature as

$$\frac{1}{R} \sum_{r=1}^R \frac{s_r}{B_r}$$

where s_r is the number of times the feature was selected in evaluation run r , B_r is the number of training iterations within each run r and R is the total number of runs.

This gives a figure between 0 and 1 for each feature which gives an indication of how important the feature can be considered according to the feature selection method implemented. In Table 8.11(a) below, features which were selected at least 95% of the time are listed, while Table 8.11(b) lists features which were selected less than 10% of the time.

Table 8.11(a): Features which were selected at least 95% of the time

Feature name	Feature importance
bandsCoef25	0.9984
bandsCoef5	0.9953
SpecSpread	0.9938
bandsCoef1	0.9922
bandsCoef24	0.9922
MFCC3	0.9769
LogSpecSpread	0.9750
MFCC2	0.9722
HamoPk15	0.9697
bandsCoef11	0.9659
bandsCoef8	0.9647
HamoPk21	0.9644
MFCC4	0.9606
MFCC8	0.9581
bandsCoef23	0.9563
HamoPk19	0.9534
HamoPk20	0.9500

Table 8.11(b): Features which were selected less than 10% of the time

Feature name	Feature importance
prj8	0.0609
prj5	0.0622
prj10	0.0656
prj22	0.0688
prj14	0.0688
prj11	0.0688
prj18	0.0703
prj13	0.0712
prj12	0.0725
prj19	0.0734
prj17	0.0734
prj16	0.0734
prj6	0.0734
prj20	0.0766
prj15	0.0772
prj21	0.0781
prj23	0.0781
prj4	0.0845
prj24	0.0859
prj9	0.0888
prj26	0.0938
prj25	0.0938

Flatness coefficients clearly are important for instrument recognition in this case, whereas projection coefficients have little relevance.

Tables 8.11(a) and (b) list overall importance of features; however, it could be useful to consider feature importance per label as this will give an indication as to which features are important in the identification of which instruments. The following tables show the calculated feature importance per label (instrument). The table cells are shaded in such a way that darker colours correspond to higher feature importance. Tables are presented for different groups of features.

Table 8.12(a): Feature importance per label - MFCCs

	MFCC1	MFCC2	MFCC3	MFCC4	MFCC5	MFCC6	MFCC7	MFCC8	MFCC9	MFCC10	MFCC11	MFCC12	MFCC13
Accordion	1.0000	1.0000	1.0000	0.6413	0.8663	0.9706	1.0000	1.0000	0.7081	1.0000	1.0000	0.9634	1.0000
Bassoon	1.0000	1.0000	1.0000	1.0000	0.9988	0.0359	0.9388	0.9931	0.3856	0.0719	0.4522	0.3419	0.9844
Cello	0.7278	0.9244	0.8844	0.8178	1.0000	0.9659	1.0000	1.0000	0.8863	0.9994	1.0000	0.0903	0.7631
Clarinet	0.9481	1.0000	0.0256	1.0000	1.0000	1.0000	0.4731	1.0000	0.1047	0.5516	0.1363	0.9481	0.0381
Double Bass	1.0000	0.9644	1.0000	0.1078	0.7694	0.3597	1.0000	0.8578	0.1991	1.0000	1.0000	1.0000	1.0000
English Horn	0.0022	0.0344	0.2453	0.9972	1.0000	1.0000	1.0000	1.0000	0.9984	0.8894	0.6922	0.3222	0.9334
Flute	0.8609	0.0625	0.8816	1.0000	0.9884	0.7319	0.3863	0.4472	0.5897	0.2941	0.4319	0.4328	0.1281
French Horn	1.0000	0.9984	0.9984	1.0000	1.0000	0.8925	0.2228	0.0000	0.9588	0.0125	0.9581	0.3078	1.0000
Guitar	0.2666	1.0000	1.0000	1.0000	0.4231	1.0000	0.1256	0.9659	0.2269	0.8156	0.1603	0.4428	1.0000
Oboe	0.3075	0.9984	1.0000	1.0000	0.0919	0.6253	1.0000	1.0000	1.0000	0.9963	1.0000	0.2163	0.0016
Piano	0.0600	1.0000	1.0000	1.0000	0.2072	1.0000	0.0209	1.0000	0.2259	1.0000	0.0194	1.0000	1.0000
Piccolo	1.0000	1.0000	1.0000	0.1384	0.9994	0.2522	0.3388	0.8828	1.0000	0.9813	0.8222	0.8816	0.5156
Saxophone	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9994	0.0863	0.5931	1.0000	0.0356	1.0000	0.9100
SynthBass	1.0000	0.8416	1.0000	1.0000	0.9984	1.0000	0.0178	1.0000	0.6338	1.0000	0.9841	0.9994	0.6572
Trombone	1.0000	0.9906	1.0000	1.0000	1.0000	1.0000	0.0519	0.6438	0.9847	1.0000	1.0000	0.3509	1.0000
Trumpet	0.9963	1.0000	1.0000	1.0000	1.0000	1.0000	0.9881	1.0000	1.0000	0.8316	0.9894	0.8753	0.0347
Tuba	1.0000	1.0000	1.0000	0.1628	1.0000	0.0219	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Viola	1.0000	1.0000	0.4731	1.0000	1.0000	1.0000	0.7909	1.0000	0.3778	0.9984	0.1294	0.9644	0.8344
Violin	1.0000	0.5547	0.9975	1.0000	0.9591	0.1875	0.9944	0.9178	1.0000	0.9275	0.3044	0.9972	1.0000
Electric Guitar	1.0000	1.0000	0.9769	1.0000	1.0000	0.4641	1.0000	0.9984	1.0000	0.7619	0.3428	0.4978	1.0000
Marimba	1.0000	1.0000	0.9675	1.0000	1.0000	0.4431	1.0000	0.9969	1.0000	0.7644	0.3513	0.5156	1.0000
Vibraphone	1.0000	0.9172	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	0.5869	1.0000	1.0000
Acoustic Bass	0.5044	1.0000	1.0000	1.0000	0.0000	1.0000	0.0628	1.0000	0.1306	1.0000	0.0125	1.0000	1.0000

Table 8.12(b): Feature importance per label – Spectral and other features

	Temporal Centroid	Log Spectral Centroid	Log Spectral Spread	Energy	Zero Crossings	Spectral Centroid	Spectral Spread	Rolloff	Log Attack Time
Accordion	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9856
Bassoon	1.0000	1.0000	0.9969	0.6119	1.0000	1.0000	1.0000	1.0000	1.0000
Cello	0.9919	0.1050	1.0000	0.4819	0.1738	0.0122	1.0000	0.9719	0.3294
Clarinet	1.0000	0.5156	0.9984	0.0006	0.0831	0.4778	0.9941	0.8788	0.9478
Double Bass	1.0000	0.0306	1.0000	1.0000	0.2019	0.9050	1.0000	0.7728	0.4544
English Horn	0.0006	1.0000	1.0000	0.0006	0.5728	0.2631	1.0000	0.0534	0.7397
Flute	1.0000	0.1619	1.0000	0.0059	0.4594	0.3091	1.0000	0.9972	0.6472
French Horn	1.0000	0.3503	1.0000	0.5266	0.5788	0.0509	1.0000	0.4944	0.0850
Guitar	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Oboe	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.5109	0.9491	1.0000
Piano	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Piccolo	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.3681	0.8041	0.1956
Saxophone	0.0091	1.0000	0.0650	0.1113	0.8725	0.9994	0.9969	1.0000	1.0000
SynthBass	0.5944	1.0000	0.5091	0.0481	1.0000	1.0000	1.0000	1.0000	1.0000
Trombone	1.0000	1.0000	1.0000	0.0100	1.0000	0.0269	1.0000	0.0775	1.0000
Trumpet	1.0000	1.0000	1.0000	0.0734	1.0000	1.0000	1.0000	1.0000	0.0213
Tuba	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Viola	0.9909	0.0250	0.9816	0.1597	0.9834	0.2063	1.0000	1.0000	0.9938
Violin	0.9994	0.3000	0.5753	0.9972	0.9453	0.3741	0.9288	0.6131	0.9791
Electric Guitar	0.0253	1.0000	1.0000	0.1338	1.0000	1.0000	1.0000	1.0000	1.0000
Marimba	0.0178	1.0000	1.0000	0.1300	1.0000	1.0000	1.0000	1.0000	1.0000
Vibraphone	1.0000	0.9553	0.0225	0.9972	0.0247	0.9928	0.0056	0.0313	0.0163
Acoustic Bass	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 8.12(c)(i): Feature importance per label – Flatness coefficients (first 17)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Accordion	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0506	1.0000	1.0000	1.0000	0.1997	1.0000	0.9666	0.9919	0.6350
Bassoon	1.0000	1.0000	1.0000	0.8756	0.8841	1.0000	1.0000	0.9953	0.4572	1.0000	1.0000	1.0000	0.0919	1.0000	0.9522	1.0000	1.0000
Cello	0.5400	0.9563	1.0000	0.2069	0.1425	1.0000	0.9994	0.9081	0.0806	0.5241	0.9916	0.1303	1.0000	1.0000	0.9941	0.9197	0.2756
Clarinet	0.9769	0.0263	0.0072	1.0000	1.0000	0.1853	0.0497	1.0000	0.3675	0.2509	1.0000	0.9400	1.0000	1.0000	0.0347	1.0000	1.0000
Double Bass	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9906	0.9947	1.0000	1.0000	1.0000	0.0606	1.0000	0.4616	0.9622	1.0000
English Horn	1.0000	0.0097	0.0000	1.0000	1.0000	0.0566	0.1572	0.9953	1.0000	0.3088	0.1600	0.4850	0.9994	0.9169	0.3719	1.0000	0.3503
Flute	1.0000	0.7891	0.4553	0.7334	0.9734	0.1622	0.2300	0.8153	1.0000	0.6834	0.5216	0.9981	1.0000	0.7450	0.9934	0.9647	0.8994
French Horn	1.0000	0.7925	0.8409	0.5722	0.5206	0.9781	0.9575	0.1719	1.0000	0.9984	0.9613	1.0000	0.3241	1.0000	0.8338	1.0000	0.4013
Guitar	0.9238	1.0000	1.0000	1.0000	0.9956	1.0000	1.0000	1.0000	0.9709	1.0000	0.9069	1.0000	1.0000	0.2338	1.0000	0.1128	1.0000
Oboe	0.9875	0.0513	0.0006	1.0000	1.0000	0.0000	1.0000	0.9850	0.9984	0.9128	1.0000	0.9903	1.0000	1.0000	0.2431	0.2000	1.0000
Piano	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9978	0.6878	1.0000	0.9878	0.9944	0.7172	0.7434	1.0000	0.1066	1.0000
Piccolo	1.0000	0.7741	0.8466	0.9903	1.0000	0.0447	0.9994	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.4400	1.0000	0.6209	0.9984
Saxophone	0.0431	1.0000	1.0000	0.9972	0.8559	1.0000	1.0000	1.0000	0.8503	0.1125	0.3613	0.0591	1.0000	0.8763	0.9931	0.0813	1.0000
SynthBass	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.4828	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Trombone	1.0000	1.0000	0.9688	1.0000	1.0000	0.4300	0.8791	0.8625	0.0225	0.4213	1.0000	0.9922	1.0000	0.1919	1.0000	0.3700	1.0000
Trumpet	0.1994	0.9681	1.0000	0.9794	1.0000	1.0000	0.9628	1.0000	1.0000	0.2850	1.0000	0.0600	0.5150	0.9547	0.0906	1.0000	1.0000
Tuba	1.0000	0.2578	0.9766	0.9413	0.9866	1.0000	1.0000	1.0000	0.9784	0.9994	0.6534	1.0000	1.0000	0.8563	1.0000	1.0000	0.3759
Viola	1.0000	0.6216	0.6534	1.0000	1.0000	0.0063	0.0291	1.0000	0.6669	0.0800	1.0000	0.2525	0.9994	1.0000	0.2575	1.0000	1.0000
Violin	1.0000	0.1213	0.4181	1.0000	1.0000	0.0094	0.1756	0.3531	0.1053	0.0422	0.9816	0.3169	0.8578	0.9438	0.5241	0.3784	0.9434
Electric Guitar	0.9956	1.0000	1.0000	0.0275	0.9678	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.5941	1.0000	1.0000	1.0000	1.0000
Marimba	0.9956	1.0000	1.0000	0.0250	0.9622	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.5803	1.0000	0.9972	1.0000	1.0000	1.0000
Vibraphone	1.0000	0.5466	0.2778	1.0000	1.0000	0.0000	0.0000	0.9994	1.0000	0.0000	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	0.7184
Acoustic Bass	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0094	0.0428	0.0000	0.7463	0.0050	0.0000	0.1259	1.0000	0.9984	1.0000

Table 8.12(c)(ii): Feature importance per label – Flatness coefficients (18-33)

	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Accordion	0.9984	1.0000	0.9909	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8206	0.7381	0.1159	1.0000
Bassoon	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.7694
Cello	0.6863	0.0572	1.0000	0.6884	0.0275	0.0163	0.5994	1.0000	0.9981	0.8097	0.2834	0.1541	0.9978	0.8078	0.2600	0.0553
Clarinet	0.5822	0.0284	1.0000	1.0000	0.9953	1.0000	1.0000	0.9984	1.0000	1.0000	0.9919	0.7544	0.2228	0.0225	0.8034	0.0084
Double Bass	0.9263	1.0000	0.0094	0.0028	0.3081	0.9994	1.0000	1.0000	1.0000	1.0000	1.0000	0.4009	0.2003	0.6231	0.8966	1.0000
English Horn	0.8444	0.3347	0.7069	0.4759	0.3713	0.1825	0.9531	0.9059	0.5159	0.2459	0.2553	0.9647	1.0000	1.0000	1.0000	0.8756
Flute	0.8463	0.9094	0.7416	0.8622	0.2941	0.8881	0.4275	0.4941	0.0944	0.0344	0.8759	1.0000	0.9994	0.9978	0.8081	0.9375
French Horn	0.1119	0.1300	0.1828	0.8644	0.8981	1.0000	1.0000	1.0000	1.0000	1.0000	0.9916	0.7603	0.2288	0.0241	0.3781	
Guitar	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9984	0.5928	0.0491	0.4041	0.9475	0.9978	1.0000
Oboe	1.0000	0.9884	0.5584	0.3416	1.0000	1.0000	1.0000	0.2444	0.9234	0.2363	0.0634	0.9741	1.0000	1.0000	1.0000	0.7616
Piano	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8822	0.0806	0.9881	1.0000	1.0000
Piccolo	1.0000	1.0000	0.5800	0.2003	0.9869	0.6997	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Saxophone	0.9694	0.9994	1.0000	1.0000	1.0000	1.0000	0.0516	1.0000	0.9903	1.0000	0.9913	0.0169	0.8797	1.0000	1.0000	1.0000
SynthBass	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Trombone	1.0000	1.0000	0.3728	1.0000	0.0259	0.1509	1.0000	0.8466	0.1303	0.8516	0.2341	0.3931	0.9400	0.9950	1.0000	0.8822
Trumpet	0.3513	0.9934	1.0000	1.0000	1.0000	1.0000	1.0000	0.9866	0.8369	0.0519	0.9175	0.9969	1.0000	1.0000	1.0000	1.0000
Tuba	1.0000	1.0000	0.9288	0.5797	1.0000	1.0000	1.0000	0.9984	0.2881	1.0000	0.0006	1.0000	1.0000	0.0078	0.0256	0.0988
Viola	0.2325	0.0944	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0331	0.1291	0.8075	0.0666	0.0681
Violin	0.1628	0.4403	0.6106	0.9941	0.2944	0.8831	1.0000	0.9984	0.9988	1.0000	0.9263	1.0000	0.9594	0.8922	1.0000	1.0000
Electric Guitar	0.9728	0.0219	0.3906	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9816	0.0588	0.1522	1.0000
Marimba	0.9766	0.0159	0.3753	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9841	0.0541	0.1316	1.0000
Vibraphone	1.0000	1.0000	0.9866	1.0000	0.0072	0.9694	1.0000	0.9491	0.3941	0.2563	0.4694	0.0000	0.9747	1.0000	0.0000	0.0006
Acoustic Bass	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9941	0.0356	0.8013	0.9984	1.0000	1.0000

