Automatic Genre Classification of MIDI Recordings

Cory McKay

Music Technology Area
Department of Theory
Faculty of Music
McGill University, Montreal

Submitted June 2004

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Arts

Acknowledgements

Many thanks to my advisor, Professor Ichiro Fujinaga, for his invaluable advice and support during all stages of this thesis. I would also like to thank Professor David Brackett and Professor William Caplin for their discussions and insights on the musicological aspects of this thesis. Thanks also to the faculty, staff and students of the McGill University Music Technology Area, who have provided an excellent environment for discussion, research and learning. The generous financial support of the *Fonds Québécois de la recherche sur la société et la culture* has also been greatly appreciated. Finally, I am especially grateful for the unfailing and unconditional support and encouragement of all kinds from Madeleine McKay, Erin McKay, Don McKay, Thérèse d'Amour and Regan Toews.

Abstract

A software system that automatically classifies MIDI files into hierarchically organized taxonomies of musical genres is presented. This extensible software includes an easy to use and flexible GUI. An extensive library of high-level musical features is compiled, including many original features. A novel hybrid classification system is used that makes use of hierarchical, flat and round robin classification. Both k-nearest neighbour and neural network-based classifiers are used, and feature selection and weighting are performed using genetic algorithms. A thorough review of previous research in automatic genre classification is presented, along with an overview of automatic feature selection and classification techniques. Also included is a discussion of the theoretical issues relating to musical genre, including but not limited to what mechanisms humans use to classify music by genre and how realistic genre taxonomies can be constructed.

Sommaire

Dans cette thèse, nous présentons un système logiciel classifiant automatiquement, et de manière hiérarchique selon leur genre musical, des pièces de musique représentées sous format MIDI. Ce système comprend une interface utilisateur graphique, souple et facile à utiliser. Une collection entendue de caractéristiques musicales, dont plusieurs sont nouvelles, a été compilée. Pour cela on utilise un système original de classification hybride fondée sur des classification hiérarchiques, non-hiérarchique, ou de sélection en tournoi. Les classificateurs de type K plus proches voisins et les réseaux de neurones sont utilisés. Des algorithmes effectuent la sélection et attribuent une pondération à chaque caractéristique. Une revue détaillée des recherches antérieures sur la classification automatique des genres est présentée, incluant un examen technique des méthodes automatiques générales de sélection et de classification de caractéristiques. On discute également des questions théoriques concernant le genre de musique, incluant de manière non limitative les mécanismes utilisés par les humains pour classifier la musique et comment des taxonomies réalistes de genre peuvent être élaborées.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Project overview

The primary goal of this project was the production of an effective and easy to use software system that could automatically classify MIDI recordings by genre after having been programmed with a given genre hierarchy and trained on sample recordings. Before this could be accomplished, of course, there were a number of intermediate tasks to complete, each with varying degrees of research value of their own.

The first task was to study and consider musical genre from theoretical and psychological perspectives in order to achieve a broader understanding of the issues involved. This was useful in gaining insights on how to implement the classification taxonomy and in understanding what kinds of assumptions might be reasonable to make and what kinds should be avoided. The results of this study are presented in Chapter 2.

The next step was to review recent research in musical genre classification in order to incorporate previous work into this project and to see how it could be built upon. This information is presented in Chapter 4. It was also important to build a solid technical background by reviewing pertinent information relating to MIDI, feature selection techniques and classification techniques. These topics are covered in Chapter 3.

The next task was the compilation of a library of features, or pieces of information that can be extracted from music and used to describe or classify it. Features relating to instrumentation, texture, dynamics, rhythm, melodic gestures and harmonic content can all be used by humans to make distinctions between genres. Features based on these parameters were considered along with features that might not be obvious to humans, but could be useful to a computer. In order to complete the feature library, a literature search of publications on music theory, musicology, data mining and music technology was performed in order to find existing features. These features were then combined with a large number of original features in order to complete the library. All of this is presented in Chapter 5.

A model genre hierarchy was then constructed and a large set of MIDI files were collected in order to train and test the system. Although the large number of genres in existence made it impossible to consider every possible genre, efforts were made to incorporate as many different ones as possible, including genres from classical, jazz and popular music. Each feature from the feature library was then extracted and stored for each MIDI file. A variety of classification methodologies, based on statistical pattern recognition and machine learning, were then applied to this data and a system was built for coordinating the classifiers and improving their collective performance. Feature

selection was performed using genetic algorithms. The details of how all of this was done are presented in Chapter 6, along with a brief outline of the advantages of the design structure of the software and its easy to use interface.

Finally, a number of classification tests were performed in order to evaluate the system and judge its performance along a number of dimensions. Chapter 7 explains the tests and presents the results. Chapter 8 summarizes the results, compares the system's performance to that of existing systems, discusses the meaning of the results, outlines the original research contributions of this thesis and presents some areas for future research.

1.2 Importance of project and applications

Genre is used by music retailers, music libraries and people in general as a primary means of organizing music. Anyone who has attempted to search through the discount bins at a music store will have experienced the frustration of searching through music that is not sorted by genre. There is no doubt that genre is one of the most important means available of classifying and organizing music. Listeners use genres to find music that they're looking for or to get a rough idea of whether they're likely to like a piece of music before hearing it. Industry, in contrast, uses genre as a key way of defining and targeting different markets. The importance of genre in the mind of listeners is exemplified by research showing that the style in which a piece is performed can influence listeners' liking for the piece more than the piece itself (North & Hargreaves 1997).

Unfortunately, consistent musical genre identification is a difficult task, both for humans and for computers. There is often no generally accepted agreement on what the precise characteristics are of a particular genre and there is often not even a clear consensus on precisely which genre categories should be used and how different categories are related to one another. The problems of determining which musical features to consider for classification and determining how to classify feature sets into particular genres make the automatic classification of music a difficult and interesting problem.

The need for an effective automatic means of classifying music is becoming increasingly pressing as the number of recordings available continues to increase at a rapid rate. It is estimated that 2000 CDs a month are released in Western countries alone (Pachet & Cazaly 2000). Software capable of performing automatic classifications would be particularly useful to the administrators of the rapidly growing networked music archives, as their success is very much linked to the ease with which users can search for types of music on their sites. These sites currently rely on manual genre classifications, a methodology that is slow and unwieldy. An additional problem with manual classification is that different people classify genres differently, leading to many inconsistencies, even within a single database of recordings.

Research into automatic genre classification has the side benefit that it can potentially contribute to the theoretical understanding of how humans construct musical genres and the mechanisms they use to classify music. The mechanisms used in human genre classification are poorly understood, and constructing an automatic classifier to perform this task could produce valuable insights.

The types of features developed for a classification system could be adapted for other types of analyses by musicologists and music theorists. Taken in conjunction with genre classification results, the features could also provide valuable insights into the particular attributes of different genres and what characteristics are important in different cases.

The feature extraction and supervised learning classification techniques developed for a genre classifier have the important benefit of being adaptable to a variety of other content-based musical analysis and classification tasks. Systems could be constructed that, to give just a few examples, compare or classify pieces based on compositional or performance style, group music based on geographical / cultural origin or historical period, search for unknown music that a user might like based on examples of what he or she is known to like already, sort music based on perception of mood, or classify music based on when a user might want to listen to it (e.g. while driving, while eating dinner, etc.). Music librarians and database administrators could use these systems to classify recordings along whatever lines they wished. Individual users could use such systems to sort their music collections automatically as they grow and automatically generate play lists with certain themes. It would also be possible for them to upload their own classification parameters to search on-line databases equipped with the same classification software.

1.3 Defining musical genre and distinguishing it from style

Writers often fail to clearly define the differences between musical genre and style. This is understandable, to a certain extent, as there are definite similarities between the terms, but it is nonetheless important to distinguish between them if one is to undertake a detailed study of genre.

Franco Fabbri defines musical genre as “a kind of music, as it is acknowledged by a community for any reason or purpose or criteria, i.e., a set of musical events whose course is governed by rules (of any kind) accepted by a community” (Fabbri 1999). Fabbri continues on to define musical style as: “a recurring arrangement of features in musical events which is typical of an individual (composer, performer), a group of musicians, a genre, a place, a period of time” (Fabbri 1999). Musical genre can thus be considered to be somewhat broader and more subjective than style from a content-based perspective, which makes genre classification both more difficult and more interesting than style classification.

A possible clarification that may be made between genre and style is to say that a style is related to individuals or groups of people involved in music production and that genre is related to groups of music and the audiences that identify with these groups. It might therefore be said that a composer's style will remain evident even if she or he writes in different genres. In general terms, the word "genre" can be taken to refer to music that is, by general social agreement, grouped together.

It should be mentioned that the distinctions made above are for the purpose of clarity in this thesis, and that they do not reflect universal agreement in the music community. Moore (2001), for example, has a point of view that is not entirely consistent with Fabri's. Although much of the music technology literature on classification tends to use the words "genre" and "style" interchangeably, the term "genre" will be used exclusively here, in the sense defined above, unless the work of an authour who used the term "style" is being discussed.

1.4 Rationale for using MIDI

Musical data is generally stored digitally as either audio data (e.g. wav, aiff or MP3) or symbolic data (e.g. MIDI, GUIDO, MusicXML or Humdrum). Audio data represents actual sound signals by encoding analog waves as digital samples. Symbolic data, in contrast, stores musical events and parameters themselves rather than actual waves. Symbolic data is therefore referred to as a "high-level" representation and audio data as a "low-level" representation. In general, symbolic representations store high-level musical information such as individual note pitches and durations.

Audio data and symbolic data each have their respective strengths and weaknesses. It was decided to use a symbolic format, namely MIDI, rather than audio data for this thesis. A brief discussion of the rationale for this decision is included in this section, as it may be a somewhat controversial decision to some.

Before doing so, however, it is appropriate to define two terms for the purpose of clarity, as there is some debate as to exactly what they mean. Throughout this text, references are made to "high-level features" and "low-level features." For the purposes of this thesis, high-level features refer to pieces of information that are based on musical abstractions and low-level features refer to signal-processing characteristics derived directly from signals that do not have explicit musical meaning. For example, tempo, meter and key are all high-level features. Number of zero crossings and spectral frequency ratios are examples of low-level features. Although both high-level and low-level features can be extracted from low-level (audio) recordings, a much wider range of high-level features can be extracted with a much greater accuracy from high-level (symbolic) recordings.

There is no doubt that there are a number of good reasons for using audio data rather than symbolic data for performing genre classifications. Most obviously, audio data is what people actually listen to in general, so an audio classification system has more apparent practical use than a MIDI classification system. Furthermore, MIDI files fail to store certain information that could be useful for performing genre classifications. For example, MIDI files store references to standardized General MIDI synthesizer patches rather than actual sounds (see Chapter 3 for more details on MIDI), with the result that significant amounts of potentially very important timbral information are unavailable. The ability to extract features related to information such as quality of singing voice, lyrics, phrasing and expression can be eliminated or become severely hampered. This is potentially a very serious problem, as there is some evidence that timbral features may be more significant than rhythmic or pitch-based features (Tzanetakis & Cook 2002). An additional problem is that MIDI was devised primarily with Western musical models in mind, which limits its usefulness beyond this paradigm. A limitation of symbolic representations is that any information not encapsulated in the representation is lost. Although MIDI does allow pitch mico-intervals and arbitrary rhythms, there are still some limitations in this respect, such as the limited patches specified in General MIDI.

It is likely for these reasons that most genre classification systems to date have focused almost exclusively on audio data (see Chapter 4). It is recognized here that audio classification is certainly a more important goal from a practical perspective than symbolic data classification. Unfortunately, success with audio classification has not yet attained a level that is commercially viable. Any possible improvements to these audio systems should therefore be considered. The current limitations of polyphonic transcription systems make it very difficult or impossible to extract accurate features based on high-level musical knowledge, such as precise note timing, voice and pitch, an important limitation considering that these features could very likely improve success rates significantly. Since these features are readily available in symbolic data, it was decided to study symbolic data in this project in order to complement and build on the work that has already been done with audio data. Although audio genre classification is of course very useful, there exists a large body of transcribed music, and it would seem foolish to ignore this information by only considering audio data.

This research is intended, in part, to show the usefulness of high-level features, with the hopes that they could eventually be integrated into an audio classification system when transcription systems become more effective or file formats that package audio data with meta-data transcriptions come into common use. High-level features have the additional advantage that they have musicological meaning, and can thus be used for theoretical studies as well as more applied applications.

As an additional note, the loss of timbral data beyond instrumentation may not be as significant as it may seem at first. A recent study of the relationship between timbre features and genre indicated that there may be a poor correlation between the two (Aucouturier & Pachet 2003). It is, of course, true that timbre certainly does play at least some roll in how humans perform classifications, particular in regards to vocal quality, but this is still an indication that the importance of timbre may have been overemphasized in past research. The fact that timbre-based features played a sometimes overwhelming role in many of the existing classification systems may have impaired their performance.

MIDI data also makes it easier to extract features relating to entire recordings, both because of the symbolic nature of the data and because of the speed with which it can be processed. In contrast, it is often necessary to derive features from segments of audio data rather than entire recordings.

An additional benefit of using symbolic data is that it makes it possible to classify music for which scores are available but audio recordings are not. Classification based on high-level features is particularly ideal for music for which no audio recordings exist, as new performances could introduce biases. Advances in optical music recognition technology could make it a simple matter to produce MIDI files that could be classified by scanning in scores.

MIDI was chosen as the particular symbolic format to use in this thesis because it is the most prevalent format and it is relatively easy to find diverse MIDI files for training and testing. In addition, it is relatively easy to translate other symbolic representations into MIDI if one wishes to classify recordings in other formats.

1.5 Rationale for using supervised machine learning

There are three main classification paradigms that could have been used in this thesis:

- **Expert Systems:** These systems use pre-defined rules to process features and arrive at classifications.
- **Supervised Learning:** Theses systems attempt to formulate their own classification rules by using machine learning techniques to train on model examples. Previously unseen examples can then be classified into one of the model categories using the rules generated during training.
- **Unsupervised Learning:** These systems cluster the data that they are fed based on similarities that they perceive themselves rather than model categories.

Expert systems are a tempting choice because known rules and characteristics of genres can be implemented directly. A great deal of potentially useful work has been done analyzing and generating theoretical frameworks in regards to classical music, for example. Given this body of research, it might well be feasible to construct a rules-based

expert system to classify these kinds of music. There are, however, many other kinds of music for which this theoretical background does not exist. Many types of Western folk music, a great deal of non-Western music and Western popular music do not, in general, have the body of analytical literature that would be necessary to build an expert system.

There have, of course, been some efforts to generate general theoretical frameworks for popular and/or non-Western music, such as in the work of Middleton (1990). Unfortunately, these studies have not been precise or exhaustive enough to be applicable to the task at hand, and it is a matter of debate as to whether it is even possible to generate a framework that could be broad enough to encompass every possible genre of music. Although there are broad rules and guidelines that can be informally expressed about particular genres (e.g. delta blues music often has a swing rhythm, is likely to follow the 12-bar blues form, will often be limited to guitar and voice and will likely make use of the pentatonic scale), it would be very difficult to design an expert system that could process rules that are often ill-defined and inconsistent across genres. A further problem is that new popular and folk genres are constantly appearing and existing ones often change. Keeping a rules-based system up to date would be a very difficult task.

So, although expert systems could potentially be applied to well-defined and unchanging types of music, such as pre-twentieth century classical music, any effort to perform classifications across a wider range of genres with rules-based systems is likely doomed to failure. Anyone trying to implement an expert system with applicability to a reasonable variety of genres would quickly get bogged down in inconsistencies and details, and would have to make judgment calls that would bias and date the system.

Systems that rely on pattern recognition and learning techniques, in contrast, hold a great deal of potential. Such systems could analyze musical examples and attempt to learn and recognize patterns and characteristics of genres in much the same way that humans do, although the precise mechanisms would differ. A side benefit of such systems is that they may recognize patterns that have not as of yet occurred to human researchers. These patterns could then be incorporated into theoretical research.

This leaves the options of supervised and unsupervised learning. It was decided that unsupervised learning would be inappropriate for this project, since the categories produced might not be meaningful to humans. Although this would avoid the problems related to defining a set genre hierarchy (see Chapter 2) and the categories produced might well be more accurate than human genre categories in terms of “objective” similarity, a genre classification system that uses its own genre categories would be useless for humans who want to use genres that are meaningful and familiar to them. Systems that use unsupervised learning to measure the similarity of recordings certainly do have their uses, but they are not well-suited to the specific problem of genre classification.

Supervised learning is therefore the best option, despite the fact that the need for a manually classified and therefore biased model training set is a drawback. Systems of this type form their own rules without needing to interact with humans, meaning that the lack of clear genre definitions is not a problem. These systems can also easily be retrained to reflect changes in the genres being classified. Furthermore, these systems are able to consider and form relations between large groups of features, a task that is difficult to encode into an expert system.

2. Musical Genre Theory

2.1 Introduction to musical genre theory

Although a great deal of thought has been put into the notion of genre by those working in the field of literary theory (the collections of works edited by Duff (2000) and Grant (2003) are excellent resources on literary genre and film genre respectively), there has been relatively little research done specifically on musical genre. What work has been done leans towards either lists of binary features that tend to be too inflexible for the purposes of an evolving genre structure or discussions centered around the socio-cultural aspects of genres that are of limited utility to content-based classification.

This topic is so large and there is so much research remaining to be done that a theoretical study of musical genre could easily fill several doctoral dissertations. Although this is somewhat beyond the scope of this thesis, an overview of the issues involved is presented here, as it is important that one be aware of the theoretical aspects of musical genre in order to properly consider their practical implications. This chapter discusses this important background relating to genre and genre classification. The details of the actual genre categories that were implemented in this project can be found in Section 6.1.

Genre categories allow humans to group music along lines of perceived similarity. It is commonly held to be true by cognitive psychologists that categorization in general allows us to comes to terms with information by grouping it in meaningful ways that can increase our understanding of it. Although musical genre is sometimes referred to as being primarily a marketing tool or an artificial labelling system used by academics, one should not lose sight of its deeper significance.

There are a number of important issues to consider relating to genre. How genres are created, how they are agreed upon and disseminated, how they are defined, how they are perceived and identified, how they change, how they are interrelated and how we make use of them are all important areas of research.

Answering these questions is not an easy task. It can be difficult to find clear, consistent and objective definitions of genres, and genres are rarely organized in a consistent or rational manner. The differences between genres are vague at times, rules distinguishing genres are often ambiguous or inconsistent, classification judgments are subjective and genres can change with time. The categories that come to be are a result of complex interactions of cultural factors, marketing strategies, historical conventions, choices made by music librarians, critics and retailers and the interactions of groups of musicians and composers.

2.2 How humans deal with genre

Very little psychological experimental research has been done on the mechanisms that humans use to define and identify musical genres. This makes it somewhat difficult to build a content-based system that can be said to be reliably modeled on the classification techniques used by humans. There is, however, some research that has been done on genre in general that can be used to supplement what research on musical genre there is.

Research by cognitive psychologist Eleanor Rosch has indicated that people tend to think of categories as having some typical, or prototypical, members, and other less typical members. For example, a robin can be considered to be a better example of a bird than an ostrich, or a chair a better example of furniture than a magazine rack (Rosch 1975; Taylor 1989). This certainly seems consistent with the way that some recordings seem typical of a musical genre, yet others seem less so, while still leaving little doubt as to their membership in that particular genre. Marie-Laure Ryan puts these ideas into the context of genre in general:

> This approach invites us to think of genres as clubs imposing a certain number of conditions for membership, but tolerating as quasi-members those individuals who can fulfill only some of the requirements, and who do not seem to fit into any other club. As these quasi-members become more numerous, the conditions for admission may be modified, so that they, too, will become full members. Once admitted to the club, however, a member remains a member, even if he cannot satisfy the new rules of admission. (Ryan 1981, 118)

This highlights some of the problematic inconsistencies in genres from a content-based perspective. It might further be added that quasi-members have membership due to similarities with the prototypical genres, although they might otherwise be dissimilar to each other. It might also be said that some members can belong to more than one genre and that the accumulation of enough quasi-members with a close similarity to each other may lead to the creation of a new genre or sub-genre. As stated by Simon Frith, “genres ‘fail’ when their rules and rituals come to seem silly and restrictive; new genres are born as the transgressions become systematic” (Frith 1996, 94).

Research has shown that individuals are more familiar with sub-genres within a musical genre that they like than within genres that they do not (Hargreaves & North 1999). Even well-trained musicians may have significantly less knowledge and understanding of genres that they are not proficient in than even casual listeners who are interested in those genres. Furthermore, the features that different individuals use to identify genres can vary based on the genres that they are familiar with. All of this implies that few, if any, humans can be relied upon to classify arbitrary genres reliably and using consistent mechanisms. A computer system that can become familiar with a much larger range of music than most humans would be willing or able to may well be

able to perform genre classifications in general better than humans, even though individual humans may still perform better in their own limited domains.

Turning now to the features that humans use to perform classifications, one might imagine that high-level musical structure and form play an important role, given that this is the area on which much of the theoretical literature has concentrated. This does not appear to be the case, however. Research by Perrott and Gjerdingen (1999) found that humans with little to moderate musical training are able to make genre classifications agreeing with those of record companies 71.7% of the time (among a total of 10 genres), based on only 300 milliseconds of audio. This is far too little time to perceive musical form or structure. This suggests that large-scale structural elements of music are in fact not needed by humans in order to make genre classifications and that there must therefore be a sufficient number of features available in very short segments of sound to successfully perform classifications. This does not mean that one should ignore musical form and structure, as these are likely useful as well, but it does mean that they are not strictly necessary. This is important if one wishes to build an audio classification system, as it takes much less computational time to process short segments of sound than to process long segments.

Another study found that, when asked to list the characteristics of a number of different genres, a test group consisting of Turkish undergraduates used the following terms most frequently: lively, exuberant, dynamic, exciting, entertaining, rhythmic, meaningful, pleasant, high quality, boring, irritating, simple, lasting, monotonous, harmonious, sentimental, restful, peaceful and soothing (Tekman & Horascsu 2002). It appears that even a relatively sophisticated sample tends to prefer words having to do with personal preferences and mood rather than direct musical descriptors. Although the descriptors that they used may well have been influenced by more detailed musical features, consciously or unconsciously, this research does appear to indicate that how individuals classify music is strongly related to the emotive meaning that is perceived.

The authour has been unable to find any other experimental research beyond these studies relating to musical genre, unfortunately. Given the lack of experimental evidence, one may only speculate on further ways that humans identify genre. It would be valuable for future psychological experiments to be carried out to investigate this speculation.

It seems intuitively likely that different individuals use different methods for distinguishing between genres, based on such factors as musical training and the genres that one is the most familiar with. Songwriters, musicians, critics, teachers, promoters and listeners in general can all be seen to describe music in different ways, and may perceive and think about it differently as well. The same individual may also use different techniques when identifying different types of music.

There are two likely mechanisms that we use for performing classifications of music. The first is to compare features that we perceive, such as rhythmic patterns or the particular instruments being played, with our existing knowledge, conscious or unconscious, of the characteristics of different genres. The second is to perform classifications by measuring the similarity of unknown pieces with pieces that are known to belong to certain categories. Although these two approaches are related, the inherent processes at work are somewhat different, and those with musical training are probably more likely to use the first approach more. The importance of the second approach is demonstrated by the apparent fact that, when asked to describe a piece of music, many people will do so by linking it to other pieces of music that are in some ways similar. This seems to indicate that exemplar-based classification is an important tool used by humans to group music into genres and identify them.

It would seem reasonable to say that lyrics play a particularly important role for many people in identifying genres, as everyday experience seems to indicate that they are the most memorable aspect of recordings to many people with limited musical training. Content (e.g. love, political messages, etc.), rhyming scheme, vocabulary, use of clichéd phrases and use of characteristic slang all likely provide useful indications of genre. Style of singing and voice quality are also likely quite important, as these do seem to vary significantly from genre to genre.

It is probable that many people use features beyond those that can be derived from the actual musical content of a recording or a performance. Genre is very much linked to the social, economic and cultural background of both musicians and listeners. Both of these groups tend to identify with and associate themselves with certain genres, with the result that their behaviour is influenced by their preferences. Or, viewed from a different perspective, many people have strong identifications with social and/or cultural groups that are associated with certain musical genres. In either case, it appears that there is often a correlation between musical genre preferences and appearance and behaviour. One need only see a photo or watch an interview with a musician, without ever having heard his or her music, to be almost certain whether the musician plays rap, heavy metal or classical music, for example.

The style of album art, web pages and music videos are all features that humans can use to identify genre. Similarly, a performer's appearance and actions on stage (facial expressions, ritual gestures, types of dancing, etc.) provide clues towards genre, as do an audience's demographics, dress and behaviour (clapping, shouting, sitting quietly, dancing, etc.). The fine distinction between some sub-genres may well be related to such sociological features more than musical content. Although the current study is only concerned with content-based features, future research that uses data mining techniques to gather these other types of features to supplement content-based features could be highly

useful. There has been some initial research in this direction (Whitman & Smaragdis 2002) that has had very encouraging results.

For further information about some of the issues raised here, one may wish to consult the work of Franco Fabbri, who is responsible for writing some of the most widely referenced work regarding musical genres from a musicological perspective. Fabbri discusses the links between genre and social, economic and cultural factors and how genres come into being in one of his early papers (Fabbri 1981). Fabbri continues this discussion in a slightly later paper (Fabbri 1982). He also presents a discussion of the issues related to musical categories, how the mind processes them and their importance in general in a more recent paper (Fabbri 1999).

Chapter 4 of a Frith's book (1996) is also informative, and contains an excellent discussion of the types of socio-cultural factors that can affect how genre distinctions are formulated and what their meaning is. Toynbee (2000) provides an interesting discussion of how genres inform musicians and of the influences of identifications with different communities as well as of the music industry.

David Brackett has also done some very interesting work on musical genre, including a discussion of how the ways in which particular genres are constructed and grouped can vary in various charts, radio formats and media-fan groups, and of issues relating to recordings crossing over from one set of groupings to another (Brackett 2002). Brackett has also written a good resource for those trying to deal with the task of characterizing genres (Brackett 1995).

2.3 Finding an appropriate labelling system

In order to train an automatic classification system using supervised learning it is first necessary to have a set of genre categories that the training examples can be partitioned into. The mechanisms in which humans label entities or concepts in general is in itself an interesting area of inquiry. Lakoff's book (1987) is an often cited source on this topic.

The lack of a commonly accepted set of clearly defined genres makes it tempting to simply devise one's own artificial labels for the purposes of making an automatic classification system. These labels could be designed using reasonable, independent and consistent categories, a logical structure and objective similarity measures. One could even use unsupervised learning techniques to help accomplish this. The genre labels in common use are often haphazard, inconsistent and illogical, and someone designing an automatic classifier would certainly like to have a system that does not suffer from these problems.

This would, of course, be a mistake. One must use the labels that are meaningful to real people in order for the labels to be useful to them, which is to say that genre categories must be consistent with how a person with moderate musical knowledge would

perform categorizations. Furthermore, genre labels are constantly being created, forgotten and modified by musicians, retailers, music executives, DJs, VJs, critics and audiences as musics develop, so a static, ideal system is not sustainable. Genre is not defined using strictly objective and unchanging qualities, but is rather the result of dynamic cultural processes. One must therefore be careful to avoid thinking of genres in terms of immutable snapshots, as both their membership and their definitions change with time.

The genre labels attached to a particular recordings can change, even though the recording itself, of course, remains static. What might now be called "rock 'n roll," for example, was once classified as "novelty," and the Bob Marley recordings that we recognise as "reggae" today were once classified as "folk music." The changes in genre definitions that can also occur as well are illustrated by the differences between what was considered "rock" music in every decade from the 1950's to now. In order to have true practical usefulness, a system of labels should be used that is entirely up to date, as it should coincide with the terms used by real people.

Historical types of music should also be included in a full taxonomy, of course, and it should not be forgotten that even these can change with time. Of course, it can be argued that historical genres tend to be more static and codified than current genres, with the result that they these genres are easier to label and describe, and that membership is fairly set. There is, for example, a large amount of literature on Baroque music theory and practice, and there is not any significant quantity of new Baroque-style pieces being composed that might cause this genre to mutate. Although this point does have some validity, historical genres can nonetheless still evolve to some degree, as can be demonstrated by a comparison of how Baroque music was performed by early 20th century musicians and by more recent ensembles using period instruments.

Syncretic music, which is to say music that combines characteristics of multiple genres, presents a further problem. Although syncretic music can sometimes lead to the creation of new sub-genres, there is at least a transitional stage where such music does not definitively fit into one genre rather than another. This creates a definite difficulty to one wishing to design a clear labelling system with discrete categories.

Considering the quantity and diversity of people that would need to be convinced to use an artificial labelling system and considering the rate at which genre taxonomies change, it is clear that any attempt to impose a given set of labels on the public is doomed to failure. The Canadian Content radio genre categories used by the Canadian government are an example of such a failure. These categories are generally inadequate and were obsolete before they were even brought into being. More information on the Canadian Content radio genres can be found in Frith's book (1996).

Another approach to finding an appropriate labelling structure is to look at the categories used by music sales charts such as Billboard, or by awards shows such as the

Grammies. Unfortunately, there are also a number of problems with this approach. Charts such as those used by Billboard often only reflect the current trends in music to the exclusion of older genres. A proper system should include old genres as well as new. Furthermore, these categories tend to reflect the labelling system that the music industry would ideally like to see for commercial reasons, not the one which is actually used by the public. Charts and award categories therefore often have labels based on marketing schemes more than common perceptions, and do not even offer the advantages of being consistent or well thought out from a taxonomical perspective. There are a number of interesting publications, such as that by Negus (1999), that offer a further analysis of the effects of business interests on musical genres and their development.

Specialty shows on radio or television do offer a somewhat better source of labels, as they often reflect categories that attract listeners interested in specific genres, both new and old. They do still suffer from the influence of commercial biases, however, as the content of shows tend to be influenced at least as much by the interests of advertisers relating to age, income and political demographics as by the musical preferences of listeners. Although university radio stations do not suffer from this problem in the same way, they are often limited in scope and by the variable expertise and knowledge of their DJs.

Retailers, particularly on the internet, may perhaps be the best source of labels. They use categories that are likely the closest to those used by most people, as their main goal is to use a taxonomy that makes it easy for customers to find music that they are looking for. Although retailers can sometimes be a little slow to respond to changes in genre, they nonetheless do respond faster than some of the alternatives discussed above, as responding to new genres and keeping existing genres up to date allows them to draw potential buyers into areas that contain other music that they may wish to buy, therefore increasing sales.

Although one might argue that it would be preferable to base labels on the views of concert goers, clubbers, musicians, DJs, VJs, music reporters and others who are on the front lines of genre development, doing so would be disadvantageous in that genres at this stage of development may be unstable. Additionally, favouring the genre labels used by specialists may result in some confusion for non-specialists. Waiting for retailers to recognize a genre and thus make it “official” is perhaps a good compromise in that one keeps somewhat abreast of new developments, while at the same time avoiding excessive specialization and excess overhead in terms of data collection and training.

The problem of inconsistency remains, unfortunately, even with the taxonomies used by retailers. Not only do record companies, distributors and retailers use different labelling systems, but the categories and classification judgements between different retailers can also be inconsistent. This is, unfortunately, an unavoidable problem, as there

are no widely accepted labelling standards or classification criteria. Employees of different organizations may not only classify the same recording differently, but may also make selections from entirely different genre taxonomies or emphasize different identifying features. One must simply accept that it is impossible to find a perfect taxonomy, and one must make do with what is available. One therefore has little choice but to adopt one of the imperfect labelling systems that are in use, and those used by retailers appear to be the best choice from a practical perspective.

2.4 Particularly problematic categories

One of the greatest problems with designing a content-based automatic classification system is dealing with genres that are only distinguishable based on patterns of social use and context rather than musical content. The different musics of the Nariño, for example, are almost identical in terms of content, but are considered to belong to entirely different genres based on the social context in which they are performed (Broere 1983). Although this is an extreme example, it is not unique, and socio-cultural features external to actual musical content play an important role in many genre categories.

There are a number of common Western categories that group together music that is dissimilar from a content-based perspective. “Top 40” music, “woman’s music” and “indie music” are all examples of categories into which music from entirely unrelated (from a content-based perspective) genres can be grouped together.

“World music” is also a problematic category to deal with, as it groups together many sub-genres that are wholly dissimilar in terms of content. Even if one includes socio-cultural features, many of the sub-genres found in record stores often only have socio-cultural commonality from the uninformed perspective of Western music industry workers. Even the criterion of coming from a non-Western country is not decisive, as there are types of Western folk music that are labelled as world music and there are types of music originating from non-Western countries that are played in Western styles and therefore do not fall under the world music umbrella. For example, the music of a Nigerian cellist playing Bach should be classified under Baroque, not under a Nigerian sub-genre of world music.

One possible solution to these problems would be to consider the language that is being sung in as an important feature. This would require either language recognition software or appropriate meta-data, neither of which are easily and consistently available. A second possibility would be to only perform classification of sub-genres, and to obtain the unrealistically broad parent genres by implication. The disadvantage of this is that one loses the potentially improved classification accuracy that could potentially be obtained by classifying at multiple levels in a genre tree at once, and using a weighted average or some other form of co-ordination to obtain a final result. A final possibility would be to

consider the demographics of the listeners and musicians as features. Although the extraction of such features is beyond the scope of this paper, it may well be the only apparent way to deal with categories that are objectively dissimilar but commonly used.

2.5 Interrelations between categories

An important part of constructing a genre taxonomy is determining how different categories are interrelated. This is, unfortunately, a far from trivial problem. Attempts to this point to implement an automatic classification system have sidestepped these issues by limiting their testing to only a few simple genres. Although this is acceptable in the early stages of development, the problem of taxonomical structures needs to be carefully considered if one wishes to construct a system that is scalable to real-world applications.

This problem is discussed in a paper by Pachet and Cazaly (2000). The authours observe that retailers tend to use a four-level hierarchy: global music categories (e.g. classical, jazz, rock), sub-categories (e.g. operas, Dixieland, heavy metal), artists and albums. Although this taxonomy is effective when navigating a physical record store, the authours argue that this taxonomy is inappropriate from the viewpoint of establishing a major musical database, since different levels represent different dimensions. In other words, a genre like “classical” is fundamentally different from the name of an artist.

Pachet and Cazaly continue on to note that internet companies, such as Amazon.com, tend to build tree-like classification systems, with broad categories near the root level and specialized categories at the leaves. The authours argue that, although this is not in itself necessarily a bad approach, there are some problems with it. To begin with, the level that a category appears at in the hierarchy can vary from taxonomy to taxonomy. Reggae, for example, is sometimes treated as root-level genre and is sometimes considered a sub-genre of world music.

A further problem is that there is a lack of consistency in the type of relation between a parent and a child. Sometimes it is genealogical (e.g. rock -> hard rock), sometimes it is geographical (e.g. Africa -> Algeria), sometimes it is based on historical periods (e.g. Baroque -> Baroque Opera), etc. Although these inconsistencies are not significant for people manually browsing through catalogues, they could be problematic for automatic classification systems that are attempting to define genres using content-based features, as different musics from the same country or same historical period can be very different musically.

Julie E. Cumming has adapted Ludwig Wittgenstein’s ideas about the “family resemblance” between genres to music (Cumming 1999), and uses this theoretical basis as justification for favouring an exclusively genealogical organization of categories. She argues that, since lists of simple and well-defined binary features are insufficient to distinguish between sometimes amorphous genres, it would be wise to consider genres in

terms of the similarities that they share with the features of genre families that they have descended from.

An additional problem to consider is that different tracks in an album or even different albums by an artist could belong to different genres. Many musicians, such as Neil Young and Miles Davis, write music in different genres throughout their careers. It seems clear that attempting to classify by musicians rather than individual recordings is problematic.

Pachet and Cazaly argue that it therefore seems apparent that, ignoring potential problems related to size, it would be preferable to base taxonomies on individual recordings, rather than on artists or albums. In a later paper, however, Aucouturier and Pachet (2003) argue that one should in fact use taxonomies based on artist rather than title, as taxonomies based on title involve many more entries and result in categories that are overly narrow and have contrived boundaries.

Pachet and Cazaly argue that it is necessary to build an entirely new taxonomy to meet the needs of any large scale musical database. They emphasize the goals of producing a taxonomy that is objective, consistent, independent from other metadata descriptors and that supports searches by similarity. They suggest the use of a tree-based system organized based on genealogical relationships, where only leaves would contain musical examples. Each node would contain its parent genre and the differences between its own genre and that of its parent.

The concerns with existing taxonomies expressed by Pachet and Cazaly are certainly valid, but their proposed solution unfortunately has some problems of its own. To begin with, defining an objective classification system is much easier said than done, and getting universal agreement on a standardized taxonomy is most probably an impossible task. Furthermore, their system does not deal with the reality that a single recording can sometimes reasonably be said to belong to more than one genre, nor does it deal with the potential problem of multiple genealogical parents that can compromise the tree structure.

It seems apparent that some modifications are needed to Pachet and Cazaly's system, but some sort of hierarchal tree-based taxonomy nonetheless appears to be a convenient and realistic genre structure. Franco Fabbri (1982) suggests that, when faced with describing a genre to a person who is unfamiliar with it, most individuals do so by defining the genre as an intersection of other similar genres with labels known to both parties, by using a broader label under which the genre in question might fall or by explaining the genre using familiar terms such as definitions and emotive meanings. The former two methodologies are certainly consistent with a hierarchal structure with visible parents and siblings.

A further issue to consider is the variable degree to which different genres branch out into sub-genres. Considered from a hierarchal tree-based perspective, this variability applies to both the depth and breadth of various branches. Some genres have many very

specialized sub-genres, such as electronic dance music (e.g. techno, jungle, rave, etc.). Others, such as pop-rock, tend to have fewer, broader and less specified sub-genres. For the purposes of creating a genre hierarchy, one must accept these inconsistencies rather than imposing unrealistically broad or narrow categories in order to avoid dissymmetry in the genre structure.

2.6 Towards a successful large-scale taxonomy

Aucouturier and Pachet (2003) divide methods of genre classification into three categories: manual, prescriptive and emergent. The manual approach involves humans performing the classification task by hand, while the prescriptive and emergent approaches involve automatic systems.

Aucouturier and Pachet define the prescriptive approach as an automatic process that involves a two-step procedure: feature extraction followed by machine learning / classification. The prescriptive approach assumes a pre-existing taxonomy that a system can learn. Aucouturier and Pachet argue, reasonably enough, that prescriptive systems tend to be based on contrived taxonomies and that a truly useful system would need to be able to deal with much larger taxonomies than can successfully be modelled and kept up to date. A further problem is that it can be difficult to find training samples that are unambiguously representative enough to train a classifier properly.

Aucouturier and Pachet argue that the emergent approach as the best alternative. Rather than using existing taxonomies, an emergent system attempts to emerge labels according to some measure of similarity. The authors suggest using similarity measurements based on audio signals as well as on cultural similarity gleaned from the application of data mining techniques to text documents. They propose the use of collaborative filtering to search for similarities in the taste profiles of different individuals and of co-occurrence analysis on the play lists of radio programs and the track listings of CD compilation albums.

The emergent approach is untested, however, and it is difficult to predict how effective it would be in real life. Implementing the data mining techniques that would be required would be quite a difficult task. Furthermore, there is no guarantee that the recordings that get clustered together would be consistent with groupings that humans use in reality or would find convenient to use, nor is there any obvious provision for defining the types of genre structures and interrelations that humans find useful when browsing through categories. Nonetheless, the emergent approach holds more promise than naïve unsupervised learning, and is certainly worthy of investigation.

In any event, such a system is clearly beyond the scope of this thesis. The use of a reasonable model taxonomy, while not ideal, is certainly effective enough for most practical use and for the purposes of this thesis. A logical future development would be to

merge the system developed here with modules that collect and consider non-content based socio-cultural data. Whether it is prescriptive or emergent systems that end up being more effective, the idea of automatically exploiting text documents to gather socio-cultural data is an interesting one, and should certainly be explored in future research.

3. Technical Background Information

3.1 MIDI

MIDI is an encoding system that is used to represent, transfer and store musical information. Instead of containing actual sound samples as audio encoding methods do, MIDI files store instructions that can be sent to synthesizers. The quality of sound produced when a MIDI file is played is therefore highly dependant on the synthesizer that the MIDI instructions are sent to. In effect, MIDI recordings give one much the same information that one would find in a musical score. MIDI, and other formats such as KERN, MusicXML or GUIDO, are often called “symbolic” formats because of this. Selfridge-Field (1997) provides a good overview of alternative symbolic formats to MIDI.

The MIDI standard is known to have a number of weaknesses and disadvantages. There is a relatively low theoretical threshold on the amount of control information that MIDI can encapsulate, for example. Furthermore, it can be difficult and time consuming to properly record sophisticated synthesis instructions. In effect, a basic MIDI recording of a human performance will almost always sound significantly worse than an audio recording, partly because it is impossible to properly record the full range of control parameters of many instruments and partly because of limitations in synthesizers.

MIDI recordings do, however, have a number of advantages over audio recordings. They are much more compact, which in turn makes them easier to store and much faster to process and analyze. MIDI recordings are also easier to edit, as they store simple instructions that are easy to view and change. This contrasts with audio recordings, where it is not currently possible (with the exception of simple monophonic music) to even correctly extract the actual notes being played.

MIDI is therefore much more convenient than audio if one wishes to extract precise high-level musical information. For the purposes of genre analysis, this is an important advantage, as it is desirable to look for patterns relating to notes, rhythms, chords and instrumentation, all of which are easy to extract from MIDI but currently difficult to impossible to extract from audio. More information on the advantages and disadvantages of MIDI in relation to genre classification can be found in Section 1.4.