Table 8.12(d)(i): Feature importance per label – Projection coefficients (first 17)

	prj1	prj2	prj3	prj4	prj5	prj6	prj7	prj8	prj9	prj10	prj11	prj12	prj13	prj14	prj15	prj16	prj17
Accordion	0.6431	1.0000	0.2697	0.1531	0.2028	0.2269	0.2169	0.2325	0.2738	0.2109	0.2013	0.1784	0.1291	0.1231	0.0834	0.0700	0.0400
Bassoon	0.6994	0.0000	0.0156	0.0094	0.0147	0.0119	0.0144	0.0088	0.0081	0.0025	0.0034	0.0075	0.0000	0.0006	0.0022	0.0000	0.0006
Cello	0.4025	0.0622	0.3503	0.1344	0.0766	0.1063	0.1019	0.0816	0.0975	0.0978	0.1494	0.1178	0.1241	0.1694	0.3431	0.3325	0.3875
Clarinet	0.6869	0.7619	0.0897	0.0241	0.0284	0.0463	0.0450	0.0425	0.0563	0.0325	0.0325	0.0250	0.0328	0.0166	0.0181	0.0097	0.0131
Double Bass	0.7703	0.9984	0.2534	0.1484	0.1778	0.2075	0.1619	0.2309	0.2566	0.2150	0.2431	0.1888	0.1863	0.2475	0.2350	0.3281	0.3228
English Horn	0.6744	0.0644	0.0256	0.0322	0.0347	0.0453	0.0391	0.0281	0.0359	0.0213	0.0181	0.0238	0.0191	0.0138	0.0091	0.0100	0.0103
Flute	0.6934	0.1172	0.0066	0.0216	0.0275	0.0366	0.0294	0.0278	0.0478	0.0325	0.0288	0.0275	0.0341	0.0247	0.0309	0.0216	0.0169
French Horn	0.0022	0.9713	0.1013	0.1009	0.0613	0.0428	0.0378	0.0156	0.0159	0.0050	0.0103	0.0006	0.0034	0.0031	0.0000	0.0006	0.0022
Guitar	0.6625	0.3772	0.2563	0.1288	0.0809	0.0809	0.0719	0.0459	0.0381	0.0156	0.0191	0.0275	0.0275	0.0263	0.0066	0.0069	0.0038
Oboe	0.6819	0.5178	0.2666	0.0738	0.0569	0.0797	0.0672	0.0431	0.0644	0.0569	0.0600	0.0559	0.0584	0.0316	0.0234	0.0203	0.0063
Piano	0.6869	0.7844	0.0731	0.3644	0.1228	0.1091	0.8200	0.1306	0.3859	0.1213	0.1900	0.1497	0.1700	0.0841	0.0947	0.1044	0.0669
Piccolo	0.6950	0.0019	0.0266	0.0156	0.0353	0.0328	0.0481	0.0447	0.0538	0.0328	0.0381	0.0484	0.0366	0.0294	0.0409	0.0413	0.0259
Saxophone	0.6863	0.5319	0.0478	0.0731	0.0369	0.0347	0.0850	0.0428	0.0309	0.0388	0.0659	0.0197	0.0234	0.0209	0.0122	0.0172	0.0200
SynthBass	0.6869	0.9984	0.2994	0.2059	0.2253	0.3025	0.2766	0.3166	0.2853	0.2369	0.2459	0.2244	0.2416	0.2044	0.1906	0.1566	0.1222
Trombone	0.8425	0.0241	0.0847	0.0503	0.0247	0.0222	0.0438	0.0241	0.0147	0.0091	0.0078	0.0038	0.0038	0.0000	0.0013	0.0006	0.0016
Trumpet	0.0828	0.8141	0.1094	0.0847	0.0631	0.0616	0.0622	0.0628	0.0366	0.0631	0.0413	0.0394	0.0278	0.0206	0.0113	0.0100	0.0119
Tuba	0.6641	0.3650	0.0466	0.0578	0.0244	0.0188	0.0488	0.0097	0.0253	0.0106	0.0219	0.0169	0.0113	0.0019	0.0013	0.0000	0.0000
Viola	0.5309	0.2466	0.1122	0.0525	0.0584	0.1094	0.1972	0.1859	0.2813	0.3603	0.3813	0.4466	0.5084	0.5869	0.5663	0.7584	0.7169
Violin	0.1572	0.2956	0.2241	0.1106	0.1444	0.1941	0.2263	0.2241	0.2503	0.2775	0.3375	0.4056	0.3828	0.3500	0.4022	0.3009	0.3559
Electric Guitar	0.6756	0.7438	0.5872	0.2697	0.1759	0.1453	0.0694	0.0256	0.0313	0.0138	0.0138	0.0100	0.0022	0.0022	0.0022	0.0000	0.0016
Marimba	0.6803	0.7350	0.5753	0.2675	0.1766	0.1478	0.0672	0.0266	0.0247	0.0125	0.0131	0.0084	0.0031	0.0016	0.0022	0.0000	0.0016
Vibraphone	0.0075	0.1113	0.1800	0.1219	0.1038	0.1394	0.0903	0.0528	0.0544	0.0728	0.0559	0.0681	0.0541	0.0425	0.0538	0.0300	0.0309
Acoustic Bass	0.6869	0.9919	0.0631	0.4875	0.1872	0.0584	0.9675	0.1375	0.5303	0.1113	0.2397	0.1725	0.1128	0.0803	0.1138	0.0713	0.0297

Table 8.12(d)(ii): Feature importance per label – Projection coefficients (18-33)

	prj18	prj19	prj20	prj21	prj22	prj23	prj24	prj25	prj26	prj27	prj28	prj29	prj30	prj31	prj32	prj33
Accordion	0.0253	0.0184	0.0072	0.0081	0.0000	0.0000	0.0013	0.0016	0.0006	0.0006	0.0028	0.0044	0.0044	0.1175	0.0128	0.1797
Bassoon	0.0006	0.0016	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Cello	0.3566	0.4097	0.5425	0.5731	0.5447	0.6088	0.6822	0.6047	0.7175	0.7728	0.7309	0.7634	0.7944	0.8616	0.8763	0.7628
Clarinet	0.0013	0.0031	0.0016	0.0031	0.0472	0.0000	0.0038	0.0275	0.0041	0.0028	0.0034	0.0022	0.0000	0.0013	0.0000	0.1263
Double Bass	0.3984	0.4547	0.4422	0.4147	0.5491	0.6359	0.6272	0.7753	0.7994	0.7994	0.7894	0.6584	0.5091	0.7638	0.6534	0.3753
English Horn	0.0034	0.0000	0.0000	0.0000	0.0000	0.0000	0.0016	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Flute	0.0113	0.0166	0.0072	0.0044	0.0122	0.0022	0.0050	0.0059	0.0047	0.0006	0.0022	0.0000	0.0000	0.0000	0.0000	0.0000
French Horn	0.0306	0.0056	0.0016	0.2119	0.1238	0.0113	0.0675	0.1753	0.1309	0.1544	0.1834	0.2981	0.0363	0.1356	0.0888	0.1119
Guitar	0.0016	0.0006	0.0006	0.0006	0.0028	0.0013	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Oboe	0.0028	0.0028	0.0000	0.0013	0.0006	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0063
Piano	0.0463	0.0619	0.0431	0.0338	0.0481	0.0250	0.0197	0.0150	0.0131	0.0047	0.0019	0.0013	0.0013	0.0019	0.0013	0.0278
Piccolo	0.0281	0.0106	0.0138	0.0119	0.0056	0.0072	0.0013	0.0050	0.0084	0.0013	0.0044	0.0000	0.0000	0.0000	0.0000	0.0000
Saxophone	0.0147	0.0081	0.0166	0.0459	0.0241	0.0309	0.0966	0.2150	0.2759	0.3753	0.4103	0.7328	0.4631	0.0650	0.6041	0.9966
SynthBass	0.0863	0.0663	0.0431	0.0250	0.0269	0.0128	0.0125	0.0050	0.0013	0.0059	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Trombone	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0047	0.0303	0.0000	0.0378	0.0156	0.1519	0.0650	0.0275	0.0119	0.1241
Trumpet	0.0000	0.0047	0.0022	0.0031	0.0006	0.0016	0.0388	0.0056	0.0147	0.0322	0.0478	0.2950	0.0809	0.0091	0.1741	0.8903
Tuba	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0897
Viola	0.7831	0.7931	0.7469	0.7578	0.7200	0.5531	0.7469	0.7922	0.7531	0.8478	0.7213	0.8053	0.8038	0.8531	0.8953	0.7750
Violin	0.3781	0.3775	0.3647	0.3738	0.3422	0.4313	0.4297	0.3909	0.3772	0.4569	0.6275	0.9219	0.7803	0.5047	0.6763	0.9984
Electric Guitar	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0072
Marimba	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0075
Vibraphone	0.0169	0.0113	0.0056	0.0063	0.0050	0.0013	0.0016	0.0000	0.0000	0.0019	0.1481	0.8766	0.7666	0.9334	0.5984	1.0000
Acoustic Bass	0.0297	0.0056	0.0088	0.0041	0.0022	0.0006	0.0006	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 8.12(e)(i): Feature importance per label – Harmonic peaks (first 14)

	HP1	HP2	HP3	HP4	HP5	HP6	HP7	HP8	HP9	HP10	HP11	HP12	HP13
Accordion	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Bassoon	0.2141	0.3522	0.6841	0.5272	0.1956	0.4731	0.1819	0.3253	0.2403	0.3119	0.1850	0.1400	0.2113
Cello	0.9944	0.4509	0.2453	0.5406	0.0447	0.0888	0.2234	0.1081	0.0600	0.3878	0.0784	0.0463	0.0428
Clarinet	1.0000	0.6894	1.0000	0.1097	0.0059	1.0000	0.0072	1.0000	0.6659	1.0000	0.4934	0.8109	0.9713
Double Bass	1.0000	0.5278	0.9672	0.4591	1.0000	0.6803	0.9919	0.9925	1.0000	0.9994	1.0000	0.9969	0.9956
English Horn	0.0378	0.0847	0.0228	0.0213	0.1538	0.1644	0.2806	0.1000	0.2759	0.4425	0.7825	0.7591	0.8797
Flute	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
French Horn	0.2613	0.3681	0.1163	0.2522	0.3809	0.4150	0.6638	0.5916	0.8803	0.9506	0.9984	0.9981	1.0000
Guitar	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Oboe	0.7850	0.1838	0.9938	0.9966	0.5025	0.0119	0.0850	0.3769	0.4563	0.7359	0.0459	0.0541	0.0784
Piano	0.5309	0.8628	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Piccolo	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Saxophone	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
SynthBass	0.8800	0.9750	0.0000	0.0000	0.0000	0.2941	0.1594	0.9859	0.9894	0.9984	0.9600	1.0000	0.9994
Trombone	1.0000	1.0000	1.0000	0.9969	0.9938	0.9631	1.0000	0.5716	0.8288	0.0838	0.9184	0.9747	1.0000
Trumpet	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Tuba	0.8359	1.0000	0.0341	0.4103	0.9541	0.0697	0.1266	0.1094	0.3981	0.6363	0.9934	0.9472	0.9972
Viola	1.0000	0.9950	1.0000	0.0188	0.0013	1.0000	0.0397	0.9941	0.7397	0.9984	0.7838	0.9806	1.0000
Violin	1.0000	1.0000	0.8763	0.3066	0.2741	0.0784	0.0509	0.2463	0.2716	0.0613	0.0434	0.1800	0.1575
Electric Guitar	0.2009	0.0000	0.9291	0.6453	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Marimba	0.1969	0.0006	0.9303	0.6416	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Vibraphone	1.0000	0.1044	0.0616	0.0000	0.0034	0.0703	0.4813	0.9963	0.0000	0.1422	0.9528	0.1191	1.0000
Acoustic Bass	0.3994	0.1794	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9697	1.0000

Table 8.12(e)(ii): Feature importance per label – Harmonic peaks (15-28)

	HP14	HP15	HP16	HP17	HP18	HP19	HP20	HP21	HP22	HP23	HP24	HP25	HP26	HP27	HP28
Accordion	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Bassoon	0.1388	0.4784	0.1413	0.1997	0.1428	0.4075	0.1644	0.2022	0.1441	0.2931	0.1247	0.2513	0.1309	0.3653	0.1422
Cello	0.2544	0.7169	0.2753	0.7288	0.0963	0.9984	0.9453	0.9981	0.2578	0.8563	1.0000	0.8994	0.9269	1.0000	0.9950
Clarinet	0.9400	0.8297	0.8472	0.2119	0.9913	0.7584	0.9731	0.8884	0.8634	0.5184	0.2213	0.5484	0.9969	0.1678	0.5672
Double Bass	1.0000	0.9972	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
English Horn	0.7050	0.9613	0.6119	0.9347	0.6666	0.9813	0.7428	0.8150	0.6275	0.9541	0.6572	0.9341	0.7347	0.9853	0.6053
Flute	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
French Horn	0.9959	1.0000	0.9903	1.0000	0.9941	1.0000	0.9669	0.9963	0.9713	1.0000	0.9088	0.9959	0.9309	0.9959	0.9259
Guitar	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9988	1.0000	0.9925	1.0000	0.9781	1.0000	0.9556
Oboe	0.4703	0.7897	0.1013	0.8956	0.0572	1.0000	0.9988	1.0000	0.6056	0.9644	1.0000	0.9681	0.9803	1.0000	1.0000
Piano	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.5213	1.0000	1.0000	1.0000
Piccolo	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Saxophone	1.0000	1.0000	0.8481	1.0000	0.2266	1.0000	0.1297	0.6016	0.8294	1.0000	0.0688	0.1931	0.1766	0.9753	0.8313
SynthBass	1.0000	0.9800	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Trombone	1.0000	0.9506	1.0000	0.8500	1.0000	0.1875	1.0000	0.4506	1.0000	0.0656	1.0000	0.0881	1.0000	0.2163	1.0000
Trumpet	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9628	1.0000	0.9963	1.0000	1.0000	0.9800	0.9709	1.0000	0.4413
Tuba	0.0700	0.9031	0.0544	0.3684	0.1072	0.4756	0.4084	1.0000	1.0000	1.0000	1.0000	1.0000	0.9988	1.0000	1.0000
Viola	0.9988	0.9931	0.9909	0.6872	1.0000	0.9750	1.0000	0.9766	0.9631	0.2372	0.0650	0.2659	0.9884	0.1044	0.3563
Violin	0.6644	0.1372	0.4522	0.0716	0.2881	0.0534	0.5616	0.3241	0.6147	0.4072	0.0825	0.9872	0.0978	0.6572	0.4284
Electric Guitar	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Marimba	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Vibraphone	1.0000	0.5491	0.9878	0.8841	0.9578	0.9322	0.9913	0.9978	0.9988	0.1156	0.1569	0.7766	0.6850	0.0678	0.0556
Acoustic Bass	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.1613	1.0000	1.0000	1.0000

Some clear patterns emerge when studying the preceding tables. For example,

- The first 5 MFCCs are important for most instruments, as are the 6th and 8th MFCCs (Table 8.12(a)).
- In some instances, similarities between instrument families can be detected. For instance, the Log Spectral Centroid is important for most instruments, but for all 4 string instruments present in the dataset (Violin, Viola, Cello and Double Bass) it is not really considered relevant (Table 8.12(b)). Similarly, projection coefficients (especially from coefficient 3 onwards) are not really relevant for any instruments other than strings (Table 8.12(d)(i) and (ii)).
- Every Harmonic Peak is relevant for the Accordion, yet for the Bassoon almost none of them are strongly relevant (Table 8.12(e)(i) and (ii)).

The above tables clearly suggest that feature selection at an instrument level may be a useful area to explore for instrument recognition, since different instruments seem to attach different relevances to different features. An interesting consequence of this result can be to engineer feature selection in such a way as to prevent classification errors due to instruments which are often confused.

8.6 Summary

In this chapter we conducted an empirical study of our proposed feature selection technique using musical instrument data. Musical instrument recognition poses unique challenges, especially when the aim is to identify instruments playing in polyphony. We started by explaining the lack of available data for instrument recognition studies, and then proceeded to define and describe the dataset used for this study. We subsequently described the methodology followed in this empirical study, and presented and discussed the results in detail, with a particular emphasis on the results of feature selection. This study has shown that there is a definite case to be made for feature selection, since comparable results can be obtained by using about 25% less features. The proposed multi-label feature selection technique also allowed for interesting comparisons of feature importance per label (instrument).

CHAPTER 9

Conclusions

9.1 Summary

In this thesis we presented an introduction to the field of music information retrieval (MIR), which is a field at the crossroads of music and the mathematical sciences such as statistics and computer science. We have specifically focused on statistical contributions to the field of MIR, with a discussion of early applications of statistics to music. We further discussed the ways in which features can be extracted from digital audio data (particularly music), and discussed some of the frequently utilised features for analysing musical data. We also discussed some of the main sub-fields of MIR, with a particular focus on classification, and considered the classifiers used in such classification problems.

We moved on to the problem of musical instrument recognition, which is one of the main classification problems in the MIR field. We outlined the problem, and defined the scope and complexities involved in such problems. We particularly highlighted the challenges presented by the lack of benchmark datasets in this regard. We again considered the classifiers commonly used for instrument recognition problems, and

also briefly mentioned the field of multi-label learning which is increasingly being used for MIR problems. Some of the main previous approaches to instrument recognition problems were discussed, with a specific focus on polyphonic problems.

The field of multi-label learning was formally defined in Chapter 4, where we presented a categorisation of multi-label methods. The different multi-label learning algorithms within each of these categories were discussed in some detail, and the relative merits of each were discussed. We explained that multi-label methods require unique ways of measuring classification error, and defined the different measures often used in this regard. We also defined the concepts label cardinality and label density, and listed some of the oft-encountered multi-label benchmark datasets. These benchmark datasets are often limited in terms of certain data characteristics such as label cardinality, and are therefore often not ideal for carrying out extensive comparative studies in the multi-label field. This suggests that there is a clear place for multi-label simulation studies, in order to objectively compare multi-label methods over a variety of data characteristics. However, relatively little has been published in this regard and there appears to be no standard method of generating synthetic multi-label data. Herein lies one of the main contributions of this thesis, as in Chapter 6 we present a novel – yet simple – way of simulating multi-label data while controlling many aspects of data characteristics.

We have mentioned the challenges presented by large datasets. One way of reducing the complexity of large datasets, is through feature selection. In Chapter 5 we discussed the problem of feature selection at length, and illustrated the difficulties introduced by working with high-dimensional data. We outlined some general approaches to feature selection, and then examined suggestions encountered in the literature for feature selection in a multi-label context. Again, relatively little has been published regarding feature selection for multi-label learning problems, so as a way of bridging this gap we introduced a novel way of feature selection for multi-label problems, based on the idea of independent probe variables. This technique is very intuitive and easy to implement and delivered promising results in empirical studies.

Results of the empirical studies are presented in Chapters 7 and 8. In Chapter 7, we first considered the results of the empirical multi-label simulation study. We found

that the number of labels as well as the presence or absence of correlations between labels have a significant effect on classification accuracy. We also illustrated the effectiveness of our proposed feature selection method by showing that irrelevant features are clearly identified and can therefore be eliminated without negatively impacting on classification accuracy.

Similar results were obtained in the empirical instrument recognition study; that is, the proposed feature selection method is clearly effective in identifying relevant features and therefore reducing the complexity of the dataset without negatively impacting on performance. In this chapter (Chapter 8) we also illustrated the complexity of musical instrument data. We again lamented the lack of benchmark data in order to objectively compare algorithms and techniques, but nevertheless found that our classification results appeared promising, given the complexities of the problem.

9.2 Directions for further research

9.2.1 Feature selection

While our proposed feature selection technique performed well in the empirical studies, there are still some open questions that need to be addressed in further research. For instance, it might be useful to perform feature selection separately for each label in a multi-label problem; in fact, the results from the musical instrument recognition study in Chapter 8 seem to back this up, since it was clearly seen that different features are relevant for different instruments. A different measure of dependence between the labels and the features (other than the correlation coefficient, which is a very naive measure) could also be considered. We also still need to find a way to determine the optimum values for the selection parameters from the data; cross-validation could be one way of accomplishing this.

9.2.2 Simulating multi-label data

The technique we proposed for simulating multi-label data performed well in our empirical study. However, there appears to be some complex interactions taking place in the simulation process which are somewhat unexpected and require further investigation. For instance, when generating labels, the obtained correlations are somewhat diluted compared to the correlations specified in the simulation parameters. This did not present a problem for our study, but some further research could involve a detailed study of the way label correlations manifest themselves and the effect of this on classification. For instance, in the simulation study we obtained a somewhat unexpected result, in that classification results are better in the presence of label correlations than in the absence thereof, given that the multi-label algorithm used does not take label correlations into account. We offered some possible explanations for this in Section 7.3.6. It also appeared that there is interaction between the feature selection process and label correlations; again, this is an unexpected result which warrants further investigation. It therefore appears as if the matter of label correlations is a complex one, and a detailed simulation study should be carried out to investigate the different effects of label correlations. However, this falls outside of the scope of this thesis.