Only the portions of the MIDI specification that are relevant to the task of genre classification are discussed in this section in any kind of detail. The aspects of MIDI that are relevant to live performance but not to MIDI files, for example, are almost entirely ignored here. There are many books on MIDI, such as that by Rothstein (1995), which can be consulted for further information on MIDI. The complete specification is

published by the MIDI Manufacturers Association (2001). The *MIDI Manufacturers Association* web site also provides additional useful information.

MIDI essentially consists of sequences of instructions called “MIDI messages.” Each MIDI message corresponds to an event or change in a control parameter. MIDI messages consist one or more bytes of data, which fall into two types: status bytes and data bytes. The status byte is always the first byte of a MIDI message, always starts with a 1 bit and specifies the type of MIDI message and the number of data bytes that will follow to complete the message. Data bytes always start with a 1 bit, which means that each data byte has 7 bits free to specify values, with a resultant range of between 0 and 127.

MIDI allows the use of up to sixteen different “channels” on which different types of messages can be sent. Each channel operates independently of the others for most purposes. Channels are numbered from 1 to 16. There is no channel 0.

There are two important classes of MIDI messages: “channel messages” and “system messages.” The former influence the behaviour of only a single channel and the latter affect the MIDI system as a whole. “Channel voice messages,” a type of channel message, are the only type of messages that are relevant to this thesis. The four least significant bits of the status byte of all channel voice messages indicate the channel number (0000 is channel 1 and 1111 is channel 16). “Note On,” “Note Off,” “Channel Pressure,” “Polyphonic Key Pressure,” “Program Change,” “Control Change” and “Pitch Bend,” as discussed below, are all channel voice messages.

A Note On messages instructs a synthesizer to begin playing a note. This note will continue playing until a Note Off message is received corresponding to it. The four most significant bits of the status byte of all Note On messages must be 1001 and, as with all channel voice messages, the four least significant bits specify the channel on which the note should be played. Note On messages have two data bytes. The first specifies pitch, from 0 to 127, and the second specifies velocity, also from 0 to 127. Pitch is numbered in semitone increments, with note 60 being designated as middle C (equal temperament tuning is used by default), although synthesizers can be instructed to use alternate arrangements and tunings. The velocity value specifies how hard a note is struck, which most synthesizers map to initial volume.

A Note Off message has an identical format to a Note On message, except that the four most significant bits of the status byte are 1000. The pitch value specifies the pitch of the note that is to be stopped on the given channel and the velocity value specifies how quickly a note is released, which is generally mapped to the fashion in which the note dies away. Many synthesizers do not implement Note Off velocities. A Note On message with velocity 0 is equivalent to a Note Off message for the given channel and pitch.

Channel Pressure messages specify the overall pressure for all notes being played on a given channel. This can be mapped by synthesizers in a variety of ways. Aftertouch

volume (loudness of a note while it is sounding) and vibrato are two common mappings. The status byte of Channel Pressure messages has 1101 as its four most significant bits, and there is one data byte that species the pressure (between 0 and 127).

Polyphonic Key Pressure messages are similar to Channel Pressure messages, except that they contain an additional byte (the one immediately following the status byte) that specifies pitch, thus restricting the effect of the message to single notes rather than to all notes on a channel. The most significant bits of the status byte are 1010.

Program Change messages allow one to specify the instrumental timbre that is to be used for all notes on the specified channel. The terms “program,” “patch” and “voice” are often used in reference to the instrumental timbres specified by Program Change messages. The most significant bits of the status byte are 1100 and there is a single data byte specifying patch number, from 0 to 127.

The particular 128 instrumental timbres and sound effects corresponding to particular patch numbers are specified by the MIDI Program Table, which is part of an addendum to the MIDI specification called General MIDI. All notes sent to a given channel will be played using the patch specified by the most recent Program Change message sent to that channel. There is one exception to this, however. All notes played on channel 10 are considered to be percussion notes, and General MIDI specifies a separate Percussion Key Map specifying 47 percussion timbres that are always used for notes sent to channel 10. The timbre that is used for notes on channel 10 is specified by the pitch value of Note Ons, not by Program Change messages. Rothstein (1995) gives the details of the MIDI Program Table and the General MIDI Percussion Map.

Control Change messages affect the sound of notes that are played on a specified channel. Common parameters include volume and modulation. There are 121 MIDI controllers, including some that are unspecified, although many synthesizers do not implement most, or even any, of these. The four most significant bits of the status byte of Control Change messages are 1011. The first data byte specifies the controller that is being referred to, from 0 to 120. There is a second data byte that specifies the setting of the controller, from 0 to 127.

If a greater resolution is required, a second Control Change message can be sent to supplement the first, resulting in a resolution of 16 384. Controllers 0 to 31 represent the most significant byte in this case, and controllers 32 to 63 represent the least significant byte. Two Control Change messages can thus cause a single change to be implemented with much greater resolution than a single Control Change message.

Control Change messages are generally intended for use when performing with continuous controllers. They are only standardized to a limited extent, and many synthesizers do not implement them. They are thus of limited applicability to this project, which analyzes MIDI files in the context of scores rather than performance nuances.

Control Change messages are often absent in MIDI files, and the lack of standardization could cause a good deal of noise. They are therefore only considered in a very limited capacity in this thesis. Rothstein (1995) gives the details of particular Control Change messages.

Pitch Bend messages allow microtonal synthesis. The four most significant bits of the status byte of such messages are 1110. There are two data bytes, the first of which specifies the least significant byte of the Pitch Bend and the second of which specifies the most significant byte. Maximum downward bend corresponds to data byte values of 0 followed by 0, centre pitch (no bend) corresponds to values of 0 followed by 64 and maximum upward bend corresponds to values of 127 followed by 127. General MIDI specifies that the default Pitch Bend range is plus or minus two semitones. This can be altered on synthesizers, however, so one must be careful to ensure that the Pitch Bend range that is actually played corresponds to what is desired.

MIDI timing is controlled by a clock which emits “ticks” at regular intervals. Clock rates are usually related to note durations in terms of parts per quarter note (ppqn). A greater ppqn corresponds to a greater rhythmic resolution, which allows one to have push or lag the beat or to represent complex tuplets with a greater precision. The most commonly used resolution is 24 ppqn, which allows sufficient resolution to permit 64^{th} note triplets. At 24 ppqn, a half note corresponds to 48 ticks, a quarter note to 24 ticks, an eighth note to 12 clicks, at sixteenth note to 6 clicks, etc. It should be noted that the actual speed of playback of quarter notes is controlled by tempo change meta-events (see below).

The alternative to ppqn resolution is the SMPTE time code (Society of Motion Picture and Television Engineers 1994), which is divided into hours, minutes, seconds and frames. This is very useful when synchronizing MIDI to media such as film or video. There are variations of SMPTE for different frame rates. The 30 frames per second rate is the one most commonly used by MIDI.

MIDI messages are often used in real-time performances as a communications protocol. It is, however, often convenient to store MIDI data in files to be accessed later, and it is this form of MIDI data that is of interest in this thesis. Although sequencers can use a variety of proprietary formats for storing data, there are three standard MIDI file formats, numbered 0, 1 and 2 (MIDI Manufacturers Association 2001). The main difference between these standard formats is the manner in which they deal with “tracks,” which can be used by sequencers to segment different voices. Format 0 files consist of a single multi-channel track, Format 1 files have multiple tracks that all have same meters and tempos (the first track contains the tempo map that is used for all tracks) and Format 2 files have multiple tracks, each with their own tempos and meters. Format 1 files are the most commonly used, and Format 2 files are very rarely used.

All standard MIDI files consist of groups of data called “chunks,” each of which consist of a four-character identifier, a thirty-two bit value indicating the length in bytes of the chunk and the chunk data itself. There are two types of chunks: header chunks and track chunks.

The header chunk is found at the beginning of the file and includes the type of file format (0, 1 or 2), number of tracks and the division. The division value can mean one of two things, depending on whether it specifies the use of either quarter-note timing resolution or SMPTE. In the former case, it specifies the timing resolution of a quarter note. In the latter case, it specifies the SMPTE frame rate and the number of ticks per SMPTE frame.

Track chunks, in turn, contain all of the information and MIDI messages specific to individual tracks. The time between MIDI events is specified using delta times, which specify the amount of time that has elapsed between the current event and the previous event on the same track. This is done because it requires less space than simply listing the absolute number of MIDI ticks that pass before an event occurs.

MIDI messages and their associated delta times are called “track events.” Track events can involve both MIDI events and “meta-events.” Meta-events provide the ability to include information such as lyrics, key signatures, time signatures, tempo changes and track names in files.

Key signature meta-events include two pieces of information: *sf* and *mi*. *sf* indicates the number of flats (negative numbers) or sharps (positive numbers). For example, C major and A minor are represented by 0 (no sharps or flats), 3 represents 3 sharps and -2 represents 2 flats. *mi* indicates whether the piece is major (0) or minor (1).

Time signature meta-events contain four pieces of information: *nn*, *dd*, *cc* and *bb*. *nn* and *dd* are the numerator and denominator of the time signature respectively. *cc* is the number of MIDI ticks in a metronome click and *bb* is the number of 32^{nd} notes in a MIDI quarter note. It should be noted that *dd* is given as a power of 2, so a *dd* of 3 corresponds to 2^3 = 8. Thus, a time signature of 5/8 corresponds to *nn* = 5 and *dd* = 3.

Tempo change meta-events consist of three data bytes specifying tempo in microseconds per MIDI quarter note. The default tempo is 120 beats per minute if no tempo is specified.

3.2 Feature extraction and evaluation

Musical feature extraction involves processing a recording with the aim of generating numerical representations of what are, hopefully, traits of the recording that are characteristic of the category or categories that it should be classified as belonging to. These features can then be grouped together into feature vectors that serve as the input to classification systems.

Deciding upon appropriate features can be a difficult task, as it is often unclear which features will be useful ahead of time. Furthermore, it is not always easy to extract the kind of features that one would ideally like. Although one can make limited assumptions as to the types of features that might work well based on one's experience and knowledge, there is often at least some amount of guesswork involved in the choice of features in cases such as genre classification, where a diverse and sometimes ambiguous range of factors are involved in category differentiation.

Although there are a wide variety of classification techniques available, their effectiveness is ultimately limited by the quality, consistency and distinguishing power of the features that they are provided with. No classification system can perform its job effectively if the features that it is given do not segment a population into different groups.

Intuitively, one might think that a solution to poor classification performance would be to increase the number of features. Although this is true up to a point, experimental evidence has shown that too many features can actually cause classification performance to degrade. In fact, the "curse of dimensionality" states that, in general, the need for additional training samples grows exponentially with the dimensionality of the feature space (Duda, Hart & Stork 2001, 169-170). More features can also necessitate greater training times. It is therefore necessary to carefully select the features that one is to use, in order to have enough features to be able to effectively discriminate between categories while at the same time not overwhelming the classifier.

From one perspective, redundancy should be avoided when choosing features in order to decrease the total number of features. However, it could also be argued that slightly different representations of similar information may prove more or less successful at distinguishing between particular genres. One should therefore not reject features out of hand simply because of some overlap with other features.

To further complicate matters, having only a few specialized features available for a particular type of classification can be problematic if new categories are added to the classification taxonomy later for which the previously used set of features are inappropriate. It is therefore useful for one to have a large and diverse set of features on hand that can be taken advantage of if new developments in a classification problem make them necessary.

It is clear that it is imperative to have an effective way of evaluating features so that the best ones can be chosen in the context of particular taxonomies and classification problems. This process is known as feature selection. There is also a related procedure, known as feature weighting, where different features are given varying degrees of influence in classifications. The choice of features that will work best ultimately depends on the classification methodology being used, the taxonomy that classifications are based

on and the size and diversity of the training and testing sample sets. The best that one can do in terms of feature evaluation and selection is to start out with a wide selection of candidate features and use feature selection methods to choose ones that work well together for the problem at hand.

Fortunately, there are a range of effective feature selection techniques available. Although few of them can guarantee optimal feature selection even for the training set, to say nothing of the overall population of samples being classified, they often can significantly improve classification results compared to the case where classification is performed without feature selection.

There is a class of dimensionality reduction techniques that operate by statistically analyzing a feature space without needing to perform actual test classifications to evaluate classification performance with test features subsets. This can be an important time saver, as training some classifiers can be computationally intensive. Principle component analysis (PCA) and factor analysis are two commonly utilized techniques of this type. They operate by projecting multi-dimensional data into a lower-dimensional subspace by forming linear combinations of new features in a way that preserves the variance of the original features.

A drawback of these techniques is that an emphasis is placed on those features that have the greatest variability in general, which may not necessarily coincide with the particular variability that one needs when dealing with a particular taxonomy. For the purpose of classification, one is interested in features for which the difference in category means is large relative to the standard deviations of each of these means. Dimensionality reduction techniques such as PCA and factor analysis simply select those features for which the standard deviations are large, and do not consider the particular categories that samples belong to in a particular taxonomy.

Although, there are a variety of multiple discriminant analysis techniques that can be used to perform dimensionality reduction with the particular needs of classification in mind, there is a further problem with all of these dimensionality reduction techniques that must be considered. A by-product of all of these processes is that one ends up with a new set of features that cannot be mapped back to individual features in the original feature space. Although this is not necessarily problematic if one is only concerned with classification performance, it is a significant drawback if one wishes to glean information about how well particular features perform relative to one another or in regards to different kinds of classifications.

Another approach to feature selection involves actually performing classifications of the training samples with different subsets of the features available and seeing which combinations perform best. Those features that do not perform well can be eliminated

from consideration when the final classifier is trained, or can be given lower weightings by the classifier.

Experimental techniques of this type have the advantage that they give one insights into which of the original features are useful in particular types of classifications and which are not. The disadvantage is that these techniques can be quite computationally intensive, particularly if there are many candidate features or if the classification techniques being used to test the performance of different feature subsets are themselves computationally expensive.

A naïve approach would be to simply statistically measure how well each feature individually segments the training data into the appropriate groups and then choose the highest performers. One problem with this approach is that it does not necessarily ensure that features will be chosen that will be effective for classifying samples into every needed category. A feature might perform very well at correctly classify samples belonging to two particular categories, for example, but consistently misclassify samples belonging to a third category.

Although this problem could be solved by incorporating a system that increases the ranking of features that help to distinguish between categories for which no other effective features have been found, there is another important problem that must be considered. Several separate features that have little discriminating power when considered individually, and which might therefore be rejected by this naïve feature evaluation approach, could in fact be very useful when considered together. A feature evaluation system must therefore consider the discriminating power of features operating collectively as well as individually.

Ideally, one would like to exhaustively test every possible subset of candidate features. This would ensure that the features selected would be optimal for the training samples. Of course, this does not necessarily guarantee that they will be optimal for the overall population, so overfitting can be a problem, but then no feature selection method of any kind can guarantee that results based on training samples will generalize well to the population as a whole.

Exhaustive examination of all possible combinations of features is certainly attainable in some cases, but in others limitations on computing power and the demands of the task make it practical only to test a fraction of all possible feature combinations. This is particularly true when dealing with large feature sets, a great number of training samples and computationally intensive classifiers. Although techniques such as branch and bound searching can be used to improve the speed of exhaustive examinations, there can still be many cases where exhaustive examinations are intractable. Fortunately, there are a variety of techniques available that allow one to find a good, although not necessary optimal, feature subset without needing to test all possible combinations.

One simple approach is to limit the number of selected features to some maximum number *n*, and then exhaustively test the performance of all feature subsets of size *n* or less. This has a good likelihood of performing well, as the curse of dimensionality implies that one would not want to use all that many features in any case. A variation of this technique is to choose the best *m* feature subsets selected this way and implement a separate classifier for each. Final classification results could be arrived at by combining the results of these classifiers using a coordination system (see Section 6.5 for more information on the coordination of classifiers). These approaches can significantly cut down on computational demands relative to exhaustive searches, but they can still be computationally intensive if *n* is high or there are many features.

Forward selection and backward selection are two common feature selection techniques that allow one to further reduce computational demands. Forward selection operates by starting with the empty set and adding features one by one until some maximum number of features are selected. This process begins by first considering all possible feature subsets consisting of one feature only, and choosing the best one. Next, all possible feature subsets consisting of this feature and one other feature are considered, and the best performer is chosen. Features are thus added iteratively one by one to the previously selected features until the maximum number of features are attained. Backward selection, in contrast, starts with the full set of features, and iteratively removes features one by one.

A problem with these two techniques is that there is no way to remove (or restore) a feature once it has been added (or taken away). This problem, called nesting, can be significant, since a feature that performs well early on in the feature selection process may actually not be one of the best features to choose. A technique called forward "plus l take away r" overcomes nesting by first applying forward selection *l* times and then applying backward selection *r* times. A variation of this technique, called sequential floating selection, dynamically assigns the values of *l* and *r*, instead of fixing them (Pudil et. al 1994).

An alternative way to select features experimentally is to use a class of techniques called genetic algorithms (GA's). GA's can be used for a wide variety of purposes, and are often used for optimization problems where exhaustive searches are not practical. Siedlecki and Sklansky (1989) pioneered the use of genetic algorithms for feature selection, and they have been used successfully in the past with respect to music classification (Fujinaga 1996). Recent research has found that GA's are particularly effective "in situations where the search space is uncharacterized (mathematically), not fully understood, or/and highly dimensional," all of which certainly apply to the task of genre classification (Hussein, Ward & Kharma 2001). There is also some evidence that,

in general, GA's perform feature selection better but slower than greedy search algorithms (Vafaie & Imam 1994).

GA's are inspired by the biological process of evolution. They make use of data structures called "chromosomes" that iteratively "breed" in order to evolve a hopefully good solution to a problem. Each chromosome consists of a bit string that encodes the solution to the problem that the GA is being used to solve. This bit string is effectively the DNA of a chromosome, and is combined with the bit strings of other chromosomes when breeding occurs. Each bit string has a fitness associated with it, which indicates how well its bit string solves the problem at hand.

In the case of feature selection, each bit in the bit string could represent a feature, with a value of 1 implying that the feature is to be used and a value of 0 implying that it is not to be used. Alternatively, bit strings could be segmented into multi-bit words, each of which encodes a numerical value that represents the weighting of a feature.

A GA begins with a population of many chromosomes whose initial bit strings are randomly generated. Reproduction, and hence the evolution of a solution, occurs through a process called crossover. Some fraction of the chromosomes, based on a GA parameter called the crossover rate, is selected for reproduction, with the remaining chromosomes being eliminated. One way of selecting the chromosomes that will reproduce, called the roulette method, assigns a probability of selection for reproduction to each chromosome based directly on its fitness. An alternative, called rank selection, ranks the chromosomes based on fitness and then bases the probability of selection for crossover on this ranking. This latter approach prevents one or a few chromosomes with very high relative fitnesses from dominating early on, as this could lead to a local minimum in error space that is far from the global minimum.

In general, the actual process of crossover involves taking two of the chromosomes selected for breeding and breaking the bit string of each one into two or more parts at randomly selected locations. The resulting child is generated with a bit string constructed from the pieces of the bit strings of its parents. An alternative approach involves going through the bit strings of the parents one bit at a time and copying the bits to the child when the values for the parents correspond and randomly selecting a bit's value when the corresponding bit of the two parents differs. Each parent chromosome can reproduce multiple times, either polygamously or monogamously.

There are several additional features that are sometimes incorporated into GA's as well. "Mutation" involves assigning a probability (usually very small) to every bit of every child's bit string that the bit will be flipped. Elitism involves automatically cloning the chromosome with the highest fitness from one generation to the next. Villages or islands involve segregating the chromosomes into different groups that evolve independently for the most part, but occasionally exchange a few chromosomes.

A simplified version of genetic algorithms, known as random mutation hill climbing, is also occasionally used. This involves eliminating the crossover step, and having the population improve through mutation only.

Genetic algorithms have been shown to find good solutions to many problems, although not necessary the optimal ones. Their success is highly dependant on the goodness of the fitness function that is used. In the case of feature selection, this fitness function is related to how well the selected feature subsets perform in classifying the training set.

GA's can be computationally expensive, particularly if fitness evaluation is expensive, since fitness must be calculated for each chromosome at each generation. This makes them suitable for classification techniques that require little or no training time, such as nearest-neighbour classifiers (see Section 3.3), but less appropriate for classifiers that are computationally expensive to train.

It should be noted that simple statistical techniques could be combined with any of the methods listed above in order to offer some additional increases to performance. For example, one could consider features one by one, and measure the standard deviations of the central means of each feature for each classification category. A high standard deviation would indicate a good discerning power for a feature. The standard deviation of a feature's values within each category could also be considered, as a large variation would indicate that the feature would likely not coherently cluster samples in the given category, and would perform poorly if one's goal is to find all samples belonging to that category, for example. Cross-correlations and covariances could also be calculated in order to see how similarly different features change across categories in order to eliminate redundant features.

Although this approach certainly has limitations that would hamper it usefulness if it was used alone, as discussed earlier in this section, it could nonetheless provide some basic information that could be used by an expert feature selection system, for example, that also makes use of some of the more sophisticated techniques discussed earlier in this section, such as GA's.

There are a number of lesser used feature selection techniques that have not been discussed here. Jain and Zongker (1997) have written a good overview of experimental feature selection methods as well as an empirical study of their relative performance. Kirby (2001) offers a good resource for those looking for more specialized techniques than those discussed here. The books of Duda, Hart and Stork (2001), Fukunaga (1972) and James (1985) provide good references on feature selection in general. Sakawa's book (2002) is an excellent resource for more information on genetic algorithms in general, and Hallinan (2001) provides an excellent literature review and practical guide to the use of genetic algorithms for feature selection.

3.3 Classification methods

There are a variety of methods available for automatically classifying information based on features. For reasons discussed in Section 1.5, only supervised learning techniques are considered here. Which classification technique is appropriate to use in a given situation depends on the type of population that is being classified, the amount of knowledge one has about the statistical properties of the population, the taxonomy that is being used to organize the population, the types of features that are being used, the computational power available and the amount of training and classification times that are considered acceptable.

There is not enough space here to cover the full range of classification techniques available in any amount of detail. This section therefore limits detailed descriptions to those classifiers which are actually used in this thesis, namely k-nearest neighbour and feedforward neural network-based classifiers. Before describing the details of these methods, however, other techniques are briefly mentioned in order to make it possible to explain why they were not used, and to make it clear under what circumstances it would be appropriate to use them. Those looking for further details should consult Duda, Hart and Stork's (2001) or Jain et al.'s review of classifiers (1999). The books of Russell and Norvig's (2002) and Mitchell (1997) are also excellent references for those looking for information on machine learning. Briscoe and Caelli (1996) cover a variety of existing machine learning systems that have been implemented, although some of these have become somewhat dated.

Rowe (2001) is a good resource for those wishing to apply artificial intelligence techniques to musical tasks. Although this book touches on the actual techniques in less detail and much less rigorously than some of the other resources discussed in this section, it is a very good introduction to artificial intelligence from a musical perspective and provides some good ideas relating to how music can be conceptualized and represented in ways that computers can deal with effectively.

In the ideal case, one would have full knowledge of the probability structure underlying the population that one is classifying. This means that the statistical distribution of the population's features would be known. If this is the case, then one can use the optimal Bayes classifier. If, rather than knowledge of the distribution based on parameter vectors one has knowledge of the causal relationships among component variables, it is then possible to use Bayesian belief networks.

It is unusual that one has this amount of knowledge about the properties of a population, unfortunately. Even if one does not know the particular parameters of a population's distribution, however, knowledge of the general form of the distribution makes it possible to use techniques such as maximum-likelihood estimation and Bayesian parameter estimation in order to estimate them. One such technique, hidden Markov

models, is of particular interest if one wishes to perform classifications that take into account the temporal order of events. Rabiner in particular has written two good tutorials on hidden Markov models (Rabiner & Juang 1986; Rabiner 1989).

There are also classification techniques available if one has no prior knowledge of the underlying probability structure of a population, and is forced to rely only on information gleaned from training samples. This class of techniques is known as nonparametric techniques, and include the nearest-neighbour algorithm and potential functions. There are also nonparametric methods for transforming the feature space with the goal of being able to employ parametric metrics after the transformation. These include analysis methods such as the Fisher linear discriminant. There is also a related class of classification techniques that involve neural networks, a set of techniques inspired by, although not accurately simulating, the processes utilized by the neurons of human and animal brains.

Another class of techniques use functions to map the features of a problem to a higher dimensional space, where appropriate category boundaries can be found. Support Vector Machines (SVM's) are a well-known example of this type of classifiers.

Finally, there is a class of classification techniques that are based on learning logical rules rather than using statistical methods. These nonmetric methods include tree-based algorithms and syntactic-based methods involving grammars. These techniques require the use of nominal data, which is to say features with discrete values that do not involve any natural notion of similarity or ordering. In practical application, continuous features can be divided into discrete categories (e.g. small or big), although the dividing point is usually somewhat arbitrary, and these techniques are better suited to features where no such arbitrary dividing point is necessary (e.g. has a tail or does not have a tail).

In the particular case of musical genre classification as dealt with in this thesis, no *a priori* knowledge is available regarding the distribution of the feature values of the population, so it would be inappropriate to make the unfounded assumptions that would be necessary to use techniques that rely upon such knowledge.

Nonmetric methods are certainly attractive, as they not only perform classifications, but also provide information about how the classifications were arrived at. Part of the goal of this thesis is to develop and use a wide library of candidate features for classification, however, where values are often continuous and dividing points are not necessarily obvious. This limits the applicability of nonmetric methods, with their requirement for nominal data, to the task at hand. Future research using specialized features and nonmetric classifiers could certainly be of interest, however.

Nonparametric techniques stand out as the best choice for situations such as automatic musical genre classification of the type studied here, where one must rely solely on training samples in order to learn the information necessary to perform classifications. Of

the nonparametric techniques available, k-nearest neighbour and neural network-based classifiers are two of the most commonly used, and have repeatedly been found to perform well for a variety of classification tasks. They therefore deserve special attention.

The k-nearest neighbour (KNN) classification technique is one of the simplest and most commonly used nonparametric methods. Training operates simply by storing the coordinates of each training sample in the multi-dimensional feature space. Test samples can then be classified by examining their surroundings in feature space and finding the labels of the nearest k training points. In other words, a cell is grown in feature space around a test point until the k nearest training points are captured. If a test point happens to fall in a region of high training point density, the cell will be relatively small, and if the test point falls in a region of low density, then the cell will be large, but will rapidly stop expanding when it enters a region in feature space of high density.

The probability that a test point belongs to a given category can be estimated by the number of training points captured divided by k. This is useful, since it allows one to obtain a score for potentially more than one category when classifications are performed, which is beneficial if an ensemble of classifiers is being used or if samples can belong to multiple categories.

The choice of k is important, as a k that is too large can include more points than there are training points of the appropriate category, thus inappropriately including points of other classes even if the training points are well clustered. A k that is too small, on the other hand, can make the classifier too sensitive to noisy training points. Rather than using an arbitrary constant for k, it is usually a function of n, the number of training samples, most commonly the square root of n. In practice, however, it might be wise to also make it a function of the number of training samples in each category, otherwise categories with too few training samples could be overlooked by the classifier as the captured area expands beyond their neighbourhood in order to capture k training points.

There are a variety of distance measures that can be used to calculate which points are closest to a test point. Euclidean distance is often used, although it is sometimes appropriate to use alternatives such as Manhattan distance, Tanimoto distance or tangent distance. It is often wise to pre-process features in order to ensure that they have roughly the same range of values, otherwise features with large values will potentially inappropriately dominate distance measurements over features with small values.

An important advantage of KNN classifiers is that they require a negligible amount of training time, since all that they must do is memorize the coordinates of the training samples. Classifying a test point can be somewhat computationally expensive using a naïve implementation of KNN, however, particularly if a large number of features (d) and, more importantly, training samples (n) are used. Using big-O notation, the naïve

implementation actually has $O(dn^2)$ complexity. A variety of alternative computationally efficient KNN implementations are discussed in Duda, Hart and Stork (2001).

A variation of the basic KNN classifier involves assigning weights to the different features, so that some play a more important role in distance measurements. This is tied to feature selection weighting methods. Although this can improve classification accuracy, it has the disadvantage that some method must be used to train the weightings, such as genetic algorithms, that will likely require additional training time.

Despite their advantages in training speed, KNN classifiers are limited in the ways in which they can model relationships between features. Classifications are based entirely on distances, and no considerations of conditional relationships can be made. For example, if one is attempting to classify musical recordings based on the fraction of all notes played by each instrument, the presence of an electric guitar would automatically eliminate all traditionally performed pre-20^{th} century classical music categories from contention. A classifier capable of building conditional relationships between features, such as a human, could then consider other features with the assumption that the recording is not classical music in mind. KNN classifiers, in contrast, are incapable of deducing such sophisticated relationships, and can only consider features as a whole. The extensive use of a flute in a recording also containing a small amount of electric guitar, for instance, might cause a KNN classifier to conclude that the recording is a classical flute sonata, rather than the correct classification of, for example, progressive rock.

Neural networks (NN's) provide a means of performing classifications that can encapsulate more sophisticated relationships than KNN classifiers. Although there are many varieties of NN's used for many types of tasks including but not limited to classification, the emphasis is placed here on multilayer feedforward NN's, which are the type most often used for supervised learning for the purpose of classification. There has been a significant amount of research showing the applicability of NN's to musical problems, such as that by Stevens and Latimer (1997), to cite just one example.

NN's are inspired by and loosely modelled on biological processes in the human brain. Neural networks are composed of units (inspired by human neurons) that are connected by links, each with an associated weight. Learning in NN's takes place by iteratively modifying the values of the weights.

Units in feedforward NN's are organized into layers. The input layer is composed of units where patterns of numbers are fed into the network. For classification purposes, each unit in the input layer is normally given values corresponding to a single feature. There is also an output layer, whose units provide the output of the network in response to the patterns placed on the input units. One approach to using networks for classification tasks involves having one output unit for each possible classification category. There may

also be one or more hidden layers, whose units and weights are used for processing and learning by the network, and which are treated as the inside of a black box.

Each unit in a feedforward NN has a link to every unit in the previous layer (if any) and to every unit in the subsequent layer (if any). Each unit also has a current "activation level," which is the value propagated down the links leading to the subsequent layer (or to the output sensors).

The activation level of a unit is calculated by adding the values on all of the links coming into the unit from units in the previous layer (or input sensors) and processing the sum through an "activation function." The values on each link are calculated by multiplying the value placed on it by the link's weight. Some feedforward NN's also include a "bias unit" for each hidden or output unit that outputs a constant value through a link

The activation function is generally a function with a range of 0 to 1 that, generally speaking, maps input sums over 0 to 1 and input values under 0 to 0. This function must be continuous, as its derivative is taken during training. A commonly used such function is the sigmoid function:

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{1}$$

So, to put it more formally, the net input into an input unit is simply the input value placed on it. The net input into a hidden or output unit *j* is given by:

$$net_j = w_b b + \sum_{i=1,n} w_{ij} o_i \tag{2}$$

where w_b is the weight of the bias link, b is the bias constant (0 if no bias units are used), each i corresponds to a unit in the preceding layer, n is the number of units in the preceding layer, w_{ij} is the weight on the link from unit i to unit j, and o_i is the output of unit i. The output of a unit j with net input net_j and a sigmoidal activation function is:

$$o_j = sigmoid(net_j) \tag{3}$$

Feedforward NN's are trained by putting training patterns on their inputs and observing how the output of the network differs from the model output. Weights are then adjusted in order to make the output closer to the model through a process called gradient decent. The process of first adjusting the weights leading into the output nodes and then successively adjusting the weights in each earlier layer is known as "backpropagation."

The training modifications to the weights leading into an output unit k are adjusted by first calculating δ_k, the error signal:

$$\delta_k = (t_K - o_k) f'(net_k) \tag{4}$$

where t_k is the target or model activation value for the unit k, o_k is the actual activation value, net_k is the net input into k, and f' is the derivative of the activation function. For the sigmoid activation function:

$$f'(net_k) = o_k(1 - o_k) \tag{5}$$

The weights leading into output unit k from each unit j in the preceding layer, w_{jk}, are adjusted as follows from iteration t to iteration $t+1$:

$$w_{jk}(t+1) = w_{jk}(t) + \eta \delta_k o_j + \alpha \Delta w_{jk}(t) \tag{6}$$

where

$$\Delta w_{jk}(t) = w_{jk}(t) - w_{jk}(t-1) \tag{7}$$

and η and α are NN parameters known as the learning rate and the momentum respectively. The momentum term is sometimes omitted. These parameters are often set to between 0 and 1, and control how quickly a NN converges towards a stable solution and how likely it is to get stuck in a poor local minimum in error space far from the global minimum. The learning rate controls how large the adjustments to the weights are each iteration (i.e. how large the steps are in error space towards a solution) and the momentum stops the network from oscillating wildly in error space from iteration to iteration by taking into account the previous adjustment to each weight. Increasing the learning rate can cause a network to converge faster, but a value that is too high may also cause the network to jump around so much that it does not converge. Increasing the momentum also usually increases the speed of convergence, but can cause poor performance if it is set too high. It should be noted that the initial value of the weights is randomly determined, and the range of initial values that are permitted here can also have an effect on how a network converges.

The error rate for a hidden unit, j, behind units k in a subsequent layer is given by:

$$\delta_j = f'(net_j) \sum_k \delta_k w_{kj} \tag{8}$$

The weight change formula for the weights of the input links to a hidden unit is the same as that for an output unit (equation 7).

There are a variety of ways of going about the training process, all of which usually produce similar results. One can simply feed the training samples into the network one by one, and adjust the weights after each sample. This can be done as above or, if one is

inclined to be more mathematically orthodox, one can only actually implement the weight adjustments after all of the adjustments have been calculated for a particular training sample. Alternatively, one can choose to not actually implement the weight changes until all of the training samples have been run through the network once, after which all of the weight changes are implemented collectively. In practice, either approach works well, but it is generally a good idea to randomly order the training samples if the former approach is used, to avoid receiving many similar patterns consecutively, which could be conducive to falling into a poor local minimum in error space.

In any event, the entire training set usually needs to be input to the network during training many times before the network converges to a solution. Each time the entire training set is processed once and the weights updated is called an epoch or a training iteration. An indication of convergence can be measured by observing the rate of change from epoch to epoch of the sum of squares error, *E*:

$$E = \frac{1}{2}\sum_{k}(t_k - o_k)^2 \tag{9}$$

between the model output activation values and the actual values. Convergence can generally be said to have occurred when this error stops changing significantly from epoch to epoch, although it is not unknown for a network to seemingly converge for a large number of epochs before suddenly changing dramatically.

Like any non-optimal system, neural networks carry the danger of converging to a local minimum in error space that corresponds to a significantly worse solution than the optimal solution. This is in general unavoidable, since the optimal solution is likely unknown and its calculation likely intractable if one is resorting to a non-optimal system such as neural networks. This makes it difficult to know whether a particular solution is a poor local minimum or not. Some experimentation with the NN parameters for a particular problem can help to reduce the likelihood of converging to a particularly poor solution.

There is some disagreement in the literature as to the appropriate number of hidden layers to use and how many hidden units should be used in all. de Villiers and Barnard (1992) do offer some convincing arguments that more than one hidden layer can actually degrade performance rather than improve it. There have been a number of formulas proposed as to the ideal number of hidden units to use. For example, Wanas et al. (1998) claim that the best performance, in terms of both performance and computation time, occurs when the number of hidden nodes is equal to *log(n)*, where *n* represents the number of training samples. There is no real consensus on this matter in the literature, and the optimal number of hidden nodes likely depends on the particularities of each individual problem. A good experimental approach is to test performance by gradually

adding hidden units one by one until the performance fails to improve by a certain amount. There are many variations of this approach, including Ash's pioneering work (1989). Another approach is to use techniques such as genetic algorithms to optimize network architecture. In any case, it is true in general that increasing the number of hidden units increases the complexity of the function that can be modelled, but also increases the training time and, potentially, the probability that the network will not converge.

The initial weightings of the networks can also have an important effect on what solution a network converges. One simple approach is to randomly determine the initial weightings of the network within some range, and experiment until an effective range is found. Maclin and Shavlik (1995) have suggested the alternative of using competitive learning to find an initialization that maximizes the number of attainable local minima that the network can converge to.

Feedforward NN's with no hidden layer (called "perceptrons") initially caused a great deal of excitement because of their ability to simulate relations. There was some disappointment when it was found that there are certain functions which they cannot simulate, such as XOR logical relations. Fortunately, it was later found that feedforward NN's with one or more hidden layers can in fact overcome this problem, and can indeed be used to model complex relations. This means that NN's can be used in cases where one wishes to model more sophisticated relationships between features than classification techniques such as KNN are capable of doing. Unfortunately, the disadvantage of neural networks is that they can take a significant amount of time to train.

Abdi and Valentin (1999) provide an excellent introduction to the inner workings of NN's, and include some revealing worked examples. Bengio (1996) provides some useful practical guidelines regarding working with NN's. Gallant (1993) is also a good resource, and includes very interesting sections on combining neural networks with expert systems and some ideas on how rules can be extracted from trained neural networks. Adeli and Hung (1995) provide useful background information on both neural networks and genetic algorithms, and some ideas on how the two techniques can be used together. Fausett (1994) is also a good general reference on neural networks.

Those interested in how different machine learning techniques can be combined to improve performance of practical systems may wish to consult the book edited by Michalski, Bratko and Kubat (1999). The book edited by Kandel and Bunke (2002) includes further information in this vein. The book edited by Jagannathan, Dodhiawala and Baum (1989) contains a good deal of information on blackboard systems that could also be useful in this respect.

Before moving on, it is appropriate to briefly discuss the division of one's sample examples into training and testing groups in order to evaluate the performance of any

supervised classifier. It can happen that an inexperienced researcher working with classifiers will observe that a classifier perfectly classifies his or her training examples, and then conclude that the classifier works very well. Unfortunately, this conclusion may not necessarily be correct. Classifiers can learn to classify a training group perfectly, yet fail to make the generalizations necessary to accurately classify the population as a whole. This effect is known as overfitting. Successful classification of the training group is not therefore necessarily indicative of a successful classifier. In order to realistically test a classifier's performance, one must reserve a set of testing samples that the classifier does not have any contact with during training. The measured performance of the classifier must then be based on how well the classifier classifies these test samples after training, as this provides a much better indication of how well the classifier is likely to perform on the population in general.

Care must be taken when selecting the training and testing samples not only that representative samples are taken from all categories in the classification taxonomy, but that all sub-groups, if any, within each category are present as well. Although the exact ratio of training to testing samples can vary, 10% to 20% of the available samples are typically reserved for testing. Generally speaking, a greater number of total available samples usually corresponds to a smaller percentage of samples reserved for testing, as fewer samples are needed to achieve statistical significance.

4. Previous Research in Music Classification

4.1 Overview

The majority of research on genre classification to date has been on the classification of audio data rather than symbolic data. Although the features extracted from audio data are usually very different from those extracted from symbolic data, most other aspects of the two types of classification are very similar. It is also important to note that the intersection of the features that can be extracted from audio and symbolic data is increasing as automatic transcription techniques improve. It is therefore useful to examine research on audio classification for ideas that can be adapted to symbolic data classification and to create a basis for future expansion of this research into the audio realm. Section 4.2 discusses previous research on the classification of audio data, and Section 4.3 covers research involving symbolic data.

It should be noted that a number of papers on systems that perform classifications based on performer and/or composer styles have also been included here. It is useful to examine these systems as well as genre classifiers, as both types of systems often use similar features and classification techniques.

A selection of papers on systems that use unsupervised learning techniques to perform classifications have been included here as well. Although unsupervised learning is not as appropriate for genre classification as supervised learning, as discussed in Section 1.5, there has been a significant amount of research using these systems, and it is important to be aware of it. The popularity of unsupervised learning in research up to this point may be due to the fact that most systems have used only a limited number of very distinct categories. It is probable that unsupervised systems would form categories that coincide with human genre categories when dealing with such taxonomies. More realistic taxonomies, with large numbers of categories that include overlap and potentially objectively irrational distinctions from a content-based perspective, would be highly unlikely to match the categories formed by unsupervised classifiers, however. So, although unsupervised classifiers have performed well to date in regards to genre classification, it is very improbable that these types of systems would scale well to realistic genre taxonomies.

This does not mean that research involving unsupervised learning is not worth pursuing, however. Classification using unsupervised learning is in a very real sense equivalent to similarity measurements, an area of research with important applications outside of the scope of genre classification. The kind of similarity groupings produced by unsupervised learning could, for example, produce a means of navigating music databases that could potentially be more useful than genre categories for certain applications, such

as if one is simply looking for new types of music that are similar to one's existing preferences in a way that is not limited by the peculiarities of genre taxonomies.

There has also been some research into automatically generating music of given styles, which is in a sense the inverse of the classification problem. Although some such systems operate using a rules based approach, others dynamically analyze music and attempt to generate music that is similar to it. The analysis portion of systems of the latter type could operate similarly to classification systems, and are therefore worth mentioning here. Two of the most salient papers on the subject are discussed in Section 4.4.

It should be noted that this chapter only briefly covers the topics and results of each of the publications that are presented. Those aspects of certain publications that were used directly in this thesis are discussed in more detain in Chapters 5 and 6.

Overall, success rates of between 50% and 92% have been achieved in previous studies when classifying between three to ten categories using only musical content-based features. As one would expect, success rates have in general been inversely proportional to the number of categories. These success rates are promising, especially considering that most humans would themselves likely be unable to achieve rates beyond 80% or 90%. A large variety of features and classification techniques have been used, leading one to think that there may be a number of different but valid solutions to genre classification.

Table 1 summarizes the results of studies where musical genre classification success rates were reported. More details on these studies are available in Sections 4.2 and 4.3. One should keep in mind that the variety of taxonomies and recordings used makes it difficult to compare success rates directly in order to conclusively judge one system as better than another.