9.2.3 Musical instrument recognition

An extension of our work on the musical instrument recognition problem could be to attempt to replicate results using different datasets. Ideally, we would like to be able to utilise real-world recordings as well. A further natural extension would be to attempt classification in cases where there are more than two instruments playing simultaneously, as this will also introduce additional complexity into the problem. Our results on relevant features by instrument could be useful in this regard, as another obvious next step could be to perform feature selection separately for each instrument. It could also be useful to engineer feature selection in such a way as to prevent classification errors due to instruments which are often confused (for instance, the violin and viola).

References

- Agus, T.R., Suied, C., Thorpe, S.J. and Pressnitzer, D. (2012). Fast recognition of musical sounds based on timbre. *Journal of the Acoustical Society of America*, **131**(5), 4124-4133.
- Allen, J.B. and Rabiner, L.R. (1977). A Unified Approach to Short-Time Fourier Analysis and Synthesis. In *Proceedings of the IEEE*, **65**(11), 1558-1564.
- Alm, J.F. and Walker, J.S. (2002). Time-Frequency Analysis of Musical Instruments. *Society for Industrial and Applied Mathematics Review*, **44**(3), 457-476.
- American Standards Association (1960). *American Standard Acoustical Terminology*. New York. Definition 12.9, Timbre, p.45.
- Aucouturier, J. and Pachet, F. (2002). Music similarity measures: What's the use? In *Proceedings of the 3rd International Society for Music Information Retrieval Conference (ISMIR 2002)*, Paris, France, pp. 157-163.
- Aucouturier, J. and Pachet, F. (2004). Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, **1**(1), 1-13.
- Baraniuk, R.G. (2011). More is Less: Signal Processing and the Data Deluge. *Science*, **331**, 717-719.
- Barbedo, J.G.A. (2011). Instrument Recognition. In *Music Data Mining*, Li, T., Ogihara, M. and Tzanetakis, G. (eds). CRC Press, Florida, pp. 95-134.
- Barbedo, J.G.A., Lopes, A. and Wolfe, P.J. (2009). Empirical methods to determine the number of sources in single-channel musical signals. *IEEE Transactions on Audio, Speech and Language Processing*, **17**(7), 1435-1444.

- Barbedo, J. and Tzanetakis, G. (2011). Musical instrument classification using individual partials. *IEEE Transactions on Audio, Speech and Language Processing*, **19**(1), 111-122.
- Barlow, C. (1980). Bus Journey to Parametron. *Feedback Papers*, **21-23**, 1-124.
- Bay, M., Ehmann, A., Beauchamp, J., Smaragdis, P. and Downie, J.S. (2012). Second fiddle is important too: Pitch Tracking Individual voices in polyphonic music. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, Porto, Portugal, pp. 319-324.
- Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M. and Sandler, M.B. (2005). A Tutorial on Onset Detection in Music Signals. *IEEE Transactions on Speech and Audio Processing*, **13**(5:2), 1035-1047.
- Benade, A.H. (1990). *Fundamentals of Musical Acoustics* (Second Revised Edition). Dover Publications, New York.
- Benetos, E. and Dixon, S. (2013). Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model. *Journal of the Acoustical Society of America*, **133**(3), 1727-1741.
- Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H. and Klapuri, A. (2012). Automatic Music Transcription: Breaking the Glass Ceiling. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, Porto, Portugal, pp. 379-384.
- Benetos, E., Kotti, M. and Kotropoulos, C. (2006). Musical instrument classification using non-negative matrix factorization algorithms. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006)*, Toulouse, France, pp. V-221-V-225.

Benetos, E., Kotti, M. and Kotropoulos, C. (2007). Large scale musical instrument identification. In *Proceedings of the 4th Sound and Music Computing Conference*, Lefkada, Greece, pp. 283-286.

Ben-Hur, A. and Weston, J. (2010). A User's Guide to Support Vector Machines. In *Data Mining Techniques for the Life Sciences*, Carugo, O. And Elsenhaber, F. (eds). Humana Press, New York, pp. 231-240.

Benson, D. (2008). *Music: A Mathematical Offering*. Online version (14 December 2008). Available at: <http://homepages.abdn.ac.uk/mth192/pages/html/music.pdf> [accessed 7 March 2013].

Beran, J. (2004). *Statistics in Musicology*. Chapman & Hall.

Bergstra, J., Casagrande, N., Erhan, D., Eck, D and Kégl, B. (2006). Aggregate Features and AdaBoost for Music Classification, *Machine Learning*, **65**(2-3), 473-484.

Bi, J., Bennett, K., Embrechts, M., Breneman, C. and Song, M. (2003). Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, **3**, 1229-1243.

Bibby, N. (2003). Tuning and Temperament: Closing the Spiral. In *Music and Mathematics: From Pythagoras to Fractals*. Fauvel, J., Flood, R. and Wilson, R. (eds). Oxford University Press, Oxford, pp. 13-28.

Blockeel, H., De Raedt, L. and Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 55-63.

Blockeel, H., Schietgat, L., Struyf, J., Dzeroski, S. and Clare, A. (2006). Decision trees for hierarchical multilabel classification: A case study in functional genomics. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Berlin, Germany, pp. 18-29.

Bosch, J.J., Janer, J., Fuhrmann, F. and Herrera, P. (2012). A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, Porto, Portugal, pp. 559-564.

Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5-32.

Brown, J.C., Houix, O. and McAdams, S. (2001). Feature dependence in the automatic identification of musical woodwind instruments. *Journal of the Acoustical Society of America*, **109**, 1064-1072.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, **2**(2), 121-167.

Burred, J.J., Robel, A. and Sikora, T. (2009). Polyphonic music information retrieval based on a dynamic model of the spectral envelope. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, Taipei, Taiwan, pp. 173-176.

Burred, J.J., Robel, A. and Sikora, T. (2010). Dynamic spectral envelope modelling for timbre analysis of musical sounds. *IEEE Transactions on Audio, Speech and Language Processing*, **18**(3), 663-674.

Cano, P., Batlle, E., Kalker, T. and Haitsma, J. (2005). A Review of Audio Fingerprinting. *Journal of VLSI Signal Processing*, **41**(3), 271-284.

Casey, M.A., Veltkamp, R.C., Goto, M., Leman, M., Rhodes, C. and Slaney, M. (2008). Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, **96**(4), 668-696.

Cemgil, A.T. and Gürgen, F. (1997). Classification of Musical Instrument Sounds Using Neural Networks. In *Proceedings of the 5th National Signal Processing and Applications Conference (SIU97)*, Bodrum, Turkey.

Chandrasekhar, V., Sharifi, M. and Ross, D.A. (2011). Survey and Evaluation of Audio Fingerprinting Schemes for Mobile Query-by-Example Applications. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, Miami, Florida, USA, pp. 801-806.

Chen, W., Yan, J., Zhang, B., Chen, Z. and Yang, Q. (2007). Document Transformation for Multi-label Feature Selection in Text Categorization. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, Omaha, Nebraska, USA, pp. 451-456.

Cheng, H., Yang, Y., Lin, Y., Liao, I. and Chen, H. (2008). Automatic Chord Recognition for Music Classification and Retrieval. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2008)*, Hannover, Germany, pp. 1505-1508.

Cherman, E.A., Monard, M.C. and Metz, J. (2011). Multi-label Problem Transformation Methods: a Case Study. *CLEI Electronic Journal*, **14**(1).

Cherman, E.A., Metz, J. and Monard, M.C. (2012). Incorporating label dependency into the binary relevance framework for multi-label classification. *Expert Systems with Applications*, **39**(2), 1647-1655.

Chetry, N. and Sandler, M. (2009). Linear predictive models for musical instrument identification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Toulouse, France, pp. V-225-V-228.

Chou, T., Chen, W., Wang, S., Chang, K. and Chen, H. (2012). Real-time Polyphonic Score Following System. In *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo Workshops (ICMEW 2012)*, Melbourne, Australia, pp. 205-210.

Christensen, M.G. and Jakobsson, A. (2009). *Multi-pitch Estimation*. Synthesis Lectures on Speech and Audio Processing, **5**(1). Morgan & Claypool.

Christensen, M.G., Stoica, P., Jakobsson, A. and Jensen, S.H. (2008). Multi-pitch Estimation. *Signal Processing*, **88**(4), 972-983.

Clare, A. and King, R.D. (2001). Knowledge discovery in multi-label phenotype data. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 42-53.

Cont, A. (2010). A Coupled Duration-Focused Architecture for Real-Time Music-to-Score Alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(6), 974-987.

Cont, A., Echeveste, J., Jacquemard, F. and Giavitto, J. (2012). Correct Automatic Accompaniment Despite Machine Listening or Human Errors in Antescofo. In *Proceedings of the 2012 International Computer Music Conference*, Ljubljana, Slovenia, pp. 194-199.

Contemporary Music Review. (2009). Special Issue: Aesthetic Decisions in Computer-Aided Composition, **28**(2), 129-244.

Costantini, G., Rizzi, A. and Casali, D. (2003). Recognition of musical instruments by generalised min-max classifiers. In *IEEE 13th Workshop on Neural Networks for Signal Processing (NNSP)*, Toulouse, France, pp. 555-564.

Cross, J. (2003). Composing with Numbers: Sets, Rows and Magic Squares. In *Music and Mathematics: From Pythagoras to Fractals*. Fauvel, J., Flood, R. and Wilson, R. (eds). Oxford University Press, Oxford, pp. 131-146.

Cukier, K. (2010). Data, data everywhere. *The Economist, Special report: Managing information*. 25 February 2010.

Dannenbergh, R.B. and Raphael, C. (2006). Music Score Alignment and Computer Accompaniment. *Communications of the ACM*, **49**(8), 38-43.

Davies, S. (1994). *Musical Meaning and Expression*. Cornell University Press.

- Deng, J.D., Simmermacher, C. and Cranefield, S. (2008). A study on feature analysis for musical instrument classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **38**(2), 429-438.
- Dixon, S. (2007). Evaluation of the Audio Beat Tracking System BeatRoot. *Journal of New Music Research*, **36**(1), 39-50.
- Doquire, G. and Verleysen, M. (2011). Feature Selection for Multi-label Classification Problems. In *Advances in Computational Intelligence: Proceedings of the 11th International Work-Conference on Artificial Neural Networks Conference (IWANN'11)*, Torremolinos-Málaga, Spain, pp. 9-16.
- Dornbush, S., Fisher, K., McKay, K., Prikhodko, A. and Segall, Z. (2005). Xpod – A Human Activity and Emotion Aware Mobile Music Player. In *Proceedings of the 2nd International Conference on Mobile Technology, Applications and Systems*, Guangzhou, China.
- Dreyfus, G. and Guyon, I. (2006). Assessment methods. In *Feature Extraction: Foundations and Applications*. Guyon, I., Nikravesh, M., Gunn, S. And Zadeh, L.A. (eds). Springer Berlin Heidelberg, pp. 65-88.
- Duan, Z., Han, J. and Pardo, B. (2009). Harmonically informed multi-pitch tracking. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, pp. 333-338.
- Dubnov, S. and Rodet, X. (1998). Timbre recognition with combined stationary and temporal features. In *Proceedings of the International Computer Music Conference (ICMC 98)*, Ann Arbor, Michigan, USA.
- Duch, W. (2006). Filter methods. In *Feature Extraction: Foundations and Applications*. Guyon, I., Nikravesh, M., Gunn, S. And Zadeh, L.A. (eds). Springer Berlin Heidelberg, pp. 89-117.

Eerola, T. and Ferrer, R. (2008). Instrument library (MUMS) revised. *Music Perception*, **25**(3), 253-255.

Eggink, J. and Brown, G. (2003). A missing feature approach to instrument identification in polyphonic music. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*.

Eggink, J. and Brown, G.J. (2004). Instrument recognition in accompanied sonatas and concertos. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Montreal, Quebec, Canada, pp. IV-217-IV-220.

Eichner, M., Wolff, M. and Hoffmann, R. (2006). Instrument classification using hidden Markov models. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, Victoria, BC, Canada, pp. 349-350.

Ellis, D.P.W. and Poliner, G.E. (2007). Identifying 'Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, **4**, Honolulu, Hawaii, USA, pp. 1429-1432.

Eronen, A. (2001). Comparison of features for musical instrument recognition. In *Proceedings of the IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, New Platz, New York, USA, pp. 19-22.

Eronen, A. and Klapuri, A. (2000). Musical instrument recognition using cepstral coefficients and temporal features. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Istanbul, Turkey, pp. II-753-II-756.

Essid, S., Richard, G. and David, B. (2004). Efficient musical instrument recognition on solo performance music using basic features. In *Proceedings of the AES International Conference*, London, UK.

Essid, S., Richard, G. and David, B. (2005). Instrument recognition in polyphonic music. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '05)*, Philadelphia, Pennsylvania, USA, pp. III-245-III-248.

Essid, S., Richard, G. and David, B. (2006a). Hierarchical classification of musical instruments on solo recordings. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Toulouse, France, pp. V-817-V-820.

Essid, S., Richard, G. and David, B. (2006b). Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions on Audio, Speech and Language Processing*, **14**, 68-80.

Essid, S., Richard, G. and David, B. (2006c). Musical instrument recognition by pairwise classification strategies. *IEEE Transactions on Audio, Speech and Language Processing*, **14**(4), 1401-1412.

Fabiani, M. (2010). Frequency, phase and amplitude estimation of overlapping partials in monaural music signals. In *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx)*, Graz, Austria.

Fan, J. and Fan, Y. (2008). High-dimensional classification using features annealed independence rules. *The Annals of Statistics*, **36**(6), 2605-2637.

Fan, J. and Lv, J. (2010). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, **20**, 101-148.

Farnsworth, P.R. (1954). A Study of the Hevner Adjective List. *Journal of Aesthetics and Art Criticism*, **13**(1), 97-103.

Fauvel, J., Flood, R. and Wilson, R. (eds). (2003). *Music and Mathematics: From Pythagoras to Fractals*. Oxford University Press, Oxford.

Feigelson, E.D. and Babu, G.J. (eds). (2012). *Statistical Challenges in Modern Astronomy V*. Springer, New York.

Fletcher, N.H. and Rossing, T.D. (1998). *The Physics of Musical Instruments*. Second Edition. Springer, New York.

Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1), 119-139.

Fu, Z., Lu, G., Ting, K. and Zhang, D. (2011). A Survey of Audio-Based Music Classification and Annotation. *IEEE Transactions on Multimedia*, **13**(2), 303-319.

Fuhrmann, F., Haro, M. and Herrera, P. (2009). Scalability, generality and temporal aspects in automatic recognition of predominant musical instruments in polyphonic music. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, pp. 321-326.

Fuhrmann, F. and Herrera, P. (2010). Polyphonic instrument recognition for exploring semantic similarities in music. In *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria.

Fuhrmann, F. and Herrera, P. (2011). Quantifying the relevance of locally extracted information for musical instrument recognition from entire pieces of music. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, Miami, Florida, USA, pp. 239-244.

Fujinaga, I. and MacMillan, K. (2000). Realtime recognition of orchestral instruments. In *Proceedings of the International Computer Music Conference*, Berlin, Germany.

Fürnkranz, J. and Hüllermeier, E. (2010). Preference Learning and Ranking by Pairwise Comparison. In *Preference Learning*, Fürnkranz, J. And Hüllermeier, E. (eds). Springer-Verlag Berlin Heidelberg, pp. 65-82.

- Fürnkranz, J., Hüllermeier, E., Mencia, E.L. and Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, **73**(2), 133-153.
- Gabrielsson, A. and Juslin, P.N. (2003). Emotional Expression in Music. In *Handbook of Affective Sciences*, Davidson, R.J., Goldsmith, H.H. and Scherer, K.R. (eds). Oxford University Press, New York, pp. 503-534.
- Gabrielsson, A. and Lindström, E. (2001). The Influence of Musical Structure on Emotional Expression. In *Music and Emotion: Theory and Research*. Juslin, P.N. and Sloboda, J.A. (eds). Oxford University Press, New York, pp. 223-248.
- Gantz, J. and Reinsel, D. (2012). *The digital universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East*. IDC iView, available online at <http://idcdocserv.com/1414> (accessed 23 July 2013).
- Gardner, M. (1978). Mathematical Games: White and Brown Music, Fractal Curves and One-over-f Fluctuations. *Scientific American*, **238**(4), 16-32.
- Geurts, P., Ernst, D. and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, **63**(1), 3-42.
- Gheyas, I.A. and Smith, L.S. (2010). Feature subset selection in large dimensionality domains. *Pattern Recognition*, **43**(1), 5-13.
- Grosche, P., Müller, M. and Serrá, J. (2012). Audio Content-Based Music Retrieval. In *Multimodal Music Processing*. Müller, M., Goto, M. and Schedl, M. (eds). Dagstuhl Follow-Ups, Dagstuhl Publishing, Germany, pp. 157-174.
- Gu, Q., Li, Z. and Han, J. (2011). Correlated Multi-Label Feature Selection. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, Glasgow, Scotland, UK, pp. 1087-1096.
- Gunasekaran, S. and Revathy, K. (2008). Fractal dimension analysis of audio signals for Indian musical instrument recognition. In *Proceedings of the International*

Conference on Audio, Language and Image Processing (ICALIP 2008), Shanghai, China, pp. 257-261.

Gunn, S.R. (1998). *Support Vector Machines for Classification and Regression*. Technical Report, Faculty of Engineering and Applied Science, Department of Electronics and Computer Science, University of Southampton.

Guyon, I. and Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, **3**, 1157-1182.

Hamel, P., Wood, S. and Eck, D. (2009). Automatic identification of instrument classes in polyphonic and poly-instrument audio. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, pp. 399-404.

Hastie, T. and Tibshirani, R. (1998). Classification by pairwise coupling. *The Annals of Statistics*, **26**(2), 451-471.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second edition, Springer.

Heittola, T., Klapuri, A. and Virtanen, T. (2009). Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, pp. 327-332.

Helmholtz, H. (1954). *On the Sensations of Tone as a Physiological Basis for the Theory of Music*. Second Edition. Dover Publications, New York.