Unfortunately, no results have been published to date, to the best of the author's knowledge, of the application of a classification system to a realistic taxonomy with large numbers of categories. It is one of the goals of this paper to fill this gap.

4.2 Classification of audio data

George Tzanetakis has performed the most widely cited research on automatic genre classification to date. His first major paper on the subject (Tzanetakis, Essl & Cook 2001) presented two real-time GUI-based systems that performed musical genre classification of audio signals. The first, GenreGram, developed for real-time radio broadcasts, displayed cylinders representing each genre that moved up and down based on the confidence at any given moment that a recording belonged to a particular genre. The second GUI, GenreSpace, provided a 3-D representation of genre space and mapped each recording to a point based on its three most distinguishing features. GenreSpace was meant to be used for representing large collections of recordings. Classifications were made based on a

Type	Researchers	Date	Techniques	Genres	Recordings	Success
Symbolic	Chai & Vercoe	2001	HMM	3	491	63%
Symbolic	Ponce de Leon & Inesta	2002	Self-organising maps	2		77%
Symbolic	Shan & Kuo	2003	Associative classification	2		84%
Metadata & Audio	Whitman & Smaragdis	2002	Neural network	5	300	100%
Audio	Burred & Lerch	2003	GMM	13	850	60%
Audio	Deshpande, Nam & Singh	2001	k-NN	3	157	75%
Audio	Grimaldi, Kokaram & Cunningham	2003	Binary k-NN	5	202	73%
Audio	Jiang et al.	2002	GMM	5	1500	91%
Audio	Karpov	2001	HMM	3	252	90%
Audio	Kosina	2002	k-NN	3	189	88%
Audio	Lambrou et al.	1998	4 distance based classifiers	3	12	90%
Audio	Li & Tzanetakis	2003	Linear discriminant analysis	10	1000	71%
Audio	Li, Ogihara & Li	2003	SVM	10	1000	79%
Audio	Matityaho & Furst	1995	Neural network	2	8	100%
Audio	McKinney & Breebaart	2003	Gaussian framework	7	188	74%
Audio	Pye	2000	GMM	6	175	92%
Audio	Solatu et al.	1998	ETM-NN	4	360	86%
Audio	Tzanetakis & Cook	2002	GMM & k-NN	10	1000	61%
Audio	Tzanetakis, Essl & Cook	2001	Gaussian classifier	6	300	62%
Audio	Xu et al.	2003	SVM	4	100	93%

Table 1: Summary of existing musical genre classification systems.

tree-based genre hierarchy containing both music and speech. A 62% success rate was achieved in classifying between six genres.

The ideas presented in this paper were further developed in a second paper (Tzanetakis & Cook 2002), where a fully functional system was described in detail. The authors proposed using features relating to timbral texture, rhythmic content and pitch content to classify pieces. The system was able to correctly distinguish between audio recordings of ten genres 61% of the time. Results of music/speech and sub-genre classifications were also presented. A variety of statistical classifiers were used. A refinement of this work was published by Li and Tzanetakis (2003) and Li, Ogihara and Li (2003), where classification results of between 70% to 80% were achieved through the use of linear discriminant analysis and Daubechies wavelet coefficients.

A further paper (Tzanetakis, Ermolinksyi & Cook 2002) argued that pitch-based features could be used to enhance content-based analysis of music. As a demonstration, a genre classification system was built to distinguish between five genres. A success rate of

50% was achieved with the use of only four pitch-based features and no rhythm or timbre related features. A k-NN classifier was used in this system.

Tzanetakis' dissertation (2002) presented a number of approaches and techniques for extracting information from audio recordings, including techniques related to genre classification. Tzanetakis reviewed and brought together research that he had previously published.

Another detailed description of a genre classification system can be found in Karin Kosina's thesis (2002). This system used KNN classification to achieve a success rate of 88% when classifying audio data into one of three genre categories. This thesis provides a good overview of background information in the field of genre classification.

Grimaldi, Kokaram and Cunningham (2003) described a system that used a discrete wavelet transform to extract time and frequency features, for a total of sixty-four time features and seventy-nine frequency features. Instead of using a single classifier to classify all genres, Grimaldi et al. used an ensemble of binary k-NN classifiers, each trained on only a single pair of genres. The final classification was arrived at through a majority vote of the classifiers. Tests achieved a success rate of 73.3% with the binary classifier ensemble, as compared to only 63.6% when a single classifier was used for all of the genres. Tests were performed using a total of five genre categories.

Xu et al. (2003) proposed using a two-level classification system, where a broad classification was first made, followed by a finer classification based on the results of the first classification. The authors implemented such a system by having the first stage classify audio data into either rock/jazz or pop/classical and then having the second stage decide between either rock and jazz or pop and classical, depending on the output of the first stage. Different sets of features were used for each of the above three classifiers. This is an approach that would fit in well with a hierarchical taxonomy. A success rate of 93% was obtained using support vector machines. Further tests of the system using nearest neighbour, Gaussian mixture model and hidden Markov model methods resulted in significantly lower success rates.

Burred and Lerch (2003) compared the performance of basic flat and hierarchical classification techniques on 13 musical and 4 non-musical genres, and achieved success rates of approximately 60% using both techniques. A three component Gaussian mixture model was used to perform classifications based on 18 features.

Pye (2000) compared a Gaussian mixture modelling classifier and a tree-based vector quantization classifier for the purposes of audio classification, and achieved an accuracy of 92% with the former when classifying between Blues, Easy Listening, Classical, Opera, Dance (Techno) and Indie Rock music.

Lambrou et al. (1998) achieved success rates of 90% or above when classifying between categories of rock, piano and jazz. A comparative analysis was made between

different wavelet analysis techniques, with the best results achieved with the adaptive splitting wavelet transform. Four different distance-based classifiers were used.

Matityaho and Furst (1995) used a large feed-forward neural network and spectral components to perform classifications. The authors achieved a 100% success rate when classifying 2.8 second segments of audio. Although this is impressive, particularly considering the limited feature set, the system only considered Classical music and Popular music, so testing with more categories would be required to truly test the system.

Deshpande, Nam and Singh (2001) built a system that used a variety of classifiers to separate audio recordings into Rock, Classical and Jazz categories. The best three-way results were obtained by the k-nearest neighbour classifier, with an accuracy of 75%.

McKinney and Breebaart (2003) published a study comparing four different audio feature sets in terms of their ability to classify music as Jazz, Folk, Electronica, R&B, Rock, Reggae and Vocal. Success rates of between 61% and 74% were achieved, depending on the feature sets used, with auditory filterbank temporal envelope-based features outperforming low-level signal parameters, Mel Cepstral Coefficients (MFCC) and psychoacoustic features. Classification was performed using a standard Gaussian framework.

Karpov (2001) used hidden Markov models and spectral features to classify recordings into four categories (Celtic, Western Classical, Techno/Trance and Rock). Success rates of over 90% were achieved with three-way classifications. Karpov offered the interesting suggestion that hidden Markov models could be used in future research to initially classify music into broadly different categories and other classifiers, such as neural networks, could then make finer classifications.

Soltau et al. (1998) proposed a method that they called Explicit Time Modeling with Neural Networks (ETM-NN) that could be applied to musical genre classification. This method is based on finding an effective way of using neural networks to deal with features based on temporal structures. They used their system to classify audio data as Rock, Pop, Techno or Classical. They argued that their ETM-NN system provides a superior alternative to the hidden Markov models that have often been used to perform classifications using temporal data. They fed the same features into both an ETM-NN and HMM, and achieved success rates of 86.1% and 79.2% respectively.

Jiang et al. (2002) presented an “octave-based spectral contrast” feature that represents relative spectral distribution in order to improve classification of audio data. A success rate of 90.8% was achieved for classifying full recordings into Baroque, Romantic, Pop, Jazz or Rock categories. The classification was performed using a Gaussian mixture model.

Jennings et al. (2003) developed a new method to quantify the behaviour of the “loudness fluctuations” of an audio signal. Although no classification success rates were

reported, correlations were found between loudness fluctuations and high art music, popular music and dance music. Although the results found by the authours are not exceptionally impressive in themselves, the signal processing techniques that were used do hold some potential.

Crump (2002) devised a system that classified audio recordings based on composer's style. The system used neural networks, and was successful in distinguishing between Bach and Mozart recordings 73% of the time using only one second long segments of music.

Frühwirth and Rauber (2001) used a self-organizing map to organize a collection of audio files according to their genres and sound characteristics. Melodic information was included in the feature vectors that were used. Classification was performed by first clustering segments of recordings based on similarity and then clustering recordings based on their segments. A further discussion of the uses of self-organizing maps for organizing music has been written by Rauber and Frühwirth (2001).

Pampalk, Rauber and Merkl (2002) constructed a system that analyzed audio data and presented it to users using a visual interface that made the relationships of different genre categories to each other intuitively apparent. A self-organizing map was used to cluster recordings based on genres. More details of this system are available in an earlier publication of Pampalk (2001).

In an effort to expand on the use of self-organizing maps, Rauber, Pampalk and Merkl (2002) used a growing hierarchical self-organizing map to create a hierarchical organization of music based on similarity of audio recordings. This is a promising approach, as the incorporation of a hierarchy into the similarity structure has the potential to create a structure that is easier for humans to navigate than categories on the same level. The features that were used incorporated ideas from psychoacoustic models, including loudness and rhythm perception.

Whitman and Smaragdis (2002) have taken the important step of combining audio content-based features with cultural features in order to classify music. They used what they called "community metadata" that was derived from text data that was mined from the web. Classification was done using a time-delay neural network in order to incorporate a short-term memory into the system. Although cultural and audio data both performed relatively poorly when classifying recordings independently between Heavy Metal, Contemporary Country, Hardcore Rap, "Intelligent Dance Music" and R&B, the authors claimed a success rate of 100% when features of both types were combined. This is very encouraging, and certainly provides justification for further research involving more categories.

4.3 Classification of symbolic data

One of the earliest works on the topic of automatic genre classification was published by Gabura (1965). This paper only deals explicitly with classical music, unfortunately, which limits its applicability. Despite this and its age, however, this paper nonetheless offers some interesting ideas that appear to have been overlooked in many later publications, particularly in regards to the use of relatively sophisticated statistics and theoretical models to derive features.

Shan and Kuo (2003) published one of the few papers dealing directly with genre classification of MIDI recordings. They extracted features based exclusively on melodies and chords, and obtained success rates between 64% and 84% for two-way classifications. Recordings all belonged to one of four categories (Enya, Beatles, Chinese folk and Japanese folk), with 38 to 55 files used for each category. This research is particularly valuable in terms of the ways in which melodic and chordal features were extracted, and it would have been interesting to see how well the system would have performed with a greater variety of features and a larger number of categories.

Chai and Vercoe (2001) used hidden Markov models to classify monophonic melodies belonging to one of three different types of Western folk music (Austrian, German and Irish). They were able to achieve 63% accuracy in three-way classifications that used only melodic features. Interestingly, they found that the number of hidden states had only a relatively minor effect on success rates and that simple Markov models outperformed more complex models.

Ponce de Leon and Inesta (2002) produced a system that extracted and segmented monophonic jazz and classical MIDI tracks in order to extract melodic, harmonic and rhythmic features. The system then used these features to form distinguishable categories using self-organising maps. About 77% of the pieces were classified correctly as belonging to a group that corresponded roughly to jazz or a group that corresponded roughly to classical music.

Lartillot et al. (2001) discussed two alternative methods of unsupervised learning, namely an improved incremental parsing method and prediction suffix trees, for the purposes of classifying recordings based on musical style. This was done using analyses of musical sequences in terms of rhythm, melodic contour and polyphonic relationships.

Dannenberg, Thom and Watson (1997) described a real-time system to classify performance styles. Improvisations were classified as “lyrical,” “frantic,” “syncopated,” “pointillistic,” “blues,” “quote,” “high” and “low.” The system was trained with MIDI recordings of trumpet performances. The following features were extracted from the MIDI data: averages and standard deviations of MIDI key number, duration, duty factor (the ratio of duration to inter-onset interval), pitch (differs from key number in that Pitch Bend information is included) and volume, as well as counts of notes, Pitch Bend

messages and volume change messages. Bayesian, linear and neural network based classification schemes were used, with an optimum successful classification rate of 90.0% achieved among the eight style classes with the Bayesian classifier.

4.4 Music generation based on learned styles

Laine and Kuuskankare (1994) used genetic algorithms to generate melodies in different styles. In order to do this, they represented melodic segments with mathematical functions. This causes one to think that the inverse process, namely fitting melodies to functions, could be useful in detecting melodic characteristics and patterns for classification purposes.

Pachet (2002) proposed an interactive system capable of generating music in real-time in a style consistent with that being performed by humans or other systems. A Markov model was used to account for rhythm, beat, harmony and imprecision. Trees of sequences were used to learn musical patterns.

There are a number of other software systems that have been devised to generate music in a particular style or styles. The work of David Cope (1991a and 1991b) is particularly well known. Many of these systems have only limited relevance to this thesis, however, primarily because they use pre-written rules-based techniques rather than dynamic pattern recognition techniques. The generative technologies used in these systems are therefore not easily adaptable to musical analysis and supervised classification. A later publication by Cope (1996) does describe some very interesting automated analysis techniques that are more relevant to genre classification, but these still emphasize factors that are primarily relevant to Western art music.

5. Feature Library

5.1 Overview of issues relating to choice of features

People often claim that they "don't know what to listen to" when they are first exposed to an unfamiliar genre of music. This shows how difficult genre recognition can be in terms of feature extraction and how listening methodologies that apply to one genre may be of little value when applied to another. It is useful to consider as wide a range of features as possible in order to characterize as full a spectrum of music as possible. It was therefore decided to devise a large library of features relating to a number of different aspects of music as part of this thesis. Although it was known that not all of these features would ultimately be used, for reasons discussed in Section 3.2, this approach had the dual benefits of causing there to be a greater probability of finding features with feature selection techniques that have good discerning power that might not have been initially obvious as good candidates and of creating a resource that could be of use in future studies for a variety of purposes.

It is obvious that even people with little musical knowledge can make at least some genre classifications. It could therefore be argued that genre classification systems should pay special attention to features that are meaningful to the average, musically untrained listener, to the detriment of more technical or sophisticated musical properties. It is maintained here, however, that the ultimate goal of a classification system is to produce a correct classification, and whatever readily available features help to do this should be used, whether or not they are perceptible to the average human. The fact that high-level musical knowledge, such as precise perception and understanding of timing, pitch and voice information, is not necessary to distinguish between genres does not mean that it might not help. In addition, machine learning and pattern recognition systems operate using significantly different mechanisms than their human analogs, so there is no reason to assume that the types of percepts suited for one are necessarily suited to the other. Since high-level musical information is readily available from symbolic formats such as MIDI, it might as well be taken advantage of. High-level musical knowledge is, by definition, musical, and is therefore likely to be of use in distinguishing between genres. As shown in Chapter 4, existing genre classification systems have only had limited success to date without high-level musical features, and one of the goals of this thesis is to demonstrate the usefulness of such features

There is, of course, a great deal of existing literature on theoretical analyses of music that could be used to generate features. The books of Cook (1987) and LaRue (1992), to give just two of many possible examples, provide complementary surveys of analytical methods. Ideally, one would like to have a grand unified analytical process that could be

applied to music in any genre and used to generate features. Unfortunately, there is no generally accepted process of this sort. However, even though no single analysis system is complete, and most are only applicable to a limited number of genres, several systems could nonetheless be used in a complementary or parallel way. For example, Schenkerian analysis could be used to analyze harmony if other features indicate a significant degree of tonality in a recording, complimented perhaps by a set theory analysis to deal with non-tonal music. To extend the example, techniques such as those of Grosvenor Cooper and Leonard Meyer (1960) could also be used to analyze rhythm and melody and the techniques of Rudolph Reti (1951) could be used to gain insights by looking at motivic patterns. Semiotic analysis (Tarasti 2002) could also potentially be useful, although somewhat difficult to implement automatically from a content-based perspective. The multi-purpose analysis systems developed by researchers such as David Temperley (2001) in general could also be taken advantage of. Although these types of analyses are intrinsically different in many ways, they could each be used to generate individual features that could prove to be complementary.

There are, unfortunately, a number of disadvantages with using this approach of combining sophisticated analytical systems. To begin with, many of these techniques require a certain amount of intuitive subjective judgement, as the analytical rules are sometimes vague or ambiguous. This is demonstrated by common occurrences of inconsistencies between the analyses of the same piece by different people using the same system. Another problem is that sophisticated theoretical analyses could be computationally expensive, thus making their use inappropriate for rapidly expanding musical databases or real-time classification. In addition, most analysis techniques have been designed primarily from the perspective of Western art music, which limits their applicability to popular and non-Western musics. This last problem, however, may be less crippling than it seems, as analyses could potentially still be generated that are internally consistent, even if the meaning of the analyses themselves, from the perspective of their theoretical background, are not relevant to certain genres. Future experimentation is necessary to further investigate this.

In any event, a generally accepted software system capable of performing a wide range of sophisticated theoretical analyses has yet to be implemented, and this is well beyond the scope of this thesis. One must therefore make do with taking simple and incomplete concepts from different analytical systems, of necessity somewhat haphazardly, and combining them with intuitively derived characteristics in order to arrive at suitable features. This is not as serious a limitation as it might seem, as features that are used for classification purposes need not be consistent or meaningful in any overarching theoretical sense. All that matters for the purposes of classification is that each feature helps to distinguish between genres.

In any case, most humans are certainly unable to perform sophisticated theoretical analyses, but are nonetheless able to perform genre classifications. It is therefore clear that such analyses are not strictly necessary in order to implement a successful automatic classification system. Furthermore, a study of how well children of different ages can judge the similarities of music belonging to different styles found that there was almost no difference between the success rates of eleven-year olds compared to college sophomores, despite the fact that, unlike the sophomores, the eleven-year olds displayed almost no knowledge of musical theory or stylistic conventions (Gardner 1973). Of course, this does not necessarily mean that features based on sophisticated analytical systems might not be useful to an automated classification system, but it does appear that they are not necessary, which is fortunate given the difficulty that would be involved in extracting them.

Once it is accepted that features need not be chosen in some unified theoretical sense, one is tempted to program characteristics of particular genres into the system (e.g. swing rhythm, characteristic drum and bass rhythms, etc.) and have the computer base features on whether or not such characteristics are present in recordings. The disadvantage of this, however, is that it becomes necessary to base features on the particular genres that are being considered. Although this may be appropriate and effective for some specialized applications,[1] such an approach would be unable to adapt easily to changes in a classification taxonomy. It would be very undesirable to have a system that is dependant not only on people laboriously programming features specific to particular genres, but having to reprogram them all every time the classification taxonomy changes. This would be a particularly arduous task when one considers the difficulties in defining genres discussed in Chapter 2. Such a system would in essence be a type of expert system which, as discussed in Section 1.5, is undesirable for the purposes of musical genre classification. One must therefore use features that are meaningful in the context of a large variety of genres.

It was decided to pay special attention to simple features that well-trained humans are suspected to use in genre determinations. Such humans are currently the most skilled genre classifiers, so the features that they use are the most likely to be useful. This does not mean that other features were ignored, however, as there may be other important characteristics of music that humans are not consciously aware of, but which do delineate differences between genres.

[1] Those interested in such a system may wish to consult work such as that by Nettl (1990) and Manuel (1998) for an overview of the types of features that are characteristic of different types of non-classical music. The body of work on classical music is too large to cite specifically.

5.2 Ethnomusicological background relating to features

Once the issues discussed in Section 5.1 were considered, it was decided to research the work of musicologists and ethnomusicologists in order to search for features that might be of use. Ethnomusicologists have done significant research into comparing the musics of different cultures and, although this has often been done with anthropological interests in mind, this work is in some cases adaptable to the purposes of this thesis. Such research tends to focus on experimental observation rather than on attempting to derive theoretical meaning from music. It is for this reason that ethnomusicological research tends to be less likely to be intrinsically tied to specific types of music or limiting assumptions than more theoretical analytical approaches.