Herrera, P., Klapuri, A. and Davy, M. (2006). Automatic Classification of Pitched Musical Instrument Sounds. In *Signal Processing Methods for Music Transcription*. Klapuri, A. and Davy, M. (eds). Springer Science & Business Media LLC, pp. 163-200.

Herrera, P., Peeters, G. and Dubnov, S. (2003). Automatic classification of musical instrument sounds. *Journal of New Music Research*, **32**(1), 3-21.

Herrera, P., Xavier, A., Batlle, E. and Serra, X. (2000). Towards Instrument Segmentation for Music Content Description: A Critical Review of Instrument Classification Techniques. In *Proceedings of the 1st International Symposium on Music Information Retrieval (ISMIR 2000)*, Plymouth, Massachusetts, USA.

Herrera, P., Yeterian, A. and Gouyon, F. (2002). Automatic classification of drum sounds – a comparison of feature selection methods and classification techniques. In *Music and Artificial Intelligence: Proceedings of the Second International Conference (ICMAI)*, Edinburgh, Scotland. Anagnostopoulou, C., Ferrand, M. and Smaill, A. (eds). Springer-Verlag Berlin, pp. 69-80.

Hevner, K. (1935). Expression in Music: A Discussion of Experimental Studies and Theories. *Psychological Review*, **42**(2), 186-204.

Homburg, H., Mierswa, I., Möller, B., Morik, K. and Wurst, M. (2005). A Benchmark Dataset for Audio Classification and Clustering. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, London, UK, pp. 528-531.

Hüllermeier, E., Fürnkranz, J., Cheng, W. and Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence*, **172**(16-17), 1897-1916.

Ihara, M., Maeda, S. and Ishii, S. (2007). Instrument identification in monophonic music using spectral information. In *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology*, Cairo, Egypt, pp. 595-599.

Irizarry, R.A. (1998). *Statistics and Music: Fitting a local harmonic model to musical sound signals*. Unpublished PhD Thesis, University of California, Berkeley, USA.

- Isacoff, S. (2002). *Temperament: How Music Became the Battleground for the Great Minds of Western Civilization*. Faber & Faber, London.
- Itoyama, K., Goto, M., Komatani, K., Ogata, T. and Okuno, H. (2008). Instrument equalizer for query-by-example retrieval – improving sound source separation based on integrated harmonic and inharmonic models. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, Philadelphia, USA, pp. 133-138.
- James, J. (1993). *The Music of the Spheres: Music, Science and the Natural Order of the Universe*. Abacus, England.
- Jensen, J.H. (2009). *Feature Extraction for Music Information Retrieval*. Unpublished PhD Thesis, Department of Electronic Systems, Aalborg University, Denmark.
- Jiang, W., Cohen, A. and Ras, Z.W. (2009a). Polyphonic music information retrieval based on multi-label cascade classification system. In *Advances in Information and Intelligent Systems*. Studies in Computational Intelligence, **251**. Ras, Z.W. and Ribarsky, W. (eds). Springer-Verlag Berlin Heidelberg, pp. 117-137.
- Jiang, W., Wieczorkowska, A. and Ras, Z. (2009b). Music Instrument Estimation in Polyphonic Sound based on Short-term Spectrum Match. In *Foundations of Computational Intelligence Volume 2: Approximate Reasoning*. Hassanien, A., Abraham, A. and Herrera, F. (eds). Springer-Verlag Berlin Heidelberg, pp. 259-274.
- Jiang, W., Zhang, X., Cohen, A. and Ras, Z. (2010). Multiple classifiers for different features in timbre estimation. In *Advances in Intelligent Information Systems*. Ras, Z. And Tsay, L. (eds). Springer-Verlag Berlin Heidelberg, pp. 335-356.
- Jincahitra, P. (2004). Polyphonic instrument identification using independent subspace analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME) Volume 2*, Taipei, Taiwan, pp. 1211-1214.

Joder, C., Essid, S. and Richard, G. (2009). Temporal integration for audio classification with application to musical instrument classification. *IEEE Transactions on Audio, Speech and Language Processing*, **17**(1), 174-186.

Kaminskas, M. and Ricci, F. (2012). Contextual Music Information Retrieval and Recommendation: State of the Art and Challenges. *Computer Science Review*, **6**(2-3), 89-119.

Kaminskyj, I. and Czaszejko, T. (2005). Automatic recognition of isolated monophonic musical instrument sounds using k-NNC. *Journal of Intelligent Information Systems*, **24**(2-3), 199-221.

Karatzoglou, A., Smola, A., Hornik, K. And Zeileis, A. (2004). kernlab – An S4 package for kernel methods in R. *Journal of Statistical Software*, **11**(9), 1-20.

Kashino, K. and Murase, H. (1999). A sound source identification system for ensemble music based on template adaptation and music stream extraction. *Speech Communication*, **27**(3-4), 337-349.

Kellner, H.A. (1981). The Mathematical Architecture of Bach's Goldberg Variations. *The English Harpsichord Magazine*, **2**(8), 183-189.

Kim, H., Moreau, N. and Sikora, T. (2004). Audio classification based on MPEG-7 spectral basis representations. *IEEE Transactions on Circuits and Systems for Video Technology*, **14**(5), 716-725.

Kim, H., Moreau, N. and Sikora, T. (2005). *MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval*. John Wiley & Sons, England.

Kimber, A.C. and Hansford, A.R. (1993). A Statistical Analysis of Batting in Cricket. *Journal of the Royal Statistical Society A*, **156**(3), 443-455.

- Kitahara, T., Goto, M., Komatani, K., Ogata, T. and Okuno, H.G. (2006). Instrogram: a new Musical Instrument Recognition Technique without using onset detection nor F0 estimation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Toulouse, France, pp. V-229-V-232.
- Kitahara, T., Goto, M., Komatani, K., Ogata, T. and Okuno, H. (2007a). Instrogram – Probabilistic representation of instrument existence for polyphonic music. *IPJS Transactions on Databases*, **48**(1), 214-226.
- Kitahara, T., Goto, M., Komatani, K., Ogata, T. and Okuno, H.G. (2007b). Instrument identification in polyphonic music – feature weighting to minimize influence of sound overlaps. *EURASIP Journal on Advances in Signal Processing*, vol. 2007, article ID 51979.
- Kitahara, T., Goto, M. and Okuno, H. (2005). Pitch-dependent identification of musical instrument sounds. *Applied Intelligence*, **23**(3), 267-275.
- Klapuri, A. (2001). Multipitch information and sound separation by the spectral smoothness principle. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '01)*, Salt Lake City, Utah, USA, pp. V-3381-V-3384.
- Klapuri, A.P. (2004). Automatic Music Transcription as We Know it Today. *Journal of New Music Research*, **33**(3), 269-282.
- Klapuri, A.P. (2010). Pattern Induction and Matching in Music Signals. In *Exploring Music Contents: Proceedings of the 7th International Symposium on Computer Music Modelling and Retrieval (CMMR'10)*, Springer-Verlag, Berlin, pp. 188-204.
- Kocev, D., Vens, C., Struyf, J. and Džeroski, S. (2007). Ensembles of multi-objective decision trees. In *Proceedings of the 18th European conference on Machine Learning*, pp. 254-269.

Kolozali, S., Barthelet, M., Fazekas, G. and Sandler, M. (2011). Knowledge representation issues in musical instrument ontology design. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, Miami, Florida, USA, pp. 465-470.

Kong, D., Ding, C., Huang, H. and Zhao, H. (2012). Multi-Label ReliefF and F-statistic Feature Selections for Image Annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, Rhode Island, USA, pp. 2352-2359.

Kostek, B. (2004). Musical instrument classification and duet analysis employing music information retrieval techniques. In *Proceedings of the IEEE*, **92**(4), 712-729.

Kostek, B. and Czyzewski, A. (2001). Representing musical instrument sounds for their automatic classification. *Journal of the Audio Engineering Society*, **49**(9), 768-785.

Kostek, B., Kupryjanow, A., Zwan, P., Jiang, W., Ras, Z.W., Wojnarski, M. and Swietlicka, J. (2011). Report of the ISMIS 2011 Contest: Music Information Retrieval. In *Foundations of Intelligent Systems: Proceedings of the 19th International Symposium (ISMIS'11)*, Warsaw, Poland, pp. 715-724.

Kotsifakos, A., Papapetrou, P., Hollmén, J., Gunopulos, D. and Athitsos, V. (2012). A Survey of Query-by-Humming Similarity Methods. In *Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA 2012)*, Heraklion, Crete, Greece.

Krishna, A.G. and Sreenivas, T.V. (2004). Music instrument recognition: from isolated notes to solo phrases. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Montreal, Quebec, Canada, pp. IV-265-IV-268.

Kubera, E., Wiczorkowska, A. and Ras, Z.W. (2013). Time Variability-Based Hierarchic Recognition of Multiple Musical Instruments in Recordings. In *Rough*

Sets and Intelligent Systems – Professor Zdzislaw Pawlak in Memoriam, 2. Skowron, A. and Suraj, Z. (eds). Springer Berlin Heidelberg, pp. 347-363.

Kubera, E., Wieczorkowska, A., Ras, Z. and Skrzypiec, M. (2010). Recognition of instrument timbres in real polytimbral audio recordings. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pp. 97-110.

Kursa, M.B., Jankowski, A. and Rudnicki, W.R. (2010a). Boruta – A System for Feature Selection. *Fundamenta Informaticae*, 101(4), 271-285.

Kursa, M.B., Kubera, E., Rudnicki, W.R. and Wieczorkowska, A. (2010b). Random musical bands playing in random forests. In *Rough Sets and Current Trends in Computing: Proceedings of the 7th International Conference (RSCTC 2010)*, Warsaw, Poland, pp. 580-589.

Kursa, M., Rudnicki, W., Wieczorkowska, A., Kubera, E. and Kubik-Komar, A. (2009). Musical Instruments in Random Forest. In *Proceedings of the 18th International Symposium on Foundations of Intelligent Systems*. Rauch, J., Ras, Z.W., Berka, P. and Elomaa, T. (eds). Springer, pp. 281-290.

Lamere, P. (2008). Social Tagging and Music Information Retrieval. *Journal of New Music Research*, 37(2), 101-114.

Lastra, G., Luaces, O., Quevedo, J.R. and Bahamonde, A. (2011). Graphical Feature Selection for Multilabel Classification Tasks. In *Proceedings of the 10th International Conference on Advances in Intelligent Data Analysis (IDA'11)*, Porto, Portugal, pp. 246-257.

Laurier, C. and Herrera, P. (2007). Audio Music Mood Classification using Support Vector Machine. *Music Information Retrieval Evaluation eXchange (MIREX 2007) extended abstract*.

Lee, J.H. and Downie, J.S. (2004). Survey of Music Information Needs, Uses and Seeking Behaviours: Preliminary Findings. In *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR 2004)*, Barcelona, Spain.

Lee, J. and Kim, D. (2013). Feature selection for multi-label classification using multivariate mutual information. *Pattern Recognition Letters*, **34**(3), 349-357.

Lee, H., Largman, Y., Pham, P. and Ng, A.Y. (2009). Unsupervised Feature Learning for Audio Classification using Convolutional Deep Belief Networks. In *Advances in Neural Information Processing Systems vol. 22*. Bengio, Y., Schuurmans, D., Lafferty, J., Williams C.K.I. and Culotta, A. (eds), pp. 1096-1104.

Lee, J., Lim, H. and Kim, D. (2012). Approximating mutual information for multi-label feature selection. *Electronics Letters*, **48**(15), 929-930.

Lerch, A. (2006). On the Requirement of Automatic Tuning Frequency Estimation. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, Victoria, BC, Canada, pp. 212-215.

Levitin, D. (2006). *This is Your Brain on Music: Understanding a Human Obsession*. Atlantic Books, London.

Levy, M. and Sandler, M. (2009). Music Information Retrieval Using Social Tags and Audio. *IEEE Transactions on Multimedia*, **11**(3), 383-395.

Lewis, R.A., Zhang, X. and Ras, Z.W. (2007). Knowledge discovery-based identification of musical pitches and instruments in polyphonic sounds. *Engineering Applications of Artificial Intelligence*, **20**(5), 637-645.

Li, T. and Ogihara, M. (2003). Detecting Emotion in Music. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR 2003)*, Baltimore, Maryland, USA.

Li, T., Ogihara, M. and Tzanetakis, G. (eds). (2011). *Music Data Mining*. CRC Press, Florida.

Li, L., Zhang, J. and Neal, R.M. (2008). A Method for Avoiding Bias from Feature Selection with Application to Naive Bayes Classification Models. *Bayesian Analysis*, 3(1), pp. 171-196.

Lidy, T., Silla, C., Cornelis, O., Gouyon, F., Rauber, A., Kaestner, C. and Koerich, A. (2010). On the suitability of state-of-the-art music information retrieval methods for analysing, categorising and accessing non-Western and ethnic music collections. *Signal Processing*, 90(4), 1032-1048.

Little, D. and Pardo, B. (2008). Learning musical instruments from mixtures of audio with weak labels. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, Philadelphia, USA, pp. 127-132.

Liu, H. and Motoda, H. (2008). *Computational Methods of Feature Selection*. Chapman & Hall/CRC.

Liu, H. and Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4), 491-502.

Livshin, A. and Rodet, X. (2003). The Importance of Cross Database Evaluation in Sound Classification. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR 2003)*, Baltimore, Maryland, USA.

Livshin, A. and Rodet, X. (2004). Musical instrument identification in continuous recordings. In *Proceedings of the 7th International Conference on Digital Audio Effects (DAFX)*, Naples, Italy, pp. 222-227.

Livshin, A. and Rodet, X. (2009). Purging Musical Instrument Sample Databases Using Automatic Musical Instrument Recognition Methods. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5), 1046-1051.

- Logan, B. (2000). Mel Frequency Cepstral Coefficients for Music Modelling. In *Proceedings of the 1st International Symposium on Music Information Retrieval (ISMIR 2000)*, Plymouth, Massachusetts, USA.
- Lohr, S. (2009). For Today's Graduate, Just One Word: Statistics. *The New York Times*. 6 August 2009.
- Loughran, R., Walker, J., O'Neill, M. and O'Farrell, M. (2008a). Musical instrument identification using principal component analysis and multi-layered perceptrons. In *Proceedings of the International Conference on Audio, Language and Image Processing (ICALIP 2008)*, Shanghai, China, pp. 643-648.
- Loughran, R., Walker, J., O'Neill, M. and O'Farrell, M. (2008b). The use of mel-frequency cepstral coefficients in musical instrument identification. In *Proceedings of the International Computer Music Conference (ICMC 2008)*, Belfast, Northern Ireland.
- Lu, L., Liu, D. and Zhang, H. (2006). Automatic Mood Detection and Tracking of Music Audio Signals. *IEEE Transactions on Audio, Speech and Language Processing*, **14**(1), 5-18.
- Luaces, O., Díez, J., Barranquero, J., Del Coz, J.J. and Bahamonde, A. (2012a). Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, **1**(4), 303-313.
- Luaces, O., Díez, J., Del Coz, J.J., Barranquero, J. and Bahamonde, A. (2012b). Synthetic Datasets for Sound Experimental Evaluation of Multilabel Classifiers. Unpublished paper. Available from http://www.aic.univovi.es/ml_generator (accessed 18 June 2013).
- Lukashevich, H., Abesser, J., Dittmar, C. and Grossmann, H. (2009). From Multi- Labelling to Multi-Domain Labelling: A Novel Two Dimensional Approach to Music Genre Classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, pp. 459-464.

Madjarov, G., Kocev, D., Gjorgjevikj, D. and Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, **45**(9), 3084-3104.

Mandelbrot, B.B. (1982). *The Fractal Geometry of Nature*. W.H. Freeman, New York.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C. and Byers, A.H. (2011). *Big data: The next frontier for innovation, competition, and productivity*. Report by McKinsey Global Institute. Available at http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation (accessed 23 July 2013).

Marillier, C.G. (1983). Computer Assisted Analysis of Tonal Structure in the Classical Symphony. *Haydn Yearbook*, **14**, 187-199.

Marques, J. and Moreno, P.J. (1999). A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines. *CRL Technical Report Series CRL/4*, Compaq Cambridge Research Laboratory.

Martin, K. and Kim, Y. (1998). Musical instrument identification – a pattern-recognition approach. Presented at the 136th Meeting of the Acoustical Society of America, Norfolk, Virginia.

Martins, L., Burred, J., Tzanetakis, G. and Lagrange, M. (2007). Polyphonic instrument recognition using spectral clustering. In *Proceedings of the 8th International Society for Music Information Retrieval Conference (ISMIR 2007)*, Vienna, Austria, pp. 213-218.

Mathieu, B., Essid, S., Fillon, T., Prado, J. and Richard, G. (2010). YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, Netherlands, pp. 441-446.

McKay, C. and Fujinaga, I. (2006). Musical Genre Classification: Is it Worth Pursuing and how can it be Improved? In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, Victoria, BC, Canada, pp. 101-106.

McKay, J., Gainza, M. and Barry, D. (2009). Evaluating ground truth for ADResS as a preprocess for automatic musical instrument identification. In *Proceedings of the 126th AES Convention*, Munich, Germany.

McKinney, M.F. and Breebaart, J. (2003). Features for Audio and Music Classification. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR 2003)*, Baltimore, Maryland, USA.

Mencia, E.L., Park, S. and Fürnkranz, J. (2010). Efficient voting prediction for pairwise multilabel classification. *Neurocomputing*, **73**, 1164-1176.

Moelants, D., Cornelis, O., Leman, M., Gansemans, J., De Caluwe, R., De Tré, G., Matthé, T. and Hallez, A. (2006). Problems and Opportunities of Applying Data- and Audio-Mining Techniques to Ethnic Music. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, Victoria, BC, Canada, pp.334-336.

Morvidone, M., Sturm, B. and Daudet, L. (2010). Incorporating scale information with cepstral features: Experiments on musical instrument recognition. *Pattern Recognition Letters*, **31**(12), 1489-1497.

Müller, M. (2011). New Developments in Music Information Retrieval. In *Proceedings of the AES 42nd International Conference: Semantic Audio*, Ilmenau, Germany, pp. 11-20.

Müller, M., Ellis, D.P.W., Klapuri, A. and Richard, G. (2011). Signal Processing for Music Analysis. *IEEE Journal of Selected Topics in Signal Processing*, **5**(6), 1088-1110.

- Müller, M., Konz, V., Scharfstein, A. Ewert, S. and Clausen, M. (2009). Towards Automated Extraction of Tempo Parameters from Expressive Music Recordings. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, pp. 69-74.
- Müller, M., Kurth, F. and Clausen, M. (2005). Chroma-based Statistical Audio Features for Audio Matching. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 275-278.
- Nettheim, N. (1992). On the Spectral Analysis of Melody. *Interface: the Journal of New Music Research*, **21**(2), 135-148.
- Nettheim, N. (1997). A Bibliography of Statistical Applications in Musicology. *Musicology Australia*, **20**(1), 94-106.
- Nielsen, A.B., Sigurdsson, S., Hansen, L.K. and Arenas-Garcia, J. (2007). On the relevance of spectral features for instrument classification. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing*. Honolulu, Hawaii, USA, pp. II-485-II-488.
- Niwa, K., Tehrani, M.P., Nishino, T. and Takeda, K. (2006). Signal Separation of Instrumental Performance and Development of Selectable Listening Position Audio System with HRTFS. In *4th Symposium on Intelligent Media Integration for Social Information Infrastructure*, 49-52.
- Okamura, M., Takehara, M., Tamura, S. and Hayamizu, S. (2012). Toward polyphonic musical instrument identification using example-based sparse representation. In *Proceedings of the 2012 Asia-Pacific Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Hollywood, California, USA.
- Olson, H.F. (1967). *Music, Physics and Engineering*. Second Edition. Dover Publications.

- Oppenheim, A.V. (1970). Speech Spectrograms Using the Fast Fourier Transform. *IEEE Spectrum*, 7(8), 57-62.
- Pachet, F. and Cazaly, D. (2000). A Taxonomy of Musical Genres. In *Computer-Assisted Information Retrieval: Proceedings of the 6th International RIAO Conference*, Paris, France, pp. 1238-1245.
- Panagakis, Y., Kotropoulos, C. and Arce, G.R. (2009). Music Genre Classification Using Locality Preserving Non-Negative Tensor Factorization and Sparse Representations. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, pp. 249-254.
- Park, S. and Fürnkranz, J. (2007). Efficient pairwise classification. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, Warsaw, Poland, pp. 658-665.
- Paulus, J. and Klapuri, A. (2009). Music Structure Analysis Using a Probabilistic Fitness Measure and a Greedy Search Algorithm. *IEEE Transactions on Audio, Speech and Language Processing*, 17(6), 1159-1170.
- Paulus, J., Müller, M. and Klapuri, A. (2010). Audio-Based Music Structure Analysis. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, Netherlands, pp. 625-636.
- Pauwels, J., Martens, J. and Leman, M. (2011). Improving the Key Extraction Performance of a Simultaneous Local Key and Chord Estimation System. In *2011 IEEE International Conference on Multimedia and Expo (ICME)*.
- Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. Wiley-IEEE Press.
- Peeters, G. (2008). A Generic Training and Classification System for MIREX08 Classification Tasks: Audio Music Mood, Audio Genre, Audio Artist and Audio Tag.

In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, Philadelphia, USA.

Pei, S. and Hsu, N. (2009). Instrumentation analysis and identification of polyphonic music using beat-synchronous feature integration and fuzzy clustering. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, pp. 169-172.

Platt, J. (2000). Probabilities for SV Machines. In *Advances in Large Margin Classifiers*, Smola, A.J., Bartlett, P., Schölkopf, B. And Schuurmans, D. (eds). MIT Press, pp. 61-74.

Poliner, G.E., Ellis, D.P.W., Ehmann, A.F., Gómez, E., Streich, S. and Ong, B. (2007). Melody Transcription from Music Audio: Approaches and Evaluation. *IEEE Transactions on Audio, Speech and Language Processing*, **15**(4), 1247-1256.

Powell, J. (2010). *How Music Works: A Listener's Guide to the Science and Psychology of Beautiful Sounds*. Penguin, London.

Qi, G.J., Hua, X.S., Rui, Y., Tan, J., Mei, T. and Zhang, H.J. (2007). Correlative multi-label video annotation. In *Proceedings of the 15th international conference on Multimedia*, New York, USA, pp. 17-26.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo.

R Core Team (2013). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. <http://www.R-project.org/>.

Rabiner, L. and Juang, B. (1993). *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series.

Rao, C.R. (1997). *Statistics and Truth: Putting Chance to Work*. Second Edition, World Scientific Publishing Co, Singapore.

- Read, J. (2008). A pruned problem transformation method for multi-label classification. In *Proceedings of the New Zealand Computer Science Research Student Conference*, pp. 143-150.
- Read, J., Bifet, A., Holmes, G. and Pfahringer, B. (2012). Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, **88**(1-2), 243-272.
- Read, J., Pfahringer, B. and Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pp. 995-1000.
- Read, J., Pfahringer, B. and Holmes, G. (2009a). Generating Synthetic Multi-label Data Streams. In *Proceedings of the ECML/PKDD 2009 Workshop on Learning from Multi-label Data (MLD'09)*, Bled, Slovenia.
- Read, J., Pfahringer, B., Holmes, G. and Frank, E. (2009b). Classifier chains for multi-label classification. In *Proceedings of the 20th European Conference on Machine Learning*, pp. 254-269.
- Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marcal, A.R.S., Guedes, C. and Cardoso, J.S. (2012). Optical Music Recognition: State-of-the-Art and Open Issues. *International Journal of Multimedia Information Retrieval*, **1**(3), 173-190.
- Refaeilzadeh, P., Tang, L. and Liu, H. (2007). On comparison of feature selection algorithms. In *Proceedings of the AAAI 2007 Workshop on Evaluation Methods for Machine Learning II*, Vancouver, Canada.
- Richard, G., Leveau, P., Daudet, L., Essid, S. and David, B. (2007). Towards polyphonic musical instruments recognition. In *Proceedings of the International Congress on Acoustics*, Madrid, Spain.
- Roads, C. (1996). *The Computer Music Tutorial*. MIT Press.

- Roaf, D. and White, A. (2003). Ringing the Changes: Bells and Mathematics. In *Music and Mathematics: From Pythagoras to Fractals*. Fauvel, J., Flood, R. and Wilson, R. (eds). Oxford University Press, Oxford, pp. 113-130.
- Röver, C., Klefenz, F. and Weihs, C. (2005). Identification of musical instruments by means of the Hough-transformation. In *Classification – the Ubiquitous Challenge: Proceedings of the 28th Annual Conference of the Gesellschaft für Klassifikation*, Dortmund, Germany, pp. 608-615.
- Russell, J.A. (1980). A Circumplex Model of Affect. *Journal of Personality and Social Psychology*, **39**(6), 1161-1178.
- Ryynänen, M. and Klapuri, A. (2007). Automatic Bass Line Transcription from Streaming Polyphonic Audio. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, **4**, 1437-1440.
- Sachs, C. (1940). *The History of Musical Instruments*. W.W. Norton, New York.
- Sánchez-Marroño, N., Alonso-Betanzos, A. and Tombilla-Sanromán, M. (2007). Filter Methods for Feature Selection – A Comparative Study. In *Proceedings of the 8th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, Birmingham, UK, pp. 178-187.
- Sanden, C. and Zhang, J.Z. (2011). Enhancing Multi-Label Music Genre Classification Through Ensemble Techniques. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Beijing, China, pp. 705-714.
- Simmermacher, C., Deng, D. and Cranefield, S. (2006). Feature analysis and classification of classical musical instruments: An empirical study. Information Science Discussion Papers Series No. 2006/10. University of Otago.

Singhi, S. and Liu, H. (2006). Feature subset selection bias for classification learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, Pittsburgh, Pennsylvania, USA, pp. 849-856.

Sloboda, J.A. and Juslin, P.N. (2001). Psychological Perspectives on Music and Emotion. In *Music and Emotion: Theory and Research*, Juslin, P.N. and Sloboda, J.A. (eds), Oxford University Press, New York, pp. 71-104.

Smaragdis, P. and Brown, J.C. (2003). Non-negative Matrix Factorization for Polyphonic Music Transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, USA, pp. 177-180.

Somerville, P. and Uitdenbogerd, A.L. (2008). Multitimbral musical instrument classification. In *Proceedings of the International Symposium on Computer Science and its Applications (CSA '08)*, Hobart, Australia, pp. 269-274.

Spolaôr, N., Monard, M.C. and Lee, H.D. (2012). A systematic review to identify feature selection publications in multi-labeled data. ICMC Technical Report No. 374. University of São Paulo.

Spolaôr, N., Cherman, E.A., Monard, M.C. and Lee, H.D. (2013). A Comparison of Multi-label Feature Selection Methods using the Problem Transformation Approach. *Electronic Notes in Theoretical Computer Science*, **292**, 135-151.

Spyromitros-Xioufis, E., Tsoumakas, G. and Vlahavas, I. (2008). An empirical study of lazy multilabel classification algorithms. In *Proceedings of the 5th Hellenic conference on Artificial Intelligence: Theories, Models and Applications*, pp. 401-406.

Spyromitros-Xioufis, E., Tsoumakas, G. and Vlahavas, I. (2011). Multi-label learning approaches for music instrument recognition. In *Foundations of Intelligent Systems: Proceedings of the 19th International Symposium (ISMIS'11)*, Warsaw, Poland, pp. 734-743.

- Srinivasan, A., Sullivan, D. and Fujinaga, I. (2002). Recognition of isolated instrument tones by conservatory students. In *Proceedings of the International Conference on Music Perception and Cognition*, Sidney, Australia, pp. 17-21.
- Stoppiglia, H., Dreyfus, G., Dubois, R. and Oussar, Y. (2003). Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, **3**, 1399-1414.
- Sturm, B.L. (2012). A Survey of Evaluation in Music Genre Recognition. Paper read at the 10th International Workshop on Adaptive Multimedia Retrieval, Copenhagen, Denmark.
- Sturm, B., Morvidone, M. and Daudet, L. (2010). Musical instrument identification using multiscale mel-frequency cepstral coefficients. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, pp. 477-481.
- Taweewat, P. and Wutiwiwatchai, C. (2013). Musical pitch estimation using a supervised single hidden layer feed-forward neural network. *Expert Systems with Applications*, **40**, 575-589.
- Taylor, C. (2003). The Science of Musical Sound. In *Music and Mathematics: From Pythagoras to Fractals*. Fauvel, J., Flood, R. and Wilson, R. (eds). Oxford University Press, Oxford, pp. 47-60.
- Taylor, E. (1991). *The AB Guide to Music Theory, Part II*. The Associated Board of the Royal Schools of Music.
- Tellegen, A., Watson, D. and Clark, L.A. (1999). On the Dimensional and Hierarchical Structure of Affect. *Psychological Science*, **10**(4), 297-303.
- Temperley, D. (2007). *Music and Probability*. MIT Press, Cambridge.
- Thayer, R.E. (1989). *The Biopsychology of Mood and Arousal*. Oxford University Press, New York.

- Tjoa, S. and Liu, K. (2010). Musical instrument recognition using biologically inspired filtering of temporal dictionary atoms. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, Netherlands, pp. 435-440.
- Trohidis, K., Tsoumakas, G., Kalliris, G. and Vlahavas, I. (2008). Multi-Label Classification of Music into Emotions. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, Philadelphia, USA, pp. 325-330.
- Trohidis, K., Tsoumakas, G., Kalliris, G. and Vlahavas, I. (2011). Multi-label classification of music by emotion. *EURASIP Journal on Audio, Speech, and Music Processing*, 2011:4.
- Tsoumakas, G., Dimou, A., Spyromitros, E., Mezaris, V., Kompatsiaris, I. and Vlahavas, I. (2009). Correlation-Based Pruning of Stacked Binary Relevance Models for Multi-Label Learning. In *Proceedings of the ECML/PKDD 2009 Workshop on Learning from Multi-Label Data (MLD'09)*, Bled, Slovenia, pp. 101-116
- Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: an overview. *International Journal of Data Warehouse and Mining*, 3(3), 1-13.
- Tsoumakas, G., Katakis, I. and Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of the ECML/PKDD Workshop on Mining Multidimensional Data*, pp. 30-44.
- Tsoumakas, G., Katakis, I. and Vlahavas, I. (2010). Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, Springer Berlin/Heidelberg, pp. 667-685.
- Tsoumakas, G. and Vlahavas, I. (2007). Random k-labelsets: an ensemble method for multilabel classification. In *Proceedings of the 18th European conference on Machine Learning*, pp. 406-417.

Turnbull, D., Barrington, L., Torres, D. and Lanckriet, G. (2008). Semantic Annotation and Retrieval of Music and Sound Effects. *IEEE Transactions on Audio, Speech and Language Processing*, **16**(2), 467-476.

Tusher, V., Tibshirani, R. and Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation response. In *Proceedings of the National Academy of Sciences of the United States of America*, **98**(9), 5116-5121.

Tuv, E., Borisov, A. and Torkkola, K. (2008). Ensemble-based variable selection using independent probes. In *Computational Methods of Feature Selection*. Liu, H. And Motoda, H. (eds). Chapman & Hall/CRC.

Tzanetakis, G. (2002). *Manipulation, Analysis and Retrieval Systems for Audio Signals*. Unpublished PhD Thesis. Department of Computer Science, Princeton University.

Tzanetakis, G. (2011). Audio Feature Extraction. In *Music Data Mining*, Li, T., Ogihara, M. and Tzanetakis, G. (eds). CRC Press, Florida, pp. 43-74.

Tzanetakis, G. and Cook, P. (2002). Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, **10**(5), 293-302.

Tzanetakis, G., Ermolinskyi, E. and Cook, P. (2002). Pitch Histograms in Audio and Symbolic Music Information Retrieval. In *Proceedings of the 3rd International Society for Music Information Retrieval Conference (ISMIR 2002)*, Paris, France, pp. 31-38.

Uchida, Y. and Wada, S. (2011). Melody and Bass Line Estimation Method Using Audio Feature Database. In *Proceedings of the IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Xi'an, China.

Verleysen, M. and Francois, D. (2005). The Curse of Dimensionality in Data Mining and Time Series Prediction. In *Proceedings of the 8th International Work-*

Conference on Artificial Neural Networks: Computational Intelligence and Bioinspired Systems (IWANN'05), Barcelona, Spain, pp. 758-770.

Vincent, E. and Rodet, X. (2004). Instrument identification in solo and ensemble music using independent subspace analysis. In *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR 2004)*, Barcelona, Spain, pp. 576-581.

Virtanen, T. (2007). Monaural sound source separation by non-negative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech and Language Processing*, **15**(3), 1066-1074.

Von Hornbostel, E.M. and Sachs, C. (1961). Classification of Musical Instruments: Translated from the Original German by Anthony Baines and Klaus P. Wachsmann. *The Galpin Society Journal*, **14**, 3-29.

Voss, R.F. and Clarke, J. (1978). "1/f Noise" in Music: Music from 1/f Noise. *Journal of the Acoustical Society of America*, **63**, 258-263.

Walter, C. (2005). Kryder's Law. *Scientific American*, **293**(2), 32-33.

Wang, A. (2003). An Industrial-Strength Audio Search Algorithm. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR 2003)*, Baltimore, Maryland, USA, pp. 7-13.

Wang, B. and Plumbley, M.D. (2006). Investigating single-channel audio source separation methods based on non-negative matrix factorization. In *Proceedings of the 7th International Conference on Computer Music Modeling and Retrieval: Exploring Music Contents (CMMR'10)*, Malaga, Spain, pp. 84-101.

Wang, F., Wang, X., Shao, B., Li, T. and Ogihara, M. (2009). Tag Integrated Multi-Label Music Style Classification with Hypergraph. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, pp. 363-368.

Weihls, C., Ligges, U., Mörchen, F. and Müllensiefen, D. (2007). Classification in Music Research. *Advances in Data Analysis and Classification*, 1(3), 255-291.

Widmer, G., Dixon, S., Knees, P., Pampalk, E. and Pohle, T. (2008). From Sound to 'Sense' via Feature Extraction and Machine Learning: Deriving High-Level Descriptors for Characterising Music. In *Sound to Sense – Sense to Sound: A state of the art in Sound and Music Computing*. Polotti, P. and Rocchesso, D. (eds). Logos Verlag, Berlin, pp. 161-194.

Wieczorkowska, A., Kolczynska, E. and Ras, Z.W. (2008). Training of classifiers for the recognition of musical instrument dominating in the same-pitch mix. In *New Challenges in Applied Intelligence Technologies*, Nguyen, N.T. and Katarzyniak, R. (eds). Springer, pp. 213-222.

Wieczorkowska, A. and Kubera, E. (2010). Identification of a dominating instrument in polytimbral same-pitch mixes using SVM classifiers with nonlinear kernels. *Journal of Intelligent Information Systems*, 34(3), 275-303.

Wieczorkowska, A., Kubera, E. and Kubik-Komar, A. (2011). Analysis of Recognition of a Musical Instrument in Sound Mixes Using Support Vector Machines. *Fundamenta Informaticae*, 107(1), 85-104.

Wieczorkowska, A., Synak, P. and Ras, Z.W. (2006). Multi-Label Classification of Emotions in Music. In *Intelligent Information Processing and Web Mining: Proceedings of the International IIS:IIPWM'06 Conference*, Ustron, Poland, pp. 307-315.

Wiering, F. (2007). Can Humans Benefit from Music Information Retrieval? In *Adaptive Multimedia Retrieval: User, Context and Feedback: Proceedings of the 4th International AMR Workshop*, Springer-Verlag Berlin, Heidelberg, pp. 82-94.

Wu, J., Vincent, E., Raczynski, S., Nishimoto, T., Ono, N. and Sagayama, S. (2011). Polyphonic pitch estimation and instrument identification by joint modelling of

sustained and attack sounds. *IEEE Journal of Selected Topics in Signal Processing*, 5(6), 1124-1132.

Wu, J. and Sagayama, S. (2011). Musical instrument identification based on new boosting algorithm with probabilistic decisions. In *Proceedings of the 8th International Symposium CMMR 2011 and the 20th International Symposium FRSM 2011: Speech, Sound and Music Processing: embracing research in India*. Springer-Verlag Berlin Heidelberg, pp. 66-78.

Yang, Y. and Chen, H. (2012). Machine Recognition of Music Emotion: A Review. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3).

Yang, S., Kim, S.K. and Ro, Y.M. (2007). Semantic home photo categorization. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(3), 324-335.

Yang, Y., Lin, Y., Cheng, H., Liao, I., Ho, Y. and Chen, H. (2008). Toward Multi-Modal Music Emotion Classification. In *Proceedings of the 9th Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing*, Tainan, Taiwan, pp. 70-79.