Perhaps the most extensive work in this vein was performed by Alan Lomax and his colleagues in the Cantometrics project (Lomax 1968). This project compared several thousand songs from hundreds of different cultural groups using thirty-seven features. These features were extracted by hand from audio recordings and, unfortunately, many of them are not extractable from MIDI recordings. These features are still discussed here, however, as they could be of significant use in future systems designed to work with audio recordings, particularly non-Western recordings. Also, a number of the Cantometrics features helped to inspire some of the features that were in fact used in this thesis. Below are the thirty-seven features proposed in the Cantometrics project:

1. Leader chorus: the importance of the lead singer relative to the chorus
2. Relation of orchestra to vocal part: importance and independence of the orchestra relative to the vocal part
3. Relation within orchestra: relative independence of the different parts of the orchestra
4. Choral musical organization: texture of the choral singing
5. Choral tonal integration: degree to which the chorus blends singing together to create perception of unity and resonance
6. Choral rhythmic organization: degree of rhythmic co-ordination of the chorus
7. Orchestral musical organization: texture of the orchestra
8. Orchestral tonal concert: degree to which the orchestra blends together to create perception of sonority
9. Orchestral rhythmic concert: degree of rhythmic co-ordination of the orchestra
10. Text part: whether singers tend to use words or other sounds. Also measures amount of repetition of text.
11. Vocal rhythm: complexity of meter used by singers
12. Vocal rhythmic organization: degree to which singers use polyrhythms
13. Orchestral rhythm: complexity of meter used by orchestra

14. Orchestral rhythmic organization: degree to which orchestra uses polyrhythms
15. Melodic shape: melodic contour of most characteristic phrases
16. Melodic form: complexity of form
17. Phrase length: temporal length of phrases
18. Number of phrases: average number of phrases occurring before full repeats
19. Position of final tone: position of the final pitch relative to the range of the song
20. Range of melody: pitch interval between the lowest and highest notes of the song
21. Average interval size: average melodic interval
22. Type of vocal polyphony: type of polyphony present, ranging from a drone to counterpoint
23. Embellishment: amount of embellishment used by the singer(s)
24. Tempo: speed of song from slow to very fast
25. Volume: loudness of song
26. Vocal rhythm: amount of rubato in the voice part
27. Orchestral rhythm: amount of rubato in the orchestral part
28. Glissando: degree to which voice(s) slide to and from notes
29. Melisma: number of pitches sung per syllable
30. Tremolo: amount of undulation on held notes
31. Glottal effect: amount of glottal activity present
32. Vocal register: whether singers are singing near the bottom, middle or top of their ranges
33. Vocal width and tension: degree to which voice sounds thin or rich
34. Nasalization: how nasal the singing sounds
35. Raspy: amount of raspiness in singing
36. Accent: strength of attack of sung tones
37. Consonants: precision of enunciation in singing

There would be a number of difficulties in incorporating these features into even an audio automatic classification system. Many of the features, such as “nasality,” are difficult to measure objectively or even extract at all automatically. Furthermore, many of the features are related to vocal lines, so segmentation would be necessary to separate the many lines that may be recorded in a single audio channel. Nonetheless, Lomax did find a good correlation between these features and cultural patterns, and they intuitively seem as if they might perform well, so the work necessary to extract these features may well be worth the effort in future research.

Feature 15 above refers to melodic contour. This is one area in which some significant research has been done, and is worth looking at in more detail. Charles Adams in particular has found that examining melodic contour can allow one to differentiate

between different musics (Adams 1976). Adams based his analyses on only the initial note (I), highest note (H), lowest note (L) and final note (F) of melodies.

There are, unfortunately, some complications in applying Adams' approach to MIDI recordings. To begin with, isolating the melodies of a piece can be difficult when dealing with music with multiple voices, as the melodies can involve notes contained in only one or in many voices. In the case of polyphonic music, one must deal with simultaneous melodies. One possible solution would be to simply assume that the most significant melody is in the highest line, and only this line, which is often but certainly not always the case. Alternatively, one could potentially implement some sort of melody detection system. An example of the last approach would be to assume that the melody is in the line that wins based on a weighted average of loudness and quantity of notes, with perhaps a bias given to the highest line.

An additional problem is that there can be repetitions of melodies or multiple independent melodies that occur sequentially. A melody/phrase segmentation system would be necessary to truly extract melodic contour features in the sense that they were intended, something which is beyond the scope of this thesis.

Fortunately, it is possible to extract at least some melodic contour features without critical problems caused by the lack of a melody segmentation system. Section 5.9 contains, among other things, a number of features inspired by Adams' work.

A number of writers have proposed a number of broad areas that could be useful to concentrate on for the purposes of musical classification. Nettl (1990) has proposed the following broad features as having significance to many different cultures:

1. Sound and singing style
2. Form
3. Polyphony (texture)
4. Rhythm and tempo
5. Melody and scale.

Julie E. Cumming has suggested a number of features in relation to motets (Cumming 1999). With the understanding that "voices" can be adapted to mean voices or instruments, a number of these features have a good deal of general applicability:

1. Texture
2. Number of voices
3. Voice ranges
4. Melodic behaviour (leaps, steps)
5. Relative speed of voices

6. Coordination of phrases between voices
7. Use of rests
8. Length
9. Complexity
10. Tone

Philip Tagg has proposed the following "checklist of parameters of musical expression" that could be adapted to generate features for both symbolic and audio systems:

1. *Aspects of time: duration of analysis object and relation of this to any other simultaneous forms of communication; duration of sections within the analysis object; pulse, tempo, meter, periodicity; rhythmic texture motifs.*
2. *Melodic aspects: register; pitch range; (melodic) motifs; tonal vocabulary; contour; timbre.*
3. *Orchestration aspects: type and number of voices, instruments, parts; technical aspects of performance; timbre; phrasing; accentuation.*
4. *Aspects of tonality and texture: tonal centre and type of tonality (if any); harmonic idiom; harmonic rhythm; type of harmonic change; chordal alteration; relationship between voices, parts, instruments; compositional texture and method.*
5. *Dynamic aspects: levels of sound strength; accentuation; audibility of parts.*
6. *Acoustical aspects: characteristics of (re-)performance 'venue'; degree of reverberation; distance between sound source and listener; simultaneous 'extraneous' sound.*
7. *Electromusical and mechanical aspects: panning, filtering, compressing, phasing, distortion, delay, mixing, etc.; muting, pizzicato, tongue flutter, etc.*

(Tagg 1982, 45-46).

Although this checklist was designed with theoretical analysis of Western music in mind, the ideas nonetheless have a more general applicability, as long as they are taken in conjunction with other features. Another useful list of parameters has been suggested by David Cope (1991b), although this list also emphasizes parameters specific to Western art music.

There are also a number of software systems that have been developed for problems other than genre classification but nonetheless contain a number of useful features. Aarden and Huron (2001), for example, have performed an interesting study where corresponding characteristics of European folk songs were studied in terms of spatial

location. A potentially useful catalogue of 60 high-level features was used, although these were limited to monophonic melodies. To give another example, Towsey et al. (2001) developed a system for compositional purposes that uses 21 high-level melodic features that have application beyond the aesthetic fitness judgements that they were used for in this study.

5.3 General comments on features in the following sections

The feature library that was actually devised for this thesis was created in the context of all of the research discussed in Sections 5.1 and 5.2 as well as the work that has previously been done in automatic genre analysis (see Chapter 4). The following seven sections provide details about these features. Although some of the features were based on features used in previous research, many of them are original, particularly in the context of computer-based analysis for the purposes of classification.

The ideology that accompanies any person's knowledge about music can cause them to give inappropriate weight to some characteristics of music, and ignore others. One will notice a bias towards Western tonal music in examining the features discussed in Sections 5.4 to 5.10. This is due partly to the training of the authour, partly to the dominance of Western genres in the taxonomy that was used and partly to the limitations of MIDI, or any other musical representation based on Western music. Nonetheless, efforts were made to include features that might not be obvious to one accustomed only to Western music. The library of features used in this thesis should be seen as a work in progress that can continually be expanded and refined as more types of music are considered and as music changes.

One will notice, upon examining the features listed in Sections 5.4 to 5.10, that there is a certain degree of redundancy in some of the features, in the sense that one feature sometimes emphasizes an aspect of another feature. This was done in order to ensure that features were available relating to both overviews of certain aspects of recordings as well as to more focused aspects that could be particularly salient. The feature library in the following sections is intended as a catalogue of features, some of which will be appropriate for certain tasks and some which will be appropriate for others. This catalogue was designed to give the feature selection and weighting algorithms as wide a range of features as possible to choose from and to provide a resource for future research. The redundancy in the features was purposely included for these reasons, and it is understood that no one classification system should be fed all of the features without some kind of feature selection process, as this would likely overwhelm it. Rather, this catalogue is intended to serve as a palette from which different classifiers can select different features according to their needs.

Efforts were made to design features that could be extracted from any symbolic musical representation, not just MIDI. So, with this in mind, as many as possible of the features proposed in Sections 5.4 to 5.10 were designed so that they could be easily adapted to music stored in any symbolic format, as long as pitch, rhythmic timing, dynamics, instrumentation and voice segregation are known reliably and exactly. Since MIDI was used for this thesis, however, the features described below use MIDI terminology so that the definitions could be more precise. Section 3.1 can be consulted by those needing more information on MIDI and the terminology related to it.

Unfortunately, there can be many different ways of encoding MIDI data. MIDI recordings can be produced by writing out music on a notation program or by performing actual real-time recordings, each of which can produce significantly differences in recordings of the same music. An authour's recording style and the particular sequencer he or she uses can also play a role. Care was therefore taken, when possible, to use features that were not sensitive to differences in the encoding style. Although this problem could have been avoided by only considering MIDI data from a single source, this was not done here, partly because there is no known source with a sufficiently diverse range of music, and partly because it was the goal of this system is to have a system capable of classifying recordings from arbitrary sources.

A number of intermediate representations of the MIDI recordings, including histogram-based representations, were constructed in order to derive some of the features proposed below. The most interesting of these representations are discussed in the following sections.

One will notice that there are a few "magic numbers" in the descriptions of some of the features. The origin of these constants is based on intuition and informal experimental experience.

Given their number, it was not possible to implement all of the features described in Sections 5.5 to 5.10. This was not a serious limitation, as the 111 features that were implemented represented a much greater number of features than have been used by most other symbolic music classification systems. The particular choice of which features to omit in the current implementation was based on a combination of the author's judgement of how useful each feature would be and how time-consuming it would be to implement, an important concern given the size and scope of the software. All of the features except for T-11, T-14, T-16, T-17, T-18, T-19, R-16, R-26, R-27, R-28, R-29, P-26, M-16, M-20 and C-1 to C-28 were implemented.

5.4 Features based on instrumentation

This class of features capitalizes on the fact that the General MIDI (level 1) specification allows recordings to make use of 128 pitched-instrument patches and a

further 47 percussion instruments in the Percussion Key Map. Although these instruments are insufficient for the full range of international music, they are, in general, diverse enough for the genres covered in this thesis.

The use of MIDI patches can be, in some cases, somewhat sensitive to encoding inconsistencies between different MIDI authours. In a few fortunately rare cases authours fail to specify patch numbers, with the result that all notes are played using a piano patch by default. Another problem is the occasional inconsistency in the choice of patches that are used for sung lines. Despite these occasional problems, however, features based on instrumentation can be highly characteristic of genres, and the use of features belonging to other classes can helped to counteract inconsistencies in authours' uses of patches.

The instrumentation related features that were implemented are as follows:

I-1 Pitched Instruments Present: A features array with one entry for each of the 128 General MIDI Instruments. Each entry was set to 1 if at least one note was played using that patch and to 0 if the patch was not used.

I-2 Unpitched Instruments Present: A features array with one entry for each of the 47 MIDI Percussion Key Map instruments. Each entry was set to 1 if at least one note was played using that patch and to 0 if the patch was not used.

I-3 Note Prevalence of Pitched Instruments: A features array with one entry for each of the 128 General MIDI Instruments. Each entry was set to the number of notes played using the corresponding MIDI patch divided by the total number of Note Ons in the piece.

I-4 Note Prevalence of Unpitched Instruments: A features array with one entry for each of the 47 MIDI Percussion Key Map instruments. Each entry was set to the number of notes played using the corresponding MIDI patch divided by the total number of Note Ons in the piece.

I-5 Time Prevalence of Pitched Instruments: A features array with one entry for each of the 128 General MIDI Instruments. Each entry was set to the total time in seconds during which a given instrument was sounding notes divided by the total length in seconds of the piece.

I-6 Variability of Note Prevalence of Pitched Instruments: Standard deviation of the fraction of notes played by each General MIDI instrument that is used to play at least one note.

I-7 Variability of Note Prevalence of Unpitched Instruments: Standard deviation of the fraction of notes played by each MIDI Percussion Key Map instrument that is used to play at least one note.

I-8 Number of Pitched Instruments: Total number of General MIDI patches that were used to play at least one note.

I-9 Number of Unpitched Instruments: Total number of MIDI Percussion Key Map patches that were used to play at least one note.

I-10 Percussion Prevalence: Total number of Note Ons belonging to percussion patches divided by total number of Note Ons in the recording.

I-11 String Keyboard Fraction: Fraction of Note Ons belonging to string keyboard patches (General MIDI patches 1 to 8).

I-12 Acoustic Guitar Fraction: Fraction of Note Ons belonging to acoustic guitar patches (General MIDI patches 25 and 26).

I-13 Electric Guitar Fraction: Fraction of Note Ons belonging to electric guitar patches (General MIDI patches 27 to 32).

I-14 Violin Fraction: Fraction of Note Ons belonging to the violin patches (General MIDI patches 41 or 111).

I-15 Saxophone Fraction: Fraction of Note Ons belonging to saxophone patches (General MIDI patches 65 to 68).

I-16 Brass Fraction: Fraction of Note Ons belonging to brass patches (including saxophones) (General MIDI patches 57 to 68).

I-17 Woodwinds Fraction: Fraction of Note Ons belonging to woodwind patches (General MIDI patches 69 to 76).

I-18 Orchestral Strings Fraction: Fraction of Note Ons belonging to orchestral string patches (General MIDI patches 41 to 47).

I-19 String Ensemble Fraction: Fraction of Note Ons belonging to orchestral string ensemble patches (General MIDI patches 49 to 52).

I-20 Electric Instrument Fraction: Fraction of Note Ons belonging to electric (non-"synth") patches (General MIDI patches 5, 6, 17, 19, 27 to 32, 34 to 40).

5.5 Features based on musical texture

This class of features takes advantage of the fact that MIDI notes can be assigned to different channels and to different tracks, thus making it possible to segregate the notes belonging to different voices. Although it would seem natural to use MIDI tracks to distinguish between voices, since only a maximum of sixteen channels are available, this was found to be an inappropriate approach. Using MIDI tracks would mean that it would be impossible to extract texture-based features from all Type 0 MIDI files, since they only allow a single track. Almost all MIDI files do use different channels for different voices, however, and it is possible to take advantage of Program Change messages to multiplex multiple voices onto a single channel in order to avoid being restricted to sixteen voices. It was therefore decided to use MIDI channels in order to distinguish between voices rather than tracks.

This approach is not perfect, as it is possible to use a single channel to hold multiple voices even without regular program change messages. A piano could be used to play a four-voice chorale, for example, with all notes occurring on one channel. This problem is unavoidable, unfortunately, as it would be necessary to design a special analysis module to automatically segregate voices in order to solve this problem, something which is beyond the scope of this thesis. Fortunately, this problem does not occur often.

The texture related features that were implemented are listed below:

T-1 Maximum Number of Independent Voices: Maximum number of different channels in which notes have sounded simultaneously.

T-2 Average Number of Independent Voices: Average number of different channels in which notes have sounded simultaneously. Rests are not included in this calculation.

T-3 Variability of Number of Independent Voices: Standard deviation of number of different channels in which notes have sounded simultaneously. Rests are not included in this calculation.

T-4 Voice Equality – Number of Notes: Standard deviation of the total number of Note Ons in each channel that contains at least one note.

T-5 Voice Equality – Note Duration: Standard deviation of the total duration of notes in seconds in each channel that contains at least one note.

T-6 Voice Equality – Dynamics: Standard deviation of the average volume of notes in each channel that contains at least one note.

T-7 Voice Equality – Melodic Leaps: Standard deviation of the average melodic leap in MIDI pitches for each channel that contains at least one note.

T-8 Voice Equality – Range: Standard deviation of the differences between the highest and lowest pitches in each channel that contains at least one note.

T-9 Importance of Loudest Voice: Difference between the average loudness of the loudest channel and the average loudness of the other channels that contain at least one note divided by 64.

T-10 Relative Range of Loudest Voice: Difference between the highest note and the lowest note played in the channel with the highest average loudness divided by the difference between the highest note and the lowest note in the piece.

T-11 Relative Range Isolation of Loudest Voice: Number of notes in the channel with the highest average loudness that fall outside the range of any other channel divided by the total number of notes in the channel with the highest average loudness.

T-12 Range of Highest Line: Difference between the highest note and the lowest note played in the channel with the highest average pitch divided by the difference between the highest note and the lowest note in the piece.

T-13 Relative Note Density of Highest Line: Number of Note Ons in the channel with the highest average pitch divided by the average number of Note Ons in all channels that contain at least one note.

T-14 Relative Note Durations of Lowest Line: Average duration of notes (in seconds) in the channel with the lowest average pitch divided by the average duration of notes in all channels that contain at least one note.

T-15 Melodic Intervals in Lowest Line: Average melodic interval in semitones of the line with the lowest average pitch divided by the average melodic interval of all lines that contain at least two notes.

T-16 Simultaneity: Average number of notes sounding simultaneously.

T-17 Variability Simultaneity: Standard deviation of number of notes sounding simultaneously.

T-18 Voice Overlap: Number of notes played within the range of another voice divided by total number of notes in the piece.

T-19 Parallel Motion: Fraction of all notes that move together within 10% of the duration of the shorter note that both move up or both move down.

T-20 Voice Separation: Average separation in semi-tones between the average pitches of consecutive channels (after sorting based on average pitch) that contain at least one note divided by 6.

5.6 Features based on rhythm

A number of scholars have expressed the view that rhythm plays a very important, or even dominant, role in many types of music. Richard Middleton (2000), for example, stresses the importance of rhythm in characterising music in a discussion of ways to approach creating a widely applicable method of music analysis. It is unfortunate that many music analysis techniques, with a few exceptions such as the work of Cooper and Meyer (1960), tend to give rhythm less attention than it deserves. Special attention was therefore given to this class of features in this thesis.

One approach to acquiring rhythmic features would be to use beat-tracking systems. Most existing beat-tracking systems provide only an estimate of the main beat and its strength, however, with little further information. More varied and detailed information is needed for the purposes of genre classification. The “beat histogram” approach used by Brown (1993) and by George Tzanetakis and his colleagues in a number of papers (Tzanetakis, Essl & Cook 2001; Tzanetakis & Cook 2002; Tzanetakis 2002) has been shown to be a valuable resource in this respect. A slightly modified version of

Tzanetakis' histogram was used to derive a number of the rhythmic features listed in this section.

It is necessary to have some understanding of how autocorrelation works in order to understand how beat histograms are constructed. Autocorrelation essentially involves comparing a signal with versions of itself delayed by successive intervals. This technique is used to find repeating patterns in signals. Autocorrelation gives the relative strength of different periodicities within a signal. In terms of musical data, autocorrelation allows one to find the relative strength of different rhythmic pulses.

In the particular case of this thesis, rhythmic histograms were constructed by considering sequences of MIDI events with MIDI ticks delineating the time domain. The following autocorrelation function was applied to each sequence of MIDI Note On messages:

$$autocorrelation[lag] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]x[n-lag] \tag{10}$$

where n is the input sample index (in MIDI ticks), N is the total number of MIDI ticks, x is the sequence of MIDI ticks and *lag* is the delay in MIDI ticks ($0 \leq \text{lag} < \text{N}$). The value of $x[n]$ is proportional to the velocity of Note Ons. This ensures that beats are weighted based on the strength with which notes are played. This autocorrelation function was applied repeatedly to each MIDI sequence with different values of lag. These lag values corresponded to both rhythmic periodicities as well as bin labels in beat histograms, and the *autocorrelation* value provided the magnitude value for each bin.

Once the histogram was completed with all reasonable values of *lag*, the histogram was downsampled and transformed so that each bin corresponded a periodicity with units of beats per minute. Finally, the histogram was normalized so that different pieces could be compared. The end result was a histogram whose bins corresponded to rhythmic pulses with units of beats per minute and whose bin frequencies indicated the relative strength of each pulse. In effect, beat histograms portray the relative strength of different beats and sub-beats within pieces.

Figure 1 displays a sample beat histogram derived from a MIDI recording of the Ramones' *Blitzkrieg Pop*. The clear periodicities that are often multiples of each other is typical of Punk music, as is the characteristic rhythmic looseness demonstrated by the spread around each beat. Other types of music demonstrated very different patterns in their beat histograms. Techno, for example, often had very clearly defined beats, without the surrounding spread. Modern Classical music, to cite another example, often had much less clearly defined beats.