Yang, Y., Liu, Ch. and Chen, H. (2006). Music Emotion Classification: A Fuzzy Approach. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*, Santa Barbara, California, USA, pp. 81-84.

Yang, B., Sun, J., Wang, T. and Chen, Z. (2009). Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, pp. 917-926.

Yoshii, K., Goto, M. and Okuno, H.G. (2007). Drum-sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression. *IEEE Transactions on Audio, Speech and Language Processing*, 15(1), 333-345.

- Yu, L. and Liu, H. (2004). Efficient Feature Selection via Analysis of Relevance and Redundancy. *Journal of Machine Learning Research*, **5**, 1205-1224.
- Zhang, Y., Burer, S. and Street, W.N. (2006). Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, **7**, 1315-1338.
- Zhang, M., Peña, J.M. and Robles, V. (2009). Feature Selection for Multi-Label Naive Bayes Classification. *Information Sciences*, **179**(19), 3218-3229.
- Zhang, X. and Ras, Z. (2007a). Analysis of Sound Features for Music Timbre Recognition. *International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, Seoul, Korea, pp. 3-8.
- Zhang, X. and Ras, Z. (2007b). Sound isolation by harmonic peak partition for music instrument recognition. *Fundamentae Informaticae*, **78**(4), 613-628.
- Zhang, X., Ras, Z.W. and Dardzinska, A. (2007). Discriminant Feature Analysis for Music Timbre Recognition and Automatic Indexing. In *Mining Complex Data: Proceedings of the 3rd ECML/PKDD International Workshop (MCD 2007)*, Warsaw, Poland, pp. 104-115.
- Zhang, M.L. and Zhou, Z.H. (2007). ML-kNN: a lazy learning approach to multi-label learning. *Pattern Recognition*, **40**(7), 2038-2048.
- Zhang, M. and Zhou, Z. (2013). A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, IEEE Computer Society Digital Library. IEEE Computer Society.
- Zhu, Y., Kankanhalli, M.S. and Gao, S. (2005). Music Key Detection for Musical Audio. In *Proceedings of the 11th International Conference on Multimedia Modelling (MMM'05)*, Melbourne, Australia, pp. 30-37.

Zlatintsi, A. and Maragos, P. (2011). Musical instruments signal analysis and recognition using fractal features. In *Proceedings of the 19th European Signal Processing Conference (EUSIPCO)*, Barcelona, Spain.

APPENDIX A

R programs

This appendix contains the R programs used in the empirical studies. All of these programs were written in R. Appendices A.1 – A.3 contain the programs used in the simulation study (Chapter 7) while appendices A.5 – A.7 contain those used in the instrument recognition study (Chapter 8). Appendix A.4 contains the program used to calculate multi-label evaluation measures and was used in both empirical studies (Chapter 7 and Chapter 8).

A.1 Simulation study – main program

This is the main program used in the empirical study described in Chapter 7. It contains calls to sub-programs for generating synthetic multi-label data (Appendix A.2), feature selection (Appendix A.3) and the calculation of multi-label evaluation measures (Appendix A.4).

```
#####  
# This function is called with 10 arguments: #  
# p: The number of features generated #  
# prel: The number of relevant features #  
# K: The number of labels #  
# Ntrain: the number of training data samples #  
# Ntest: the number of test data samples #  
# alfa: Parameter used in the probe variable selection function #  
# Bselect: Parameter used in the probe variable selection function #  
# densities: label densities #  
# sigmamat: correlation matrix for correlations between labels #  
# sigmax: correlation matrix for correlations between features #  
#####  
  
function(p,prel,K,Ntrain,Ntest,alfa,Bselect,densities,  
        sigmamat,sigmax) {
```

```

# Set number of labels to be predicted
numlabels = floor(K*mean(1-densities))

# Number of Monte Carlo simulations
NMC=100

# Define matrices to be used later in the program
simfullmeasure=matrix(0,NMC,6)
simselmeasure=matrix(0,NMC,6)
simselvars=matrix(0,NMC,p)

# Generate synthetic training and test data
for (mmm in 1:NMC) {

data_output = sintdata(K,Ntrain,densities,sigmamat,p,prel,sigmax)
Xtrain = data_output[[1]]
Ytrain = data_output[[2]]

data_output = sintdata(K,Ntest,densities,sigmamat,p,prel,sigmax)
Xtest = data_output[[1]]
Ytest = data_output[[2]]

#####
# Set parameters to be used by the SVM. #
# Also create matrices for storing the results. #
#####

Cpar = 1
opt.C = Cpar

Fmat_svm = matrix(0,Ntest,K)
Fmat_sel_svm = matrix(0,Ntest,K)
Zfull = matrix(0,nrow=Ntest,ncol=K)
ranksfull = matrix(0,nrow=Ntest,ncol=K)
Zsel = matrix(0,nrow=Ntest,ncol=K)
rankssel = matrix(0,nrow=Ntest,ncol=K)

```

```
#####
# Perform binary relevance analysis using all the features      #
#####

# First scale the training and test data
meanfull =
  matrix(apply(Xtrain[,1:ncol(Xtrain)],2,mean),ncol=ncol(Xtrain))
varifull =
  matrix(apply(Xtrain[,1:ncol(Xtrain)],2,var),ncol=ncol(Xtrain))
Xtrainscalesvm = scale(Xtrain)
Xtestscalsvm = scale(Xtest,center=meanfull,scale=sqrt(varifull))

# Define SVM parameters
findgam = sigest(Xtrain,frac=1,scaled=TRUE)
opt.gam = findgam[2]
rbf = rbfdot(sigma=opt.gam)
gmat = kernelMatrix(rbf,Xtestscalsvm,Xtrainscalesvm)

# Start of binary relevance loop
for (j in 1:K) {
  Ytrainsvm = Ytrain[,j]
  Ytrainsvm[Ytrain[,j]==0] = -1
  Ytrainsvm = factor(Ytrainsvm)
  svmfit = ksvm(x=Xtrainscalesvm,y=Ytrainsvm,type="C-svc",
               kernel="rbfdot",kpar=list(sigma=opt.gam),
               C=Cpar,prob.model=FALSE)
  coefsvm = as.matrix(unlist(coef(svmfit)))
  bcoef = unlist(b(svmfit))
  indeks = unlist(SVindex(svmfit))
  coefvol = rep(0,Ntrain)
  coefvol[indeks] = coefsvm
  q3 = matrix(t(as.matrix(coefvol))%*%t(gmat),ncol=1)
  fvalues = q3-bcoef[1]
  Fmat_svm[,j] = fvalues
}

for (i in 1:Ntest) {
```

```

    av = Fmat_svm[i,]
    avs = sort(av,decreasing=TRUE,index.return=TRUE)
    Zfull[i,avs$ix[1:numlabels]] = 1
    ranksfull[i,] = rank(-av)
}

a = measures(Ytest,Zfull,ranksfull)
for (j in 1:6) simfullmeasure[mmm,j] = a[[j]]

#####
# Perform binary relevance analysis using only selected features #
#####

# Perform feature selection
variables = selection(alfa,Bselect,K,Xtrain,Ytrain)
Xtrainsel = Xtrain[,variables]
Xtrainscalesvm = scale(Xtrainsel)
Xtestsel = Xtest[,variables]

# Scale training and test data
meansel = matrix(apply(Xtrainsel[,1:ncol(Xtrainsel)],2,mean),
                 ncol=ncol(Xtrainsel))
varsel = matrix(apply(Xtrainsel[,1:ncol(Xtrainsel)],2,var),
                ncol=ncol(Xtrainsel))
Xtestsvmscale = scale(Xtestsel,center=meansel,scale=sqrt(varsel))

# Define SVM parameters
findgam = sigest(Xtrainsel,frac=1,scaled=TRUE)
opt.gam = findgam[2]
rbf = rbfdot(sigma=opt.gam)
gmat = kernelMatrix(rbf,Xtestsvmscale,Xtrainscalesvm)

# Start of binary relevance loop
for (j in 1:K) {
    Ytrainsvm = Ytrain[,j]
    Ytrainsvm[Ytrain[,j]==0] = -1
    Ytrainsvm = factor(Ytrainsvm)

```

```

svmfit = ksvm(x=Xtrainscalesvm,y=Ytrainsvm,type="C-svc",
             kernel="rbfdot",kpar=list(sigma=opt.gam),C=Cpar,
             prob.model=FALSE)
coefsvm = as.matrix(unlist(coef(svmfit)))
bcoef = unlist(b(svmfit))
indeks = unlist(SVindex(svmfit))
coefvol = rep(0,Ntrain)
coefvol[indeks] = coefsvm
q3 = matrix(t(as.matrix(coefvol))%*%t(gmat),ncol=1)
fvalues = q3 - bcoef[1]
Fmatsel_svm[,j] = fvalues
}

for (i in 1:Ntest) {
  av = Fmatsel_svm[i,]
  avs = sort(av,decreasing=TRUE,index.return=TRUE)
  Zsel[i,avs$ix[1:numlabels]] = 1
  rankssel[i,] = rank(-av)
}

simselfvars[mmm,variables] = 1
a = measures(Ytest,Zsel,rankssel)
for (j in 1:6) simselmeasure[mmm,j] = a[[j]]
}

output = list(simfullmeasure,simselmeasure,simselfvars0)
return(output)
}

```

A.2 Simulation study – data generation

```
#####
# This function is called with 7 arguments:                                     #
# K: The number of labels                                                       #
# N: the number of data cases to be generated                                 #
# densities: label densities                                                    #
# sigmamat: correlation matrix for correlations between labels                 #
# p: The number of features generated                                          #
# prel: The number of relevant features                                       #
# sigmax: correlation matrix for correlations between features                 #
#####

function (K,N,densities,sigmamat,p,prel,sigmax) {

# Define matrices to be used later on in the program
xmat = matrix(0,N,p)
ymat = matrix(0,N,K)
thethamat = matrix(0,p,K)

for (j in 1:K){
  thethatmat[1:prel,j] = runif(prel,j*0.49,j*0.51)
  thethatmat[(prel+1):p,j] = rep(0,p-prel)
}

cvek = qnorm(densities,mean=0,sd=1,lower.tail=FALSE)
itel = 0
while (itel < N) {
  tmat = mvrnorm(1,rep(0,K),sigmamat)
  yvek = as.numeric(tmat>cvek)
  if (sum(yvek)>0) {
    itel = itel+1
    ymat[itel,] = yvek
    indeks = which(ymat[itel,1]==1)
    for (m in 1:length(indeks)) xmat[itel,] =
      xmat[itel,]+mvrnorm(1,thethamat[,indeks[m]],
        sigmax)
  }
}
}

```

```
        xmat[itel,] = xmat[itel,]/length(indeks)
    }}

data_output = list(xmat,ymat)
return(data_output)

}
```

A.3 Simulation study – feature selection

```
#####
# This function is called with 5 arguments:                                     #
# alfa: value for judging relevance of features                               #
# B: the number of randomly permuted matrices computed                       #
# K: The number of labels                                                    #
# xmat: matrix containing X data                                             #
# ymat: matrix containing Y data                                             #
#####

function(alfa,B,K,xmat,ymat)
{
  alfaB = floor(alfa * B)
  xmat = as.matrix(xmat)
  ymat = as.matrix(ymat)
  N = nrow(xmat)
  p = ncol(xmat)
  Zmat = matrix(0,N,p)
  Lmat = ymat
  CormatLX = matrix(0,K,p)
  CormatLZ = rep(0, K * p * B)
  dim(CormatLZ) = c(K,p,B)

  for (k in 1:K) for (j in 1:p) CormatLX[k,j] =
    abs(cor(Lmat[,k],xmat[,j]))
  for (ir in 1:B) {
    indekse = sample(1:N,N,replace = FALSE)
    Zmat = xmat[indekse,]
    for (k in 1:K) for (j in 1:p) CormatLZ[k,j,ir] =
      abs(cor(Lmat[,k],Zmat[,j]))
  }
  Cmat = matrix(0,K,p)
  Amat = matrix(0,K,p)
  for (k in 1:K) for (j in 1:p) {
    rvec = CormatLZ[k,j,]
    wvec = sort(rvec,decreasing=FALSE)
  }
}
```

```
    Cmat[k,j] = wvec[alfaB]
    if (CormatLX[k,j] > Cmat[k,j])
      }
variables = which(apply(Amat,2,sum) > 1)
return(variables)
}
```

A.4 Multi-label evaluation measures

This program computes various measures of classification accuracy for multi-label problems.

```
#####
# The input to the program is:                                     #
# ylabels: an indicator matrix containing the true labels for a   #
# set of Nnew new cases                                         #
# zlabels: an indicator matrix containing the predicted labels for #
# the Nnew new cases                                           #
# rankedlabels: the structure of this matrix is as follows: in  #
# row we have the ranks 1,2,...,1 with rank 1 signifying the    #
# most relevant label, etc.                                     #
# The following measures are computed:                           #
#                                                                 #
# Example-based measures:                                       #
# 1. Hamming Loss (Hloss)                                       #
# 2. Precision (precision)                                     #
# 3. Recall (recall)                                           #
# 4. Accuracy (accuracy)                                       #
#                                                                 #
# Rankings-based measures:                                     #
# 1. One Error (one.error)                                     #
# 2. Coverage (coverage)                                       #
#####

function (ylabels,zlabels,rankedlabels) {

Nnew = nrow(ylabels)
q = ncol(ylabels)

#First compute the example-based measures
yminz = ylabels-zlabels
yprodz = ylabels*zlabels
ydeltaz = apply(yminz,1,function(x) sum(abs(x)))
nylabels = apply(ylabels,1,sum)
```

```
nzlabels = apply(zlabels,1,sum)

proportion1 = ydeltaz/q
yintersectionz = apply(yprodz,1,sum)
yunionz = nylabels + nzlabels - yintersectionz
proportion2 = yintersectionz[nzlabels>0]/nzlabels[nzlabels>0]
proportion3 = yintersectionz/nylabels
proportion4 = yintersectionz/(nylabels+nzlabels)
proportion5 = yintersectionz/yunionz

Hloss = mean(proportion1)
precision = mean(proportion2)
recall = mean(proportion3)
accuracy = mean(proportion5)

#Now compute the rankings-based measures
ylabtimesrank = ylabels*rankedlabels
not.ylabtimesrank =
    (matrix(1,nrow=Nnew,ncol=q)-ylabels)*rankedlabels

one.error = mean(apply(ylabtimesrank,1,function(x) min(x[x>0])!=1))
coverage = mean(apply(ylabtimesrank,1,max)/apply(ylabels,1,sum))-1

output = list(Hloss,precision,recall,accuracy,one.error,coverage)
return(output)

}
```

A.5 Instrument recognition – main program

This is the main program used in the empirical study described in Chapter 8. It contains calls to sub-programs for data sampling (Appendix A.x), feature selection (Appendix A.x) and the calculation of multi-label evaluation measures (Appendix A.x).

```
#####
# This function is called with 5 arguments:                                     #
# alfa: Parameter used in the probe variable selection function               #
# Bselect: Parameter used in the probe variable selection function            #
# Btrain: The number of random training sets to be generated                 #
# K: The number of musical instruments (labels) present                      #
# U: Parameter used in the probe variable selection function                 #
#####

function (alfa,Bselect,Btrain,K,U) {

# Call the library packages required
library(kknn)
library(e1071)
library(kernlab)

# Read in the data files
mixtrain = read.csv("C:\\instrumentsMixTrain.csv",header=TRUE)
singletrain = read.csv("C:\\singleInstrumentsTrain.csv",header=TRUE)
testdata = read.csv("C:\\instrumentsTest.csv",header=TRUE)

# Remove Flux variable from datasets
mixtrain = as.matrix(mixtrain[,-22])
singletrain = as.matrix(singletrain[,-22])
testdata = as.matrix(testdata[,-22])

# Split test data into X and Y parts
nkol = ncol(testdata)
Xtestkeep = testdata[,1:(nkol-2)]
Ytestkeep = testdata[, (nkol-1):nkol]
Ytest = Ytestkeep
```

```

Ntest = nrow(Xtestkeep)

# Define matrices to be used later in the program
Fmat_svmfull = matrix(0,Ntest,K)
Fmat_svmsel = matrix(0,Ntest,K)
rankedlabels_svmfull = matrix(0,Ntest,K)
rankedlabels_svmsel = matrix(0,Ntest,K)

postprobmat_knnfull = matrix(0,Ntest,K)
postprobmat_knnsel = matrix(0,Ntest,K)
rankedlabels_knnfull = matrix(0,Ntest,K)
rankedlabels_knnsel = matrix(0,Ntest,K)

selectmat = matrix(0,Btrain,nkol-2)
AAmat = matrix(0,K,nkol-2)

svm_afull = matrix(0,Ntest,6)
svm_asel = matrix(0,Ntest,6)
knn_afull = matrix(0,Ntest,6)
knn_asel = matrix(0,Ntest,6)
svm_measfull = matrix(0,Btrain,6)
svm_meassel = matrix(0,Btrain,6)
knn_measfull = matrix(0,Btrain,6)
knn_meassel = matrix(0,Btrain,6)

#####
# The simulation loop starts.                                     #
# In this loop we repeatedly generate new training data from the #
# large input training matrices                                 #
#####

for (ib in 1:Btrain) {

# remove SVM kernel matrices to avoid memory problems
if (ib>1) {
  rm(gmat1full,gmat1sel,gmat2full,gmat2sel,gmat3full,gmat3sel)
  gc()
}
}

```

```

    }
# Call sub-program to create sample training data
data_output = Pdatamake(mixtrain,singletrain)
Xtrain_full = as.matrix(data_output[[1]])
Ytrain_full = as.vector(data_output[[2]])
Ntrain = nrow(Xtrain_full)

#####
# Variable selection is performed and the relevant columns of the #
# training and test data matrices are retained, thereby forming #
# matrices Xtrain and Xtest. #
# Also set Ytrain. #
#####

var_output =
  Pselectvariables(alfa,Bselect,K,Xtrain_full,Ytrain_full,U)
variables = var_output[[1]]
selectmat[ib,variables] = 1
Aamat = Aamat + var_output[[2]]

# Create training dataset with all features
pfull = ncol(Xtrain_full)
Xtrainfull = Xtrain_full
Xtestfull = Xtestkeep
Ytrainfull = Ytrain_full

# Create training dataset with selected features
psel = length(variables)
Xtrainsel = Xtrain_full[,variables]
Xtestsel = Xtestkeep[,variables]
Ytrainsel = Ytrain_full

#####
# We ensure that the rows of Xtrainfull and Xtrainsel are ordered #
# successive groups corresponding to the different instruments #
#####

```

```

# Sort full feature dataset
ysortfull = sort(Ytrainfull,decreasing=FALSE,index.return=TRUE)
xsortfull = Xtrainfull[ysortfull$ix,]
Xtrainfull = as.matrix(xsortfull)
yyfull = Ytrainfull[ysortfull$ix]
Ytrainfull = as.vector(yyfull)

# Sort selected feature dataset
ysortsel = sort(Ytrainset,decreasing=FALSE,index.return=TRUE)
xsortsel = Xtrainset[ysortsel$ix,]
Xtrainset = as.matrix(xsortsel)
yyset = Ytrainset[ysortsel$ix]
Ytrainset = as.vector(yyset)

#####
# Compute the number of training data cases for each of the labels, #
# as well as their total and the cumulative totals. #
#####

traincount = NULL
for (j in 1:K) traincount[j] = sum(Ytrainfull==j)
totalcount = sum(traincount)
cumtraincount = cumsum(traincount)

#####
# Set parameters to be used by the SVM #
#####

opt.gamfull = 1/pfull
opt.gamset = 1/pset
opt.C = Cpar
rbfset = rbfdot(sigma=opt.gamset)
rbffull = rbfdot(sigma=opt.gamfull)

```

```
#####
# Certain computations for the SVM can be done outside the binary #
# relevance loop. #
# #####

# Scale the full feature training and test datasets
meanfull = matrix(apply(Xtrainfull[,1:ncol(Xtrainfull)],2,mean),
                  ncol=ncol(Xtrainfull))
variancefull = matrix(apply(Xtrainfull[,1:ncol(Xtrainfull)],2,var),
                      ncol=ncol(Xtrainfull))
Xtrainscalesvmfull = scale(Xtrainfull)
Xtestsvmsvmfull =
  scale(Xtestfull,center=meanfull,scale=sqrt(variancefull))

# Calculate kernel matrices for full feature training set for SVM
# This is done in three steps to avoid memory problems in R due to
# large matrices.

gmat1full = kernelMatrix(rbffull,Xtestsvmsvmfull[1:5000,],
                         Xtrainscalesvmfull)
gmat2full = kernelMatrix(rbffull,Xtestsvmsvmfull[5001:10000,],
                         Xtrainscalesvmfull)
gmat3full = kernelMatrix(rbffull,Xtestsvmsvmfull[10001:14662,],
                         Xtrainscalesvmfull)

# Scale the selected feature training and test datasets
meansel = matrix(apply(Xtrainsel[,1:ncol(Xtrainsel)],2,mean),
                 ncol=ncol(Xtrainsel))
variancesel = matrix(apply(Xtrainsel[,1:ncol(Xtrainsel)],2,var),
                    ncol=ncol(Xtrainsel))
Xtrainscalesvmssel = scale(Xtrainsel)
Xtestsvmsvmssel =
  scale(Xtestsel,center=meansel,scale=sqrt(variancesel))

# Calculate kernel matrices for selected feature training set
# This is again done in three steps.
```

```

gmat1sel = kernelMatrix(rbfSel,XtestScalesvmsel[1:5000,],
                        Xtrainscalesvmsel)
gmat2sel = kernelMatrix(rbfSel,XtestScalesvmsel[5001:10000,],
                        Xtrainscalesvmsel)
gmat3sel = kernelMatrix(rbfSel,XtestScalesvmsel[10001:14662,],
                        Xtrainscalesvmsel)

#####
# The binary relevance loop starts here.                                     #
#####

For (j in 1:K) {

#####
# First we fit the SVM                                                         #
#####

# Fit SVM on full feature training dataset
Ytrainsvmfull = NULL
Ytrainsvmfull[1:totalcount] = -1
if (j==1) begin = 1 else begin = cumtraincount[j-1]+1
eindig = cumtraincount[j]
Ytrainsvmfull[begin:eindig] = 1
Ytrainsvmfull = factor(Ytrainsvmfull)

svmfitfull = ksvm(x=Xtrainscalesvmfull,y=Ytrainsvmfull,type="C-svc",
                 kernel="rbfdot",kpar=list(sigma=1/pfull),C=Cpar,prob.model=FALSE)

coefsvmfull = as.matrix(unlist(coef(svmfitfull)))
bcoefffull = unlist(b(svmfitfull))
indexfull = unlist(SVindex(svmfitfull))
coefvolfull = rep(0,totalcount)
coefvolfull[indeksfull] = coefsvmfull
q3full = matrix(t(as.matrix(coefvolfull))%*%t(gmat1full,ncol=1))
fvaluesfull = q3full-bcoefffull[1]
Fmat_svmfull[1:5000,j] = fvaluesfull
q3full = matrix(t(as.matrix(coefvolfull))%*%t(gmat2full,ncol=1))

```

```

fvaluesfull = q3full-bcoefffull[1]
Fmat_svmfull[5001:10000,j] = fvaluesfull
q3full = matrix(t(as.matrix(coefvolfull))%*%t(gmat3full,ncol=1)
fvaluesfull = q3full-bcoefffull[1]
Fmat_svmfull[10001:14662,j] = fvaluesfull

# Fit SVM on selected feature training dataset
Ytrainsvmsel = NULL
Ytrainsvmsel[1:totalcount] = -1
if (j==1) begin = 1 else begin = cumtraincount[j-1]+1
eindig = cumtraincount[j]
Ytrainsvmsel[begin:eindig] = 1
Ytrainsvmsel = factor(Ytrainsvmsel)

svmfitsel = ksvm(x=Xtrainscalesvmsel,y=Ytrainsvmsel,type="C-svc",
  kernel="rbfdot",kpar=list(sigma=1/psel),C=Cpar,prob.model=FALSE)

coefsvmsel = as.matrix(unlist(coef(svmfitsel)))
bcoefssel = unlist(b(svmfitsel))
indexsel = unlist(SVindex(svmfitsel))
coefvolssel = rep(0,totalcount)
coefvolssel[indeksel] = coefsvmsel
q3sel = matrix(t(as.matrix(coefvolssel))%*%t(gmat1sel,ncol=1)
fvaluessel = q3sel-bcoefssel[1]
Fmat_svmssel[1:5000,j] = fvaluessel
q3sel = matrix(t(as.matrix(coefvolssel))%*%t(gmat2sel,ncol=1)
fvaluessel = q3full-bcoefssel[1]
Fmat_svmssel[5001:10000,j] = fvaluessel
q3sel = matrix(t(as.matrix(coefvolssel))%*%t(gmat3sel,ncol=1)
fvaluessel = q3sel-bcoefssel[1]
Fmat_svmssel[10001:14662,j] = fvaluessel

#####
# For kNN we use a sample from the rows of the training data,      #
# constructed in such a way that the number of positives is      #
# approximately equal to the number of negatives.                  #
#####

```

```

# Prepare to take a stratified sample of Xtrain
numberofones = traincount[j]
fraction = numberofones/totalcount

# The following quantities Ntrain and Ntrainvec refer to the #
# stratified sample #

Ntrainvec = NULL
for (jj in 1:K) {if (jj=j) Ntrainvec = c(Ntrainvec,numberofones)
  else Ntrainvec = c(Ntrainvec,round(fraction*traincount[jj]))}
Ntrain = sum(Ntrainvec)
instrpointertrain = cumsum(Ntrainvec)

# Draw a stratified random sample from the rows of Xtrain
trainindekse = NULL
for (jj in 1:K) {
  if (jj==1) begin = 1 else begin = cumtraincount[jj-1]+1
  eindig = cumtraincount[jj]
  if (jj==j) trainindekse = c(trainindekse,begin:eindig)
  if (jj!=j) {
    aantal = Ntrainvec[jj]
    trainindekse =
      c(trainindekse,sample(begin:eindig,aantal,replace=FALSE))
  }
}

# Fit kNN on full feature training dataset
Xtrainknnfull = Xtrainfull[trainindekse,]
Ytrainknnfull = NULL
meanfull = matrix(apply(Xtrainknnfull[,1:ncol(Xtrainknnfull)],2,
  mean),ncol=ncol(Xtrainknnfull))
variancefull = matrix(apply(Xtrainknnfull[,1:ncol(Xtrainknnfull)],
  2,var),ncol=ncol(Xtrainknnfull))
Xtrainscaleknnfull = scale(Xtrainknnfull)
Xtestscaleknnfull = scale(Xtestfull,center=meanfull,
  Scale=sqrt(variancefull))
Ytrainknnfull[1:Ntrain] = -1

```

```

if (j==1) begin = 1 else begin = instrpointertrain[j-1]+1
eindig = instrpointertrain[j]
Ytrainknnfull[begin:eindig] = 1
Ytrainknnfull = factor(Ytrainknnfull)
dataframefull = data.frame(Xtrainscaleknnfull,Ytrainknnfull)

knnfitfull = kknn(Ytrainknnfull ~ .,train=dataframefull,
                  test=data.frame(Xtestscaleknnfull),11, distance=2,
                  kernel="gaussian")
postprobat_knnfull[,j] = knnfitfull$prob[,2]

# Fit kNN on selected feature training dataset
Xtrainknnssel = Xtrainssel[trainindekse,]
Ytrainknnssel = NULL
meansel = matrix(apply(Xtrainknnssel[,1:ncol(Xtrainknnssel)],2,
                      mean),ncol=ncol(Xtrainknnssel))
variancesel = matrix(apply(Xtrainknnssel[,1:ncol(Xtrainknnssel)],
                          2,var),ncol=ncol(Xtrainknnssel))
Xtrainscaleknnssel = scale(Xtrainknnssel)
Xtestscaleknnssel = scale(Xtestssel,center=meansel,
                          Scale=sqrt(variancesel))
Ytrainknnssel[1:Ntrain] = -1
if (j==1) begin = 1 else begin = instrpointertrain[j-1]+1
eindig = instrpointertrain[j]
Ytrainknnssel[begin:eindig] = 1
Ytrainknnssel = factor(Ytrainknnssel)
dataframesel = data.frame(Xtrainscaleknnssel,Ytrainknnssel)

knnfitsel = kknn(Ytrainknnssel ~ .,train=dataframesel,
                  test=data.frame(Xtestscaleknnssel),11, distance=2,
                  kernel="gaussian")
postprobat_knnssel[,j] = knnfitsel$prob[,2]

}

```

```
#####
# For each test case, rank the SVM f-values and the kNN posterior #
# probabilities according to the binary relevance method.      #
#####

# First define the matrices needed
trueinstr = matrix(0,Ntest,K)
rankedlabels_svmfull = matrix(0,Ntest,K)
rankedlabels_knnfull = matrix(0,Ntest,K)
rankedlabels_svmsel = matrix(0,Ntest,K)
rankedlabels_knnssel = matrix(0,Ntest,K)
predictedinstruments_svmfull = matrix(0,Ntest,K)
predictedinstruments_knnfull = matrix(0,Ntest,K)
predictedinstruments_svmsel = matrix(0,Ntest,K)
predictedinstruments_knnssel = matrix(0,Ntest,K)

for (i in 1:Ntest) {
  trueinstr[i,Ytest[i,]] = 1

  postsortedfull = sort(Fmat_svmfull[i,],decreasing=TRUE,
                        index.return=TRUE)
  postsortedsel = sort(Fmat_svmsel[i,],decreasing=TRUE,
                       index.return=TRUE)
  rankedlabels_svmfull[i,] = postsortedfull$ix
  rankedlabels_svmsel[i,] = postsortedsel$ix
  predictedinstruments_svmfull[i,postsortedfull$ix[1:2]] = 1
  predictedinstruments_svmsel[i,postsortedsel$ix[1:2]] = 1

  postsortedfull = sort(postprobmat_knnfull[i,],decreasing=TRUE,
                        index.return=TRUE)
  postsortedsel = sort(postprobmat_knnssel[i,],decreasing=TRUE,
                       index.return=TRUE)
  rankedlabels_knnfull[i,] = postsortedfull$ix
  rankedlabels_knnssel[i,] = postsortedsel$ix
  predictedinstruments_knnfull[i,postsortedfull$ix[1:2]] = 1
  predictedinstruments_knnssel[i,postsortedsel$ix[1:2]] = 1
}

```

```

#####
# Now calculate multi-label evaluation measures                                     #
#####

svm_afull = measures(trueinstr,predictedinstruments_svmfull,
                    rankedlabels_svmfull)
  for (j in 1:6) svm_measfull[ib,j] = svm_afull[[j]]

svm_asel = measures(trueinstr,predictedinstruments_svmsel,
                    rankedlabels_svmsel)
  for (j in 1:6) svm_meassel[ib,j] = svm_asel[[j]]

knn_afull = measures(trueinstr,predictedinstruments_knnfull,
                    rankedlabels_knnfull)
  for (j in 1:6) knn_measfull[ib,j] = knn_afull[[j]]

knn_asel = measures(trueinstr,predictedinstruments_knnsel,
                    rankedlabels_knnsel)
  for (j in 1:6) knn_meassel[ib,j] = knn_asel[[j]]

#####
# We reach the end of the simulation loop.                                     #
#####

}

Aamat = AAmat/Btrain
output = list(svm_measfull,svm_meassel,knn_measfull,knn_meassel,
             selectmat,AAmat)

}

```

A.6 Instrument recognition – program for data sampling

```
#####
# This function is called with 2 arguments:                                     #
# mixinstrtrain: this is the mixture instrument dataset                         #
# singleinstrtrain: this is the single instrument dataset                       #
#####

function (mixinstrtrain,singleinstrtrain) {

nmix = 750
nsingle = 30

# First draw a sample of the mixture data
indeksemix = sample(1:nrow(mixinstrtrain),nmix,replace=FALSE)
xmix = mixinstrtrain[indeksemix,1:122]
ymix = mixinstrtrain[indeksemix,123:124]
y1 = ymix[,1]
y2 = ymix[,2]
xmix2 = rbind(xmix,xmix)
yvek = c(y1,y2)
xymix = cbind(xmix2,yvek)

# Now draw a sample from the single instrument data, 30 of each               #
# instrument                                                                    #
ncount = NULL
for (j in 1:19) ncount[j] = sum(singleinstrtrain[,123]==j)
cumncount = cumsum(ncount)
indeksesingle = NULL
for (j in 1:19) {
  if (j==1) begin = 1 else begin = cumncount[j-1]+1
  eindig = cumncount[j]
  indeksesingle = c(indeksesingle,sample(begin:eindig,nsingle,
    replace=FALSE))
}
xsingle = singleinstrtrain[indeksesingle,1:122]
ysingle = singleinstrtrain[indeksesingle,123]
```

```
xysingle = cbind(xsingle, ysingle)

colnames(xymix) = colnames(xysingle)
xy = rbind(xymix, xysingle)

xmat = xy[, 1:122]
yvek = xy[, 123]

dataoutput = list(xmat, yvek)

return(dataoutput)

}
```

A.7 Instrument recognition – program for feature selection

```
#####
# This program performs feature selection by using probe variables.#
# The context is binary relevance in a multi-label setting.      #
# There are 6 input parameters:                                  #
# alfa: the significance level for identifying irrelevant variables#
# B: the number of repetitions to determine critical values      #
# K: the number of labels (instruments)                         #
# Xmat: the matrix of training values on all the variables      #
# Yvec: the vector giving the group membership for each data case #
# U: the feature cut-off point indicating how many labels a    #
# feature should be relevant for in order to be selected      #
#####

function (alfa,B,K,Xmat,Yvec,U) {

  alfaB = floor(alfa*B)
  Xmat = as.matrix(Xmat)
  Yvec = as.vector(Yvec)
  N = nrow(Xmat)
  P = ncol(Xmat)

  Zmat = matrix(0,N,p)
  Lmat = matrix(0,N,K)
  for (j in 1:K) Lmat[Yvec==j,j]=1
  CormatLX = matrix(0,K,p)
  CormatLZ = rep(0,K*p*B)
  dim(CormatLZ) = c(K,p,B)

  for (k in 1:K) for (j in 1:p) CormatLX[k,j] =
    abs(cor(Lmat[,k],Xmat[,j]))

  for (ir in 1:B) {
    indekse = sample(1:N,N,replace=TRUE)
    Zmat=Xmat[indekse,]
    for (k in 1:K) for (j in 1:p) CormatLZ[k,j,ir] =
```

```
      abs(cor(Lmat[,k],Zmat[,j]))
    }
  Cmat = matrix(0,K,p)
  Amat = matrix(0,K,p)

  for (k in 1:K) for (j in 1:p) {
    rvec = CormatLZ[k,j,]
    wvec = sort(rvec,decreasing=FALSE)
    Cmat[k,j] = wvec[alfaB]
    if(CormatLX[k,j]>Cmat[k,j]) Amat[k,j]=1
  }

  variables = which(apply(Amat,2,sum)>U)

  return(list(variables,Amat))

}
```

APPENDIX B

Detailed results from simulation study

This appendix contains the detailed results from the simulation study described in Chapter 7. Appendix B.1 shows the initial runs performed to obtain a suitable value for the hyperparameter C for the SVM. Appendix B.2 shows the values of the different error measures for each of the 72 parameter configurations (as described in Chapter 7) for the cases where no feature selection was performed, while Appendix B.3 shows the values of these error measures in the case where feature selection was performed. Appendix B.4 lists the number of relevant and irrelevant features selected by the feature selection procedure for each of the 72 parameter configurations.