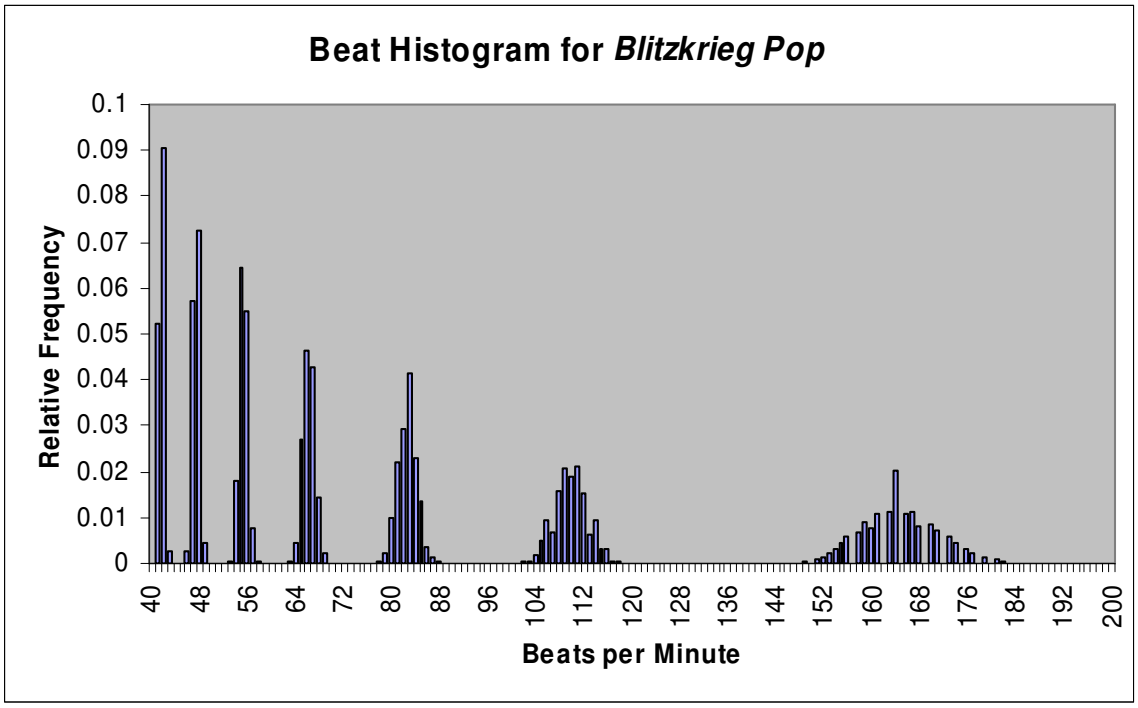

Figure 1: Beat histogram for the Ramones' *Blitzkrieg Pop*.

Although beat histograms have only limited utility as a features in and of themselves, they are very useful in providing an intermediate data structure from which other features can be extracted. The two highest peaks of the beat histograms tend to have particular importance, as they are likely to represent the main beat of the music or one of its multiples or factors.

It is important to keep in mind that MIDI timing can be affected by both the number of MIDI ticks that go by and tempo change meta-events that control the rate at which MIDI ticks go by. Tempo change meta-events must therefore be monitored.

MIDI allows one to think of timing in terms of both raw time and rhythmic note values (i.e. half notes, quarter notes, etc.) by equating a certain number of ticks with a quarter note (although live MIDI recordings are not always quantized). Although these rhythmic note values, along with time signature and tempo change meta-events, can potentially provide features with a high discriminating power, they are somewhat sensitive to the MIDI encoding style of MIDI files' authours and to the sequencers they've used. This inconsistency is the reason that an emphasis is put on features derived from rhythmic histograms in the rhythmic features listed in this section.

The rhythmic features that were implemented are listed below:

R-1 Strongest Rhythmic Pulse: Bin label of the beat bin with the highest magnitude.

R-2 Second Strongest Rhythmic Pulse: Bin label of the beat bin of the peak with the second highest magnitude.

R-3 Harmonicity of Two Strongest Rhythmic Pulses: The bin label of the higher (in terms of bin label) of the two beat bins of the peaks with the highest magnitude divided by the bin label of the lower.

R-4 Strength of Strongest Rhythmic Pulse: Magnitude of the beat bin with the highest magnitude.

R-5 Strength of Second Strongest Rhythmic Pulse: Magnitude of the beat bin of the peak with the second highest magnitude.

R-6 Strength Ratio of Two Strongest Rhythmic Pulses: The magnitude of the higher (in terms of magnitude) of the two beat bins corresponding to the peaks with the highest magnitude divided by the magnitude of the lower.

R-7 Combined Strength of Two Strongest Rhythmic Pulses: The sum of the frequencies of the two beat bins of the peaks with the highest frequencies.

R-8 Number of Strong Pulses: Number of beat peaks with normalized frequencies over 0.1.

R-9 Number of Moderate Pulses: Number of beat peaks with normalized frequencies over 0.01.

R-10 Number of Relatively Strong Pulses: Number of beat peaks with frequencies at least 30% as high as the magnitude of the bin with the highest magnitude.

R-11 Rhythmic Looseness: Average width of beat histogram peaks (in beats per minute). Width is measured for all peaks with frequencies at least 30% as high as the highest peak, and is defined by the distance between the points on the peak in question that are 30% of the height of the peak.

R-12 Polyrhythms: Number of beat peaks with frequencies at least 30% of the highest magnitude whose bin labels are not integer multiples or factors (using only multipliers of 1, 2, 3, 4, 6 and 8) (with an accepted error of +/- 3 bins) of the bin label of the peak with the highest magnitude. This number is then divided by the total number of beat bins with frequencies over 30% of the highest magnitude.

R-13 Rhythmic Variability: Standard deviation of the bin values (except the first 40 empty ones).

R-14 Beat Histogram: A feature array with entries corresponding to the magnitude values of each of the bins of the beat histogram (except the first 40 empty ones).

R-15 Note Density: Average number of notes per second.

R-16 Note Density Variability: The recording is broken into 5 second long windows. The note density is calculated for each. This feature is the standard deviation of these note densities.

R-17 Average Note Duration: Average duration of notes in seconds.

R-18 Variability of Note Duration: Standard deviation of note durations in seconds.

R-19 Maximum Note Duration: Duration of the longest note (in seconds).

R-20 Minimum Note Duration: Duration of the shortest note (in seconds).

R-21 Staccato Incidence: Number of notes with durations of less than a 10^{th} of a second divided by the total number of notes in the recording.

R-22 Average Time Between Attacks: Average time in seconds between Note On events (irregardless of channel).

R-23 Variability of Time Between Attacks: Standard deviation of the times, in seconds, between Note On events (irregardless of channel).

R-24 Average Time Between Attacks For Each Voice: Average of average time in seconds between Note On events on individual channels that contain at least one note.

R-25 Average Variability of Time Between Attacks For Each Voice: Average standard deviation, in seconds, of time between Note On events on individual channels that contain at least one note.

R-26 Incidence of Complete Rests: Total amount of time in seconds in which no notes are sounding on any channel divided by the total length of the recording.

R-27 Maximum Complete Rest Duration: Maximum amount of time in seconds in which no notes are sounding on any channel.

R-28 Average Rest Duration Per Each Voice: Average, in seconds, of the average amounts of time in each channel in which no note is sounding (counting only channels with at least one note) divided by the total duration of the recording.

R-29 Average Variability of Rest Durations Across Voices: Standard deviation, in seconds, of the average amounts of time in each channel in which no note is sounding (counting only channels with at least one note).

R-30 Initial Tempo: Tempo in beats per minute at the start of a recording.

R-31 Initial Time Signature: A feature array with two elements. The first is the numerator of the first occurring time signature and the second is the denominator of the first occurring time signature. Both are set to 0 if no time signature is present.

R-32 Compound Or Simple Meter: Set to 1 if the initial meter is compound (numerator of time signature is greater than or equal to 6 and is evenly divisible by 3) and to 0 if it is simple (if the above condition is not fulfilled).

R-33 Triple Meter: Set to 1 if numerator of initial time signature is 3, set to 0 otherwise.

R-34 Quintuple Meter: Set to 1 if numerator of initial time signature is 5, set to 0 otherwise.

R-35 Changes of Meter: Set to 1 if the time signature is changed one or more times during the recording.

5.7 Features based on dynamics

The term "loudness" is used in this thesis to refer to velocity values scaled by volume channel messages:

note loudness = note velocity x (channel volume / 127) (11)

All features based on dynamics use relative measures rather than absolute measures (such as average volume) because the default volume and velocity values set by sequencers can vary, and many MIDI authours simply encode their files without varying these values.

The features related to dynamics that were implemented are listed below:

D-1 Overall Dynamic Range: The maximum loudness minus the minimum loudness value.

D-2 Variation of Dynamics: Standard deviation of loudness levels of all notes.

D-3 Variation of Dynamics In Each Voice: The average of the standard deviations of loudness levels within each channel that contains at least one note.

D-4 Average Note To Note Dynamics Change: Average change of loudness from one note to the next note in the same channel.

5.8 Features based on pitch statistics

Statistics based on pitch can help to characterize genres in terms of degree of tonality, types of scales used and pitch variety. The features in this section differ from those in Sections 5.9 and 5.10 in that the latter take into account temporal locations of notes, whereas these features only consider recordings as a whole. It should be mentioned that all notes occurring on channel 10 were ignored for all of these features, as pitch values on that channel correspond to percussion patches, not to pitches.

Some features of this class were based on MIDI Pitch Bends. Although the use of Pitch Bends is somewhat variable from authour to authour, and therefore not entirely dependant on the music itself, features relating to Pitch Bend have the potential to be very discriminating, so they were included here. Efforts were made to use features with as limited sensitivity to non-musical factors as much as possible.

Slightly modified versions of the three types of "pitch histograms" that were implemented by George Tzanetakis and his colleagues (Tzanetakis & Cook 2002; Tzanetakis, Ermolinskyi & Cook 2002; Tzanetakis 2002) were used as bases from which pitch-based features could be derived. The first histogram, called the "basic pitch histogram," consisted of 128 bins, one for each MIDI pitch. The magnitude of each bin

corresponded to the number of times that Note Ons occurred at that particular pitch. This histogram gave insights into the range and spread of notes.

The second histogram was called the "pitch class histogram," and had one bin for each of the twelve pitch classes. The magnitude of each bin corresponded to the number of times Note Ons occurred in a recording for a particular pitch class. Enharmonic equivalents were assigned the same pitch class number. This histogram gave insights into the types of scales used and the amount of transposition that was present.

Finally, the "fifths pitch histogram," also with twelve bins, was generated by reordering the bins of the pitch class histogram so that adjacent bins were separated by a perfect fifth rather than a semi-tone. This was done using the following equation:

$$\beta = (7\alpha) \bmod (12) \tag{12}$$

where β is the fifths pitch histogram bin and α is the corresponding pitch class histogram bin. The number seven is used because this is the number of semi-tones in a perfect fifth, and the number twelve is used because there are twelve pitch classes in total. This histogram was useful for measuring dominant tonic relationships and for looking at types of transpositions.

All three histograms were normalized after being generated so that histograms would not be influenced by the lengths or note densities of recordings. The features based on pitch statistics that were implemented are listed below:

P-1 **Most Common Pitch Prevalence:** Fraction of Note Ons corresponding to the most common pitch.

P-2 **Most Common Pitch Class Prevalence:** Fraction of Note Ons corresponding to the most common pitch class.

P-3 **Relative Strength of Top Pitches:** The magnitude of the 2nd most common pitch divided by the magnitude of the most common pitch.

P-4 **Relative Strength of Top Pitch Classes:** The magnitude of the 2nd most common pitch class divided by the magnitude of the most common pitch class.

P-5 **Interval Between Strongest Pitches:** Absolute value of the difference between the pitches of the two most common MIDI pitches.

P-6 **Interval Between Strongest Pitch Classes:** Absolute value of the difference between the pitches of the two most common pitch classes.

P-7 **Number of Common Pitches:** Number of pitches that account individually for at least 9% of all notes.

P-8 **Pitch Variety:** Number of pitches used at least once.

P-9 **Pitch Class Variety:** Number of pitch classes used at least once.

P-10 Range: Difference between highest and lowest pitches.

P-11 Most Common Pitch: Bin label of the most common pitch divided by the number of possible pitches.

P-12 Primary Register: Average MIDI pitch.

P-13 Importance of Bass Register: Fraction of Note Ons between MIDI pitches 0 and 54.

P-14 Importance of Middle Register: Fraction of Note Ons between MIDI pitches 55 and 72.

P-15 Importance of High Register: Fraction of Note Ons between MIDI pitches 73 and 127.

P-16 Most Common Pitch Class: Bin label of the most common pitch class.

P-17 Dominant Spread: Largest number of consecutive pitch classes separated by perfect 5ths that accounted for at least 9% each of the notes.

P-18 Strong Tonal Centres: Number of peaks in the fifths pitch histogram that each account for at least 9% of all Note Ons.

P-19 Basic Pitch Histogram: A features array with bins corresponding to the values of the basic pitch histogram.

P-20 Pitch Class Distribution: A feature array with 12 entries where the first holds the magnitude of the bin of the pitch class histogram with the highest magnitude, and the following entries holding the successive bins of the histogram, wrapping around if necessary.

P-21 Fifths Pitch Histogram: A feature array with bins corresponding to the values of the 5ths pitch class histogram.

P-22 Quality: Set to 0 if the key signature indicates that a recording is major, set to 1 if it indicates that it is minor and set to 0 if key signature is unknown.

P-23 Glissando Prevalence: Number of Note Ons that have at least one MIDI Pitch Bend associated with them divided by total number of pitched Note Ons.

P-24 Average Range of Glissandos: Average range of Pitch Bends, where range is defined as the greatest value of the absolute difference between 64 and the second data byte of all MIDI Pitch Bend messages falling between the Note On and Note Off messages of any note.

P-25 Vibrato Prevalence: Number of notes for which Pitch Bend messages change direction at least twice divided by total number of notes that have Pitch Bend messages associated with them.

P-26 Prevalence of Micro-Tones: Number of Note Ons that are preceded by isolated Pitch Bend messages as a fraction of total number of Note Ons.

5.9 Features based on melody

Although the pitch statistics discussed in Section 5.8 are both meaningful and useful, they do not reflect any information relating to the order in which pitches are played. Neglecting information about sequence would be very limiting, as melody is a very important part of how many humans hear and think about music. Ideally, one would like to collect information about all of the melodies in a recording and how they repeat, change and interact with each other. The literature on melodic contour discussed in Section 5.2 could prove useful as well. Unfortunately, all of this would require a phrase segregation system that is beyond the scope of this thesis.

What one can do fairly easily, however, is collect statistics about melodic motion and intervals. In order to do this, the use of a "melodic interval histogram" is proposed here. Each bin of such a histogram is labelled with a number indicating the number of semi-tones separating sequentially adjacent notes in a given channel (independently of direction of melodic motion). In the normalized implementation of this histogram used here, the magnitude of each bin indicates the fraction of all melodic intervals that correspond to the melodic interval of the given bin. All notes occurring in any given channel were treated as a melody. Although this was not a perfect solution, especially for instruments such as pianos that can play harmonies or multiple melodies simultaneously, it was the only apparent cheaply available solution, and is entirely suitable in many cases. A second intermediate data structures was used as well. This consisted of an array with each indice corresponding to a MIDI channel and each entry consisting of a list of all melodic intervals on the appropriate channel in the order that they occurred. The intervals in the second data structure were negative for downward motion and positive for upwards motion.

The features based on melody that were implemented are listed below:

M-1 Melodic Interval Histogram: A features array with bins corresponding to the values of the melodic interval histogram.

M-2 Average Melodic Interval: Average melodic interval.

M-3 Most Common Melodic Interval: Melodic interval with the highest magnitude.

M-4 Distance Between Most Common Melodic Intervals: Absolute value of the difference between the most common melodic interval and the second most common melodic interval.

M-5 Most Common Melodic Interval Prevalence: Fraction of melodic intervals that belong to the most common interval.

M-6 Relative Strength of Most Common Intervals: Fraction of melodic intervals that belong to the second most common interval divided by the fraction of melodic intervals belonging to the most common interval.

M-7 Number of Common Melodic Intervals: Number of melodic intervals that represent at least 9% of all melodic intervals.

M-8 Amount of Arpeggiation: Fraction of horizontal intervals that are repeated notes, minor thirds, major thirds, perfect fifths, minor sevenths, major sevenths, octaves, minor tenths or major tenths.

M-9 Repeated Notes: Fraction of notes that are repeated melodically.

M-10 Chromatic Motion: Fraction of melodic intervals corresponding to a semi-tone.

M-11 Stepwise Motion: Fraction of melodic intervals that corresponded to a minor or major third.

M-12 Melodic Thirds: Fraction of melodic intervals that are major or minor thirds.

M-13 Melodic Fifths: Fraction of melodic intervals that are perfect fifths.

M-14 Melodic Tritones: Fraction of melodic intervals that are tritones.

M-15 Melodic Octaves: Fraction of melodic intervals that are octaves.

M-16 Embellishment: Fraction of notes that are surrounded on both sides by Note Ons the same channel that have durations at least 3 times as long as the central note.

M-17 Direction of Motion: Fraction of melodic intervals that are rising rather than falling.

M-18 Duration of Melodic Arcs: Average number of notes that separate melodic peaks and troughs in any channel.

M-19 Size of Melodic Arcs: Average melodic interval separating the top note of melodic peaks and the bottom note of melodic troughs.

M-20 Melodic Pitch Variety: Average number of notes that go by in a channel before a note is repeated. Notes that do not recur after 16 notes are not counted.

5.10 Features based on chords

This class of features is based on the intervals between notes that sound simultaneously. Although it is certainly not assumed that any recording is tonal, a number of features were used that are related to tonality. This was done simply because tonal relationships do play an important role in many of the genres that were considered in this thesis, and the degree and types of tonality present can be representative of genre. Some of the techniques for chord analysis discussed by Rowe (2001) were taken advantage of here.

Two new types of histograms are proposed as an aid to deriving chordal features. The first, called a “vertical interval histogram,” consists of bins labelled with different vertical intervals. The magnitude of each bin is found by going through MIDI recordings tick by tick and recording all vertical intervals (exhaustively between all notes sounding

simultaneously) that are sounding at each tick. The magnitude of each bin is then set to the appropriate sum and the bins are normalized.

This histogram does not, however, give explict insights into what kinds of chords are present at any given time. A “chord type histogram” is proposed in order to fill in this gap. This histogram has bins labelled with types of chords (two pitch class chords, major triad, minor triad, other triad, diminished, augmented, dominant seventh, major seventh, minor seventh, other chord with four pitch classes and chord with more than four pitch classes). All inversions were treated as equivalent and octave doubling was ignored. The frequencies were counted in much the same way as in the vertical interval histogram, and were normalized as well.

Neither of these histograms provides any information about arpeggiation, unfortunately, but some information related to this is collected in the melodic features. A more sophisticated system in the future could integrate vertical statistics with arpeggios and could collect information about inversions as well as chord transitions in order to obtain details about chord progressions, whether they be tonal or not. This is, however, beyond the scope of this thesis.

The following features were implemented in order to collect information relating to chords:

C-1 Vertical Intervals: A feature set consisting of the frequencies of each of the bins in the vertical interval histogram described above.

C-2 Chord Types: A feature set consisting of the frequencies of each of the bins in the chord typed histogram discussed above.

C-3 Most Common Vertical Interval: The bin label of the vertical interval histogram bin with the highest magnitude.

C-4 Second Most Common Vertical Interval: The bin label of the vertical interval histogram bin with the second highest magnitude.

C-5 Distance Between Two Most Common Vertical Intervals: The difference between the bin labels of the two most common vertical intervals.

C-6 Prevalence of Most Common Vertical Interval: The fraction of vertical intervals corresponding to the most common vertical interval.

C-7 Prevalence of Second Most Common Vertical Interval: The fraction of vertical intervals corresponding to the second most common vertical interval.

C-8 Ratio of Prevalence of Two Most Common Vertical Intervals: The fraction of vertical intervals corresponding to the second most common vertical interval divided by the fraction of vertical intervals corresponding to the most common vertical interval.

C-9 Average Number of Simultaneous Pitch Classes: Average number of different pitch classes sounding simultaneously.

C-10 Variability of Number of Simultaneous Pitch Classes: Standard deviation of the number of different pitch classes sounding simultaneously.

C-11 Minor Major Ratio: Number of minor vertical intervals divided by number of major vertical intervals.

C-12 Perfect Vertical Intervals: Fraction of all vertical intervals corresponding to perfect intervals.

C-13 Unisons: Fraction of all vertical intervals corresponding to unisons.

C-14 Vertical Minor Seconds: Fraction of all vertical intervals corresponding to minor seconds.

C-15 Vertical Thirds: Fraction of all vertical intervals corresponding to thirds.

C-16 Vertical Fifths: Fraction of all vertical intervals corresponding to fifths.

C-17 Vertical Tritones: Fraction of all vertical intervals corresponding to tritones.

C-18 Vertical Octaves: Fraction of all vertical intervals corresponding to octaves.

C-19 Vertical Dissonance Ratio: Total number of vertical 2nds, tritones, 7ths and 9ths divided be total number of vertical unisons, 4ths, 5ths, 6ths, octaves and 10ths.

C-20 Partial Chords: Fraction of all vertical intervals involving only two pitch classes.

C-21 Minor Major Triad Ratio: Number of minor triads divided by number of major triads.

C-22 Standard Triads: Fraction of all chords that are either major or minor triads.

C-23 Diminished and Augmented Triads: Fraction of all chords that are either diminished or augmented triads.

C-24 Dominant Seventh Chords: Fraction of all chords that are dominant sevenths.

C-25 Seventh Chords: Fraction of all chords that are dominant seventh, major seventh or minor seventh chords.