B.1 Results of initial simulation runs – Determining a value for SVM hyperparameter C

Configuration number	Total nr. of features	Nr. of relevant features	Feature correlations	SVM hyperparameter	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	Σ_X	C						
A	100	20	$\Sigma_{X_{uncorr}}$	0.1	0.28007	0.81180	0.83260	0.66735	0.07060	0.08692
B	1000	100	$\Sigma_{X_{uncorr}}$	0.1	0.28035	0.81078	0.83301	0.66675	0.06846	0.09254
C	100	20	$\Sigma_{X_{uncorr}}$	1	0.28005	0.81244	0.83195	0.66779	0.07019	0.08762
D	1000	100	$\Sigma_{X_{uncorr}}$	1	0.27969	0.81134	0.83338	0.66753	0.07008	0.09404
E	100	20	$\Sigma_{X_{uncorr}}$	10	0.30237	0.79487	0.81242	0.64623	0.07489	0.10976
F	1000	100	$\Sigma_{X_{uncorr}}$	10	0.28031	0.81191	0.83235	0.66735	0.06803	0.09148
G	100	20	$\Sigma_{X_{uncorr}}$	100	0.30155	0.79504	0.81321	0.64697	0.07432	0.10907
H	1000	100	$\Sigma_{X_{uncorr}}$	100	0.28037	0.81173	0.83194	0.66746	0.06935	0.09324
I	100	20	$\Sigma_{X_{relcorr}}$	0.1	0.28204	0.81020	0.83058	0.66556	0.10224	0.11940
J	100	20	$\Sigma_{X_{relrrcorr}}$	0.1	0.28175	0.81056	0.83072	0.66583	0.10012	0.11813
K	100	20	$\Sigma_{X_{relcorr}}$	100	0.31506	0.78374	0.80038	0.63419	0.11381	0.15880
L	100	20	$\Sigma_{X_{relrrcorr}}$	100	0.31246	0.78789	0.80117	0.63749	0.11089	0.15425
M	1000	100	$\Sigma_{X_{relcorr}}$	100	0.25345	0.80894	0.82958	0.66409	0.08145	0.10551
N	1000	100	$\Sigma_{X_{relrrcorr}}$	100	0.28360	0.80836	0.82972	0.66380	0.08189	0.10645

Other parameter values:

- $N_{train} = 100$
- $\Sigma_{Y_{uncorr}}$
- $K = 3$

B.2 Detailed results of simulation runs for different parameter configurations – NO feature selection

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	K	N_{train}	Σ_X	Σ_Y						
1	100	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.28095	0.81046	0.83257	0.66622	0.07124	0.08780
2	100	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.29022	0.80483	0.82331	0.65781	0.07766	0.10434
3	100	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.28247	0.80899	0.83062	0.66492	0.07810	0.09724
4	100	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{uncorr}}$	0.38744	0.54757	0.82960	0.51462	0.31357	0.42636
5	100	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.41974	0.52542	0.78616	0.48919	0.38879	0.53319
6	100	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.39017	0.54632	0.82469	0.51304	0.30173	0.42185
7	200	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.28694	0.80618	0.82633	0.66090	0.07456	0.10244
8	200	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.28858	0.80465	0.82604	0.65889	0.07525	0.10334
9	200	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.28132	0.80978	0.83235	0.66570	0.07666	0.09571
10	200	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.40716	0.53406	0.80335	0.49908	0.36235	0.49687
11	200	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.42165	0.52248	0.78462	0.48702	0.39539	0.54194
12	200	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.41215	0.52978	0.79701	0.49470	0.35606	0.49571
13	100	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.33130	0.70903	0.84085	0.57610	0.10999	0.24777
14	100	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.33801	0.70307	0.83458	0.56914	0.15715	0.31243
15	100	20	6	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.32973	0.70962	0.84197	0.57766	0.10557	0.24043
16	100	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.44566	0.41918	0.85060	0.39229	0.39538	0.84502
17	100	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.46868	0.40246	0.81177	0.37241	0.45411	1.01555
18	100	20	6	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.44725	0.41829	0.84551	0.39145	0.40227	0.86233
19	200	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.33008	0.71021	0.84085	0.57773	0.14091	0.28279

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	K	N_{train}	Σ_X	Σ_Y						
20	200	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.33254	0.70865	0.83854	0.57522	0.15506	0.30649
21	200	20	6	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.32837	0.71272	0.84060	0.58001	0.11288	0.25596
22	200	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.45237	0.41470	0.84022	0.38658	0.40631	0.89075
23	200	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.46966	0.40053	0.81067	0.37095	0.46062	1.03158
24	200	20	6	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.45291	0.41421	0.83581	0.38663	0.42058	0.91786
25	100	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.36699	0.63889	0.84545	0.51498	0.20630	0.62675
26	100	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.37024	0.63549	0.84296	0.51137	0.21977	0.66477
27	100	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.36653	0.63791	0.84656	0.51489	0.19672	0.62059
28	100	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.49151	0.35967	0.79864	0.33076	0.47378	1.61772
29	100	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.49852	0.35419	0.78618	0.32424	0.49072	1.68581
30	100	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.49257	0.35850	0.79726	0.32954	0.48441	1.62681
31	200	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.36571	0.64001	0.84681	0.51625	0.21458	0.64869
32	200	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.36599	0.63964	0.84688	0.51594	0.21485	0.65524
33	200	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.36668	0.63838	0.84624	0.51494	0.19850	0.62455
34	200	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.49217	0.35845	0.79845	0.32971	0.48377	1.62862
35	200	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.49604	0.35571	0.79094	0.32627	0.48761	1.67069
36	200	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.49313	0.35808	0.79638	0.32907	0.49452	1.63506
37	100	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.28032	0.81069	0.83482	0.66605	0.06810	0.07402
38	100	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.28718	0.80520	0.82785	0.65982	0.08302	0.10121
39	100	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.27557	0.81391	0.83655	0.67172	0.08027	0.08586
40	100	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.33231	0.59000	0.89516	0.56189	0.23340	0.28260
41	100	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.41268	0.52924	0.79641	0.49437	0.38229	0.51584

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	K	N_{train}	Σ_X	Σ_Y						
42	100	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.32232	0.59648	0.90634	0.57048	0.20795	0.23890
43	200	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.27856	0.81282	0.83357	0.66907	0.08245	0.09470
44	200	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.28291	0.81002	0.83024	0.66459	0.08144	0.09826
45	200	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.27705	0.81338	0.83542	0.67008	0.08421	0.09034
46	200	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.34622	0.57949	0.87549	0.55105	0.24363	0.31221
47	200	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.41442	0.52779	0.79505	0.49254	0.38196	0.51514
48	200	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.33212	0.59002	0.89021	0.56379	0.21702	0.26700
49	100	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.32571	0.71373	0.84480	0.58208	0.07523	0.18717
50	100	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.33442	0.70763	0.83740	0.57349	0.12333	0.27053
51	100	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.32707	0.71282	0.84415	0.58073	0.07542	0.18641
52	100	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.42228	0.43678	0.89366	0.41259	0.36284	0.69238
53	100	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.45686	0.41164	0.83092	0.38293	0.44776	0.96447
54	100	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.42125	0.43773	0.89219	0.41397	0.35790	0.68230
55	200	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.32587	0.71346	0.84472	0.58184	0.07300	0.19081
56	200	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.33383	0.70753	0.83799	0.57398	0.12274	0.26931
57	200	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.32630	0.71319	0.84512	0.58144	0.07556	0.18615
58	200	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.42759	0.43308	0.88423	0.40806	0.37554	0.73444
59	200	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.45676	0.41090	0.83200	0.38256	0.44848	0.96530
60	200	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.42417	0.43556	0.88725	0.41144	0.36459	0.70819
61	100	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.36304	0.64105	0.85090	0.51867	0.15665	0.52253
62	100	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.36802	0.63722	0.84726	0.51360	0.17974	0.56282
63	100	20	12	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.36472	0.63918	0.84970	0.51645	0.15954	0.51642

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	K	N_{train}	Σ_X	Σ_Y						
64	100	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.48538	0.36413	0.80797	0.33610	0.46380	1.60935
65	100	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.49576	0.35592	0.78832	0.32651	0.48113	1.71677
66	100	20	12	1000	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.48493	0.36415	0.81120	0.33612	0.45636	1.56044
67	200	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.36272	0.64180	0.85063	0.51933	0.16106	0.53180
68	200	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.36668	0.63799	0.84840	0.51503	0.18209	0.56539
69	200	20	12	1000	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.36336	0.64097	0.85020	0.51823	0.16121	0.52201
70	200	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.48506	0.36487	0.80867	0.33670	0.46809	1.60105
71	200	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.49320	0.35782	0.79272	0.32889	0.48464	1.69756
72	200	20	12	1000	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.48444	0.36462	0.81197	0.33677	0.45773	1.56059
73	100	20	12	100	$\Sigma_{X_{uncorr}}$	NEGS*	0.48214	0.39604	0.77226	0.36082	0.46076	1.38708

* Covariance matrix set up with negative correlations of -0.9 instead of positive correlations of +0.9 as in the case of $\Sigma_{Y_{corr}}$

B.3 Detailed results of simulation runs for different parameter configurations – WITH feature selection

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	K	N_{train}	Σ_X	Σ_Y						
1	100	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.28954	0.80402	0.82375	0.65843	0.08267	0.10257
2	100	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.29680	0.79989	0.81716	0.65163	0.10234	0.12533
3	100	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.29260	0.80140	0.82150	0.65530	0.09378	0.11566
4	100	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{uncorr}}$	0.35400	0.57265	0.87128	0.54253	0.27147	0.33905
5	100	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.42284	0.52310	0.78289	0.48640	0.39544	0.54082
6	100	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.40388	0.53604	0.80823	0.50138	0.35370	0.48059
7	200	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.29027	0.80369	0.82281	0.65791	0.08255	0.10561
8	200	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.29743	0.79801	0.81778	0.65057	0.09865	0.12475
9	200	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.28825	0.80458	0.82590	0.65918	0.09086	0.10897
10	200	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.35855	0.57052	0.86329	0.53986	0.27619	0.35265
11	200	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.42423	0.52055	0.78145	0.48485	0.40192	0.54909
12	200	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.40076	0.53832	0.81196	0.50394	0.34380	0.46843
13	100	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.33369	0.70724	0.83842	0.57365	0.10122	0.23777
14	100	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.34263	0.69961	0.82933	0.56472	0.16182	0.31922
15	100	20	6	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.33274	0.70736	0.83877	0.57463	0.11460	0.25347
16	100	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.43817	0.42480	0.86225	0.39909	0.38392	0.78072
17	100	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.47504	0.39769	0.79968	0.36694	0.45605	1.04244
18	100	20	6	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.45371	0.41345	0.83588	0.38564	0.40972	0.88819
19	200	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.33367	0.70751	0.83805	0.57403	0.11665	0.25364

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	K	N_{train}	Σ_X	Σ_Y						
20	200	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.33944	0.70347	0.83160	0.56839	0.15644	0.31194
21	200	20	6	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.33042	0.71117	0.83886	0.57793	0.10700	0.24565
22	200	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.44169	0.42271	0.85577	0.39626	0.38881	0.80258
23	200	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.47331	0.39780	0.80348	0.36785	0.46024	1.04446
24	200	20	6	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.45122	0.41548	0.83984	0.38801	0.46029	0.87577
25	100	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.36949	0.63702	0.84246	0.51241	0.20666	0.62685
26	100	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.37345	0.63308	0.83871	0.50813	0.22264	0.66858
27	100	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.36816	0.63668	0.84434	0.51327	0.19596	0.62431
28	100	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.49406	0.35776	0.79304	0.32837	0.47679	1.64786
29	100	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.50379	0.35024	0.77599	0.31957	0.49442	1.72667
30	100	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.49667	0.35542	0.78996	0.32575	0.48097	1.65718
31	200	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.36909	0.63748	0.84312	0.51283	0.20911	0.64160
32	200	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.36989	0.63671	0.84260	0.51198	0.21603	0.65526
33	200	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.36843	0.63707	0.84420	0.51313	0.19729	0.62194
34	200	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.49256	0.35816	0.79753	0.32927	0.48054	1.63152
35	200	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.49923	0.35332	0.78549	0.32338	0.49394	1.68771
36	200	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.49561	0.36523	0.79202	0.32680	0.48312	1.64642
37	100	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.28089	0.81027	0.83293	0.66596	0.07571	0.08599
38	100	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.29258	0.80115	0.82349	0.65453	0.09116	0.10979
39	100	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.28656	0.80567	0.82737	0.66103	0.08718	0.10286
40	100	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.31202	0.60522	0.92559	0.57711	0.20691	0.21690
41	100	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.41256	0.52933	0.79595	0.49468	0.37926	0.51441

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	K	N_{train}	Σ_X	Σ_Y						
42	100	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.36945	0.56113	0.85158	0.52982	0.31002	0.39499
43	200	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.27820	0.81309	0.83391	0.66941	0.07664	0.08829
44	200	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.29057	0.80428	0.82411	0.65705	0.08917	0.10744
45	200	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.28452	0.80778	0.82884	0.66294	0.08399	0.09710
46	200	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.31583	0.60229	0.92025	0.57412	0.20853	0.22752
47	200	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.41336	0.52859	0.79577	0.49363	0.38200	0.51557
48	200	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.35670	0.57159	0.86671	0.54090	0.28551	0.35618
49	100	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.32553	0.71386	0.84450	0.58226	0.07731	0.18685
50	100	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.33489	0.70727	0.83628	0.57300	0.12982	0.27001
51	100	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.32830	0.71190	0.84244	0.57962	0.08221	0.19667
52	100	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.41955	0.43882	0.89671	0.41535	0.34933	0.65153
53	100	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.46382	0.40641	0.81704	0.37700	0.43958	0.98256
54	100	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.43256	0.42925	0.87334	0.40388	0.37744	0.75118
55	200	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.32611	0.71328	0.84446	0.58148	0.07472	0.18456
56	200	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.33568	0.70615	0.83631	0.57205	0.12735	0.27064
57	200	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.32800	0.71192	0.84323	0.57971	0.07939	0.19163
58	200	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.42087	0.43812	0.89450	0.41432	0.35613	0.67366
59	200	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.46165	0.40723	0.82151	0.37847	0.44417	0.98469
60	200	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.42686	0.43354	0.88297	0.40921	0.36675	0.70849
61	100	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.36312	0.64099	0.85042	0.51852	0.15751	0.50645
62	100	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.36763	0.63751	0.84716	0.51391	0.18443	0.55215
63	100	20	12	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	0.36487	0.63906	0.84913	0.51627	0.16051	0.49689

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Hamming Loss	Precision	Recall	Accuracy	One-Error	Coverage
	p	p_r	K	N_{train}	Σ_X	Σ_Y						
64	100	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.48860	0.36172	0.80290	0.33313	0.46156	1.61254
65	100	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.49897	0.35351	0.78290	0.32367	0.47902	1.71575
66	100	20	12	1000	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.49022	0.36018	0.80245	0.33136	0.45374	1.58760
67	200	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.36328	0.64137	0.84986	0.51861	0.15905	0.52160
68	200	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.36733	0.63751	0.84764	0.51430	0.18291	0.55879
69	200	20	12	1000	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.36402	0.64048	0.84906	0.51751	0.16015	0.50877
70	200	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	0.48727	0.36321	0.80473	0.33466	0.46450	1.60762
71	200	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	0.49671	0.35519	0.78656	0.32572	0.48634	1.70951
72	200	20	12	1000	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	0.48892	0.36126	0.80450	0.33266	0.45841	1.58332
73	100	20	12	100	$\Sigma_{X_{uncorr}}$	NEGS*	0.48344	0.39507	0.77014	0.35942	0.46539	1.40189

* Covariance matrix set up with negative correlations of -0.9 instead of positive correlations of +0.9 as in the case of $\Sigma_{Y_{corr}}$

**B.4 Detailed results of simulation runs for different parameter configurations –
Number of relevant and irrelevant features selected**

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature corre- lations	Label corre- lations	Nr. of relevant features selected	Nr. of irrelevant features selected
	p	p_r	K	N_{train}	Σ_X	Σ_Y		
1	100	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	12.4	3.0
2	100	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	12.5	5.6
3	100	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	12.3	5.7
4	100	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{uncorr}}$	15.4	4.1
5	100	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	15.7	3.6
6	100	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	15.7	5.1
7	200	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	12.6	13.4
8	200	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	11.6	13.5
9	200	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	12.4	12.7
10	200	20	3	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	15.9	9.4
11	200	20	3	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	15.8	8.4
12	200	20	3	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	17.1	8.5
13	100	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	19.5	16.1
14	100	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	19.5	16.7
15	100	20	6	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	19.5	15.2
16	100	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	7.6
17	100	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	7.5
18	100	20	6	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	20.0	7.5
19	200	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	19.5	38.2
20	200	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	19.6	36.6
21	200	20	6	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	19.5	36.9
22	200	20	6	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	16.3
23	200	20	6	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	16.1
24	200	20	6	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	20.0	16.9
25	100	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	27.8
26	100	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	27.5
27	100	20	12	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	20.0	28.4
28	100	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	21.6
29	100	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	21.4
30	100	20	12	100	$\Sigma_{X_{reliircorr}}$	$\Sigma_{Y_{corr}}$	20.0	20.4
31	200	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	19.9	65.9
32	200	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	63.8

Configu- ration number	Total nr. of fea- tures	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature cor- relations	Label cor- relations	Nr. of relevant features selected	Nr. of irrelevant features selected
	p	p_r	K	N_{train}	Σ_X	Σ_Y		
33	200	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	69.1
34	200	20	12	100	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	48.2
35	200	20	12	100	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	48.4
36	200	20	12	100	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	53.7
37	100	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	3.1
38	100	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	5.6
39	100	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	5.2
40	100	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	3.8
41	100	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	3.8
42	100	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	4.9
43	200	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	13.0
44	200	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	13.1
45	200	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	13.3
46	200	20	3	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	8.6
47	200	20	3	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	8.7
48	200	20	3	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	8.6
49	100	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	16.1
50	100	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	15.9
51	100	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	15.6
52	100	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	7.5
53	100	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	7.2
54	100	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	8.1
55	200	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	37.4
56	200	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	37.4
57	200	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	37.2
58	200	20	6	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	16.8
59	200	20	6	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	15.9
60	200	20	6	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	14.7
61	100	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	28.8
62	100	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	28.2
63	100	20	12	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	29.5
64	100	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	21.8
65	100	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	21.8
66	100	20	12	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	20.5
67	200	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	62.8
68	200	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	62.9

Configuration number	Total nr. of features	Nr. of relevant features	Nr. of labels	Nr. of training cases	Feature correlations	Label correlations	Nr. of relevant features selected	Nr. of irrelevant features selected
	p	p_r	K	N_{train}	Σ_X	Σ_Y		
69	200	20	12	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	58.1
70	200	20	12	1000	$\Sigma_{X_{uncorr}}$	$\Sigma_{Y_{corr}}$	20.0	48.2
71	200	20	12	1000	$\Sigma_{X_{relcorr}}$	$\Sigma_{Y_{corr}}$	20.0	48.3
72	200	20	12	1000	$\Sigma_{X_{relirrcorr}}$	$\Sigma_{Y_{corr}}$	20.0	51.0
73	100	20	12	100	$\Sigma_{X_{uncorr}}$	NEGS*	20.0	27.1

* Covariance matrix set up with negative correlations of -0.9 instead of positive correlations of +0.9 as in the case of $\Sigma_{Y_{corr}}$