C-26 Complex Chords: Fraction of all chords that contain more that four pitch classes.

C-27 Non-Standard Chords: Fraction of all chords that are not two pitch class chords, not major or minor triads and not seventh chords.

C-28 Chord Duration: The average duration of a chord in seconds.

6. Implementation of classification system

6.1 Selection of model genre taxonomy

This section discusses the model genre taxonomies that were used to train and test the classification system. Chapter 2 can be consulted in order to read more about the background that led to the implementation decisions presented here.

It is important for a classification system to be able to classify recordings into categories that are meaningful to the average, potentially musically illiterate person. At the same time, however, it is also desirable that a system be able to make the kind of fine classifications that are useful to music professionals. A hierarchal tree-based structure of categories was chosen as the taxonomical structure to use for this system, as it fulfils these dual requirements. Broad categories, such as Classical or Jazz, are found at the root of the tree (topmost level of the tree), and categories become increasingly fine as one progresses towards the leaves (i.e. nodes in the tree without children).

The various branches of the tree were permitted to vary in terms of both depth and breadth. This was necessary in order to accommodate the different degrees to which different real-life genres are sometimes split into narrow sub-genres and sometimes simply left as broad categories.

It was decided to use a taxonomy based on individual recordings rather than artists as a whole, despite the problems related to scalability discussed in Chapter 2. Using artists would have involved too many contradictions that could confuse the classification system. Many artists have produced music in a number of different genres, and it would be inappropriate to attempt to force a genre recognition system to accept artificial relations between such pieces. This is not to say that relations based on artist are not meaningful, of course, since it would certainly be useful to build a system that could search for features based on a composer's style rather than genre, for example, but this is beyond the scope of this thesis.

The tree-based system used here had two important differences from the types of trees traditionally used: a given recording could be associated with more than one leaf genre and a sub-genre could be a direct descendant of more than one parent genre (e.g. Bossa Nova is a descendant of both Jazz and World Beat in the implementation shown in Figure 3). These two modifications did complicate the organizational clarity offered by traditional trees, but they were necessary to deal with the realties that the boundaries between different genres are often vague, sub-genres are often the result of a complex amalgamation of potentially disparate parent genres and many recordings do not fall unambiguously into single genre categories.

This hierarchical organization allows users to look at whatever level of the hierarchy is appropriate for their needs. Users can start at a root-level genre and descend to increasingly deep levels of the tree if they wish to refine a search. Alternatively, they can start at a leaf and travel up the tree if they wish to gain a broader perspective. Dividing genres hierarchically is not only advantageous in the sense that it is useful to humans performing searches, but it also makes it possible to compare how well the system distinguishes between fairly dissimilar music from the parent genres compared to the more similar sub-genres.

This kind of structure does have the disadvantage that even small updates to the taxonomy (which would be necessary in order to keep up-to-date with the constant changes in the labels that people use in real life) would require re-training of the entire classification system. This is not as much of a problem as one might think, however. One needs simply update the hierarchy and the training examples, and the learning of the new structure could be done automatically off-line with no further human intervention other than some test validation at the end of the learning process. This updating process could be done regularly as a matter of routine. The requirement of having to manually research and implement changes in the hierarchy is certainly inconvenient, but this is still a great improvement over the fully manual classification system that is currently in use. The use of data mining techniques such as those discussed in Chapter 2 could potentially be used to automate even these tasks in a future version of this software.

As can be seen in Chapter 4, existing automatic classification systems have rarely classified between more than 9 or 10 categories, and have frequently used fewer categories. This is certainly a reasonable choice as an intermediate attainable goal in developing genre classification systems. Taxonomies of this size are also realistic for certain limited types of tasks, and the use of broad categories avoids the pitfalls related to being forced to label training and testing recordings with narrower categories that tend to be more subjective. Of course, one must have a diverse enough set of training samples to represent all of the sub-types of broad genres if one wishes to perform realistic classifications.

In any case, it was decided to design an initial rough taxonomy (Figure 2) in order to compare the performance of this system with existing studies. Although these comparisons cannot be definitive, since different categories and different recordings have been used in different studies, and most of the existing studies have analyzed audio recordings rather than symbolic recordings, they can at least give some general idea of relative performance. The taxonomy shown is Figure 2 was designed to include categories with some similarity as well as categories that are significantly different.

Jazz
- Bebop
- Jazz Soul
- Swing

Popular
- Rap
- Punk
- Country

Western Classical
- Baroque
- Modern Classical
- Romantic

Figure 2: Reduced classification taxonomy.

Of course, the taxonomy shown in Figure 2 is not sophisticated or large enough to be useful for general real-life purposes. It was therefore decided to develop a much larger and better designed taxonomy in order to test the practical potential of the system developed here. This expanded taxonomy is shown in Figure 3.

Country
- Bluegrass
- Contemporary
- Trad. Country

Jazz
- *Bop*
  - Bebop
  - Cool
- *Fusion*
  - Bossa Nova
  - Jazz Soul
  - Smooth Jazz
- Ragtime
- Swing

Modern Pop
- Adult Contemp.
- *Dance*
  - Dance Pop
  - Pop Rap
  - Techno
- Smooth Jazz

Rap
- Hardcore Rap
- Pop Rap

Rhythm and Blues
- *Blues*
  - Blues Rock
  - Chicago Blues
  - Country Blues
  - Soul Blues
- Funk
- Jazz Soul
- Rock and Roll
- Soul

Rock
- *Classic Rock*
  - Blues Rock
  - Hard Rock
  - Psychedelic
- *Modern Rock*
  - Alternative Rock
  - Hard Rock
  - Metal
  - Punk

Western Classical
- Baroque
- Classical
- *Early Music*
  - Medieval
  - Renaissance
- Modern Classical
- Romantic

Western Folk
- Bluegrass
- Celtic
- Country Blues
- Flamenco

Worldbeat
- *Latin*
  - Bossa Nova
  - Salsa
  - Tango
- Reggae

Figure 3: Full classification taxonomy.

The desire to have a realistic taxonomy required significantly more thought than the taxonomy shown in Figure 2. As discussed in Chapter 2, on-line retailers tend to offer the most reliable and useful taxonomies. The expanded taxonomy developed here therefore emphasized this source of information, but were also made use of an amalgamation of information found in scholarly writings on popular music, popular music magazines, music critic reviews, taxonomies used by music vendors, schedules of radio and video specialty shows, fan web sites and the personal knowledge of the authour.

Particular use was made of the *All Music Guide,* an excellent on-line resource, and of the *Amazon.com* on-line store. These sites are widely used by people with many different musical backgrounds, so their systems are perhaps the best representations available of the types of genres that people actually use. These two sites are also complimentary, in a sense. The *All Music Guide* is quite informative and well researched, but does not establish clear relationships between genres. *Amazon.com*, in contrast, has clearly structured genre categories, but no informative explanations of what they mean.

Given the limitations on the number and types of MIDI files that can and have been encoded using MIDI and made available on-line, it was unfortunately only practical to use a subset of the categories that would have ideally been included in the taxonomy. The amount of time needed to manually find, download and classify recording also imposed limitations on the number of categories that could be used here. This taxonomy is, however, significantly larger, more specialized and more diverse than that used in any other known automatic classification system to date, and it includes a range of categories that is certainly sufficient for real-life, practical use.

The taxonomy shown in Figure 3 is not always perfectly logical or consistent. This was necessary in order to test the system realistically, as the types of genre structures that humans actually use are often illogical and inconsistent. The taxonomy proposed here is not presented as ideal, perfect or complete, but rather as a realistic taxonomy that is good enough for testing and for practical use, yet is certainly open to refinement in the future. This taxonomy encapsulates many of the difficulties, ambiguities and inconsistencies found in any realistic taxonomy and is large and sophisticated enough that it provides a significantly more difficult and realistic test bed than has been applied to any previous automatic genre classification system known to the author.

There are a total of 9 root level labels, 8 intermediate labels and 38 unique leaf labels in the taxonomy shown in Figure 3, for a total of 55 unique labels (with duplicates only counted once). Detailed explanations of these categories can, in most cases, be found by referencing the *All Music Guide.*

The software developed for this thesis has been designed to allow users to enter in their own taxonomies if they wish by customizing labels, modifying the architecture of the tree and controlling the selection and model classification of the training data if they wish. This makes it possible for users with specialized interests to custom train the software to be able to deal with the specific genres and sub-genres that interest them. Since there is not, a universally accepted genre taxonomy to date, this approach allows individual users to use taxonomies that fit their needs and perspectives. Furthermore, it allows modification of the taxonomy as genres change or new genres are introduced.

6.2 Selection of training and testing data

One would ideally like to have a standardized test bed of recordings that could be used to compare the performances of different classification systems. Unfortunately, no such large scale, widely acceptable and affordable set of MIDI recordings exists, so it was necessary to manually collect a custom recording library in order to train and test the classification system developed in this thesis.

The first step towards accomplishing this was the compilation of a catalogue of web sites from which MIDI files could be downloaded. Although an emphasis was placed on sites that focused on particular genres, they were insufficient in number to meet the demands of this thesis, so a number of general purpose sites and MIDI search engines were included as well.

The sites in this directory were then surveyed manually and all files that sounded like they might belong to the genre categories in the model taxonomy were downloaded. This recording library was then supplemented by specific prototypical recordings that were sought out if they had not already been found. This was done by constructing lists of ten to twenty pieces that were typical of each genre according to the *All Music Guide* web site and using MIDI search engines to find as many of these specific recordings as possible. At this point, 30 to 45 MIDI recordings were available for each leaf genre. Deficits in any individual genres were remedied by performing further searches for model recordings.

It was decided to use a combination of both prototypical examples and generally selected examples in this manner so as to attain a training set that would help to make it clear to the pattern recognition system what is typical of particular genres while at the same time including diverse enough examples to avoid overspecialization. Using only prototypical examples would have the dangers of biasing the system towards the authour's perception of genre categories and of making it unable to deal with recordings that are somewhat atypical of genres that they nonetheless clearly belong to.

It should be noted that, within each leaf category, MIDI files were taken from a variety of sources whenever possible. This was done in order to even out any encoding particularities, as recordings from a single source could have encoding characteristics that the system could use to classify them, which would artificially inflate classification performance.

All of the downloaded files were then reviewed one by one by the authour, and classified based on the author’s experience, the *All Music Guide* and the label of the piece on the site that it was downloaded from, if available. Twenty-five pieces were then selected from the available pool for each leaf genre. Single pieces were permitted to belong to more than one genre, when appropriate. The particular twenty-five recordings that were chosen were selected in order to fulfill three requirements: that all recordings could reasonably be said to belong to the given leaf genre, that as full a range of sub-types

within each leaf genre as possible were represented and that a few pieces at least somewhat atypical of the genre were present. These requirements helped to ensure that training would occur over a broad enough range that the classification system could perform well in a real-world situation and that success rates were not artificially inflated by using overly specialized training and testing examples.

The particular ceiling of twenty-five recordings per leaf genre was selected because of the time requirements involved in manually finding, downloading and classifying recordings, particularly given the number of genres considered here. An additional problem was that MIDI files are much harder to find in some genres than others. It was decided to use an equal amount of recordings for all leaf genres, as failure to do this might cause the pattern recognition to find local minima by ignoring genres with few recordings. This had the consequence of causing the genres with the fewest easily available MIDI files to set the ceiling on the number of recordings used per leaf genre.

As a final step, a module was included in the software to report recordings that were statistical outliers after feature extraction was completed for all recordings. This was done in order to flag any recordings that may have been accidentally misclassified manually or if a file was corrupted. Recordings were only reclassified or replaced if an obvious error had been made during manual classification, however. Recordings that were outliers in the feature space but nonetheless had been correctly classified based on information on the web and elsewhere were therefore not altered or removed, as to do so would have artificially inflated success rates.

Space limitations prevent the inclusion of an appendix listing the MIDI files used and statistics regarding their encoding in this document, unfortunately. Such a list can be obtained from the author by writing to him at cory.mckay@mail.mcgill.ca, however.

As a side note, many MIDI files contain meta-data indicating genre. This meta-data is not universally present, however, and there is not any consistent set of genre labels or classification standards that are used. Such meta-data is of very limited use without a reliable standardized system, as its presence and quality depends entirely on the person or people who made a particular file. Genre identifications stored in MIDI files were therefore ignored and classifications were performed based solely on musical content.

An automated system for finding and downloading files and/or access to a large database of easily accessibly music would have greatly helped to increase the number of recordings collected. Commercial or larger scale academic research in the future would benefit from such systems, as twenty-five recordings per category is a relatively small number given the number of categories, and a larger recording library would likely improve performance and make it possible to use more genre categories. A committee-based manual classification process would also be appropriate given the availability of willing committee members.

6.3 Feature extraction and selection

The group of 111 features that was implemented (see Chapter 5) was large enough that the features were not likely to perform well all together, particularly given the training / testing library only consisted of 950 recordings (the "curse of dimensionality" is explained in Section 3.2). Some kind of feature selection and/or weighting, as covered in Section 3.2, was therefore necessary. It was decided not to use any of the dimensionality reduction techniques that cause one to lose the separability of the original features, as which features perform well in distinguishing between different genres has musicological research interest. Exhaustive search strategies were also rejected, as the large number of candidate features made them intractable. Although techniques such as sequential floating selection could certainly have been used, it was decided to use genetic algorithms (GA's) instead, as they are particularly well suited to feature spaces such as that used here (see Section 3.2). Genetic algorithms also have the advantage of allowing one to find feature weightings as well as simple on/off feature selections.

After some informal experimentation with different GA parameters, it was decided to use a roulette crossover system with mutation and with elitism, but without villages. A population of 125 chromosomes was used with a crossover rate of 0.4 and a mutation probability of 0.02. Evolution was continued until the fitness of the best performing chromosome in the population did not improve for 75 consecutive generations or a given maximum was reached. In general, the choice of particular GA parameters is as much of an art as a science. These values worked better than others that were tried, so they were kept. Although more extensive and formal experimentation could have been performed, it was decided that the research resources would be better spent elsewhere.

Two types of feature selection were performed for each classifier: basic on/off feature selection and feature weighting. Feature weighting was performed using 5 bit words for each feature, with a resultant non-normalized weighting between 0 and 1 in 32 increments. Experiments were conducted to see the relative performance of no feature selection at all, only on/off feature selection, only feature weighting and on/off feature selection followed by weighting of the survivors. The results are described in Chapter 7.

A close examination of the features described in Sections 5.4 to 5.10 will reveal that there are two types of features present: those that consist of a single value (referred to here as one-dimensional features) and those that consist of an array of values (referred to here as multi-dimensional features). The reasons for this division into the two types of features are explained in Section 6.4.

As will be seen in Section 6.4, each of these two feature types was treated in a separate way. All one-dimensional features were classified using a single k-nearest neighbour (KNN) classifier, and each multi-dimensional feature was classified using its own feedforward neural network (see Section 3.3 for a description of these classification

techniques). This led to two stages of feature selection, each of which consisted of both on/off selection and feature weighting.

In the first stage, feature selection and/or weighting were applied to the collection of all one-dimensional features. Since only the training recordings were accessible to the system (using testing recordings at any stage of the training violates the validity of the final testing), it was necessary to temporarily partition the training data into feature selection training and testing groups. Genetic algorithms were then used to find the best features and their associated weights using these two training recording sub-sets. The fitness of each chromosome was evaluated by measuring how well a KNN classifier with the selection or weightings of the bit string and trained on the feature selection training group classified the feature selection test group. Once the features and their associated weightings were evolved, a new KNN classifier was trained using these settings and all of the feature values in the entire original training set.

The second stage was just as much a classifier selection process as it was a feature selection process. Which of the classifiers to use and their associated weightings in the combined classification process (see Section 6.5) were found here using a similar genetic-algorithm-based process as in the first stage. The classifiers considered consisted of the KNN classifier trained in the first stage as well as the neural networks, which had each been trained to classify a single multi-dimensional feature. The fitness of each chromosome was determined by how well the classifier selections and weightings associated with its bit string corresponded to the model classifications.

So, to sum up, the one-dimensional features were first selected and/or weighted using a single KNN classifier. This single KNN classifier was then combined with one neural network-based classifier for each multi-dimensional feature, and selection/weighting was performed again, but this time on this entire ensemble of classifiers.

Some initial pre-processing was applied to all of the one-dimensional feature values before selection and/or weighting. It was desirable to use scaling so that all feature values would fall in roughly the same range of values so that all features would start off with equal effective weightings. If this had not been done, then a feature with a range of values between 0 and 1000, for example, would have had far more effect on the distances calculated by a KNN classifier than a feature whose value varied between 0 and 0.1.

Furthermore, it was desirable to moderate the effect of extreme feature values that fell far out of the normal range of a feature. If a particular feature usually varies between 0 and 1, for example, but one recording has a value 1000 for that feature, it is likely that this is due to some kind of noise. If left unmoderated, this extreme value could cause a KNN classifier to misclassify the recording, even if all of its other features would lead to a successful classification. An erroneous value such as this could also detrimentally influence feature selection, if left uncorrected.

The first stage of this pre-processing consisted of finding the standard deviation and mean of each feature across all testing recordings. The maximum for each feature was then set to two standard deviations above the mean, and all values above this maximum were reduced to the maximum value. A similar process was used to ensure that no values were below two standard deviations less than the mean. The following equation was then used to scale all feature values to fall between 0 and 1:

$$v_2 = \frac{v_1 - v_{\min}}{v_{\max} - v_{\min}} \tag{13}$$

where v_2 is the scaled feature value, v_1 is the original feature value, v_{min} is the value 2 standard deviations below the mean and v_{max} is the value 2 standard deviations above the mean.

6.4 Implementation of classifiers

As will be seen in Section 6.5, a hierarchical classification system was used that divided the taxonomy into portions and classified each portion separately, before combining the results to arrive at a final classification. Each portion of the taxonomy, with its subset of all possible categories, was classified into one or more of its candidate categories by a collection of classifiers referred to here as a "classifier ensemble." At the level of abstraction used in Section 6.5, each such ensemble can be seen as a black box object that takes in the features of recordings and outputs scores, between 0 and 1, for each candidate category. This section explains the contents of each of these black boxes.

One will recall from Section 3.3 that nonparametric classifiers are the best type of classifiers to use for tasks such as this sort of musical genre classification, where the underlying statistical distributions of the population's feature values are unknown, where the data is not nominal and where there are multiple candidate categories. In particular, k-nearest neighbour (KNN) and feedforward neural network (NN) classifiers were chosen for use here, as they are often used and known to be effective.

As discussed in Section 6.3, features were divided into two groups, namely one-dimensional features and multi-dimensional features. The one-dimensional features each consisted of single values that were self-contained and meaningful in themselves. The multi-dimensional features, in contrast, each consisted of several values that were closely related to one another. Although it is of course true that all features are potentially interrelated, those sub-features grouped into multi-dimensional features were particularly subject to this interdependence.

As was seen in Section 3.3, KNN classifiers have the advantage of requiring a negligible amount of computation to train, but have the disadvantage that they cannot model complex logical relationships between features. NN's, on the other hand, are

relatively slow to train, but can model sophisticated relationships after training. These relative strengths and weaknesses correspond closely to the characteristics of the two types of features, so it was decided to use a single KNN classifier to perform classifications using all of the one-dimensional features and a separate NN for each multi-dimensional feature. The use of a KNN classifier reduced training time, and the use of NN's made use of their more sophisticated classification ability where it was most needed.

Each KNN classifier calculated a score for each candidate category based on the number of points captured that belonged to each category divided by *k*. The NN classifiers calculated a score for each candidate category based on the value of each output unit, each of which corresponded to a different candidate category.

This collection of component classifiers, namely a single KNN classifier and one NN classifier for each multi-dimensional feature, made up a single complete classifier ensemble. Each of these component classifiers, when provided with the appropriate features of a recording, output a score for each candidate category, and a final set of scores for each category for the ensemble was found by calculating a weighted average of the scores for each category for each component classifier. Which component classifiers were actually used and how their relative weightings were found using genetic algorithms is discussed in Section 6.3. A graphical depiction of a classifier ensemble is shown in Figure 4.

Now that the composition of each classifier ensemble is understood, it is appropriate to delve into some of the details of how each component classifier was implemented. As stated above, each KNN classifier was trained on all of the one-dimensional features, with feature selection and/or weighting performed using genetic algorithms. The value of *k* was set to the square root of the number of training samples. The relative weightings for all features, including multi-dimensional features, could be found by combining these weightings with those found during classifier weighting.

Each NN took in a single multi-dimensional feature. Each dimension of the feature was assigned to a single input unit. The sigmoid function was used as the activation function. One output unit was created for each candidate category, and it was trained to a target value of 0.8 if a training sample belonged to a given category and to 0.2 if it did not. During classification, values on the output units were cut off so that no values could not exceed 0.8 or fall below 0.2. Values read off the output units were automatically scaled so that a value of 0.8 would correspond to a final output of 1 and a value of 0.2 would correspond to a final output of 0. All of this was done rather than simply training units to values of 0 and 1 because the sigmoid function can have difficulty approaching these output extremes.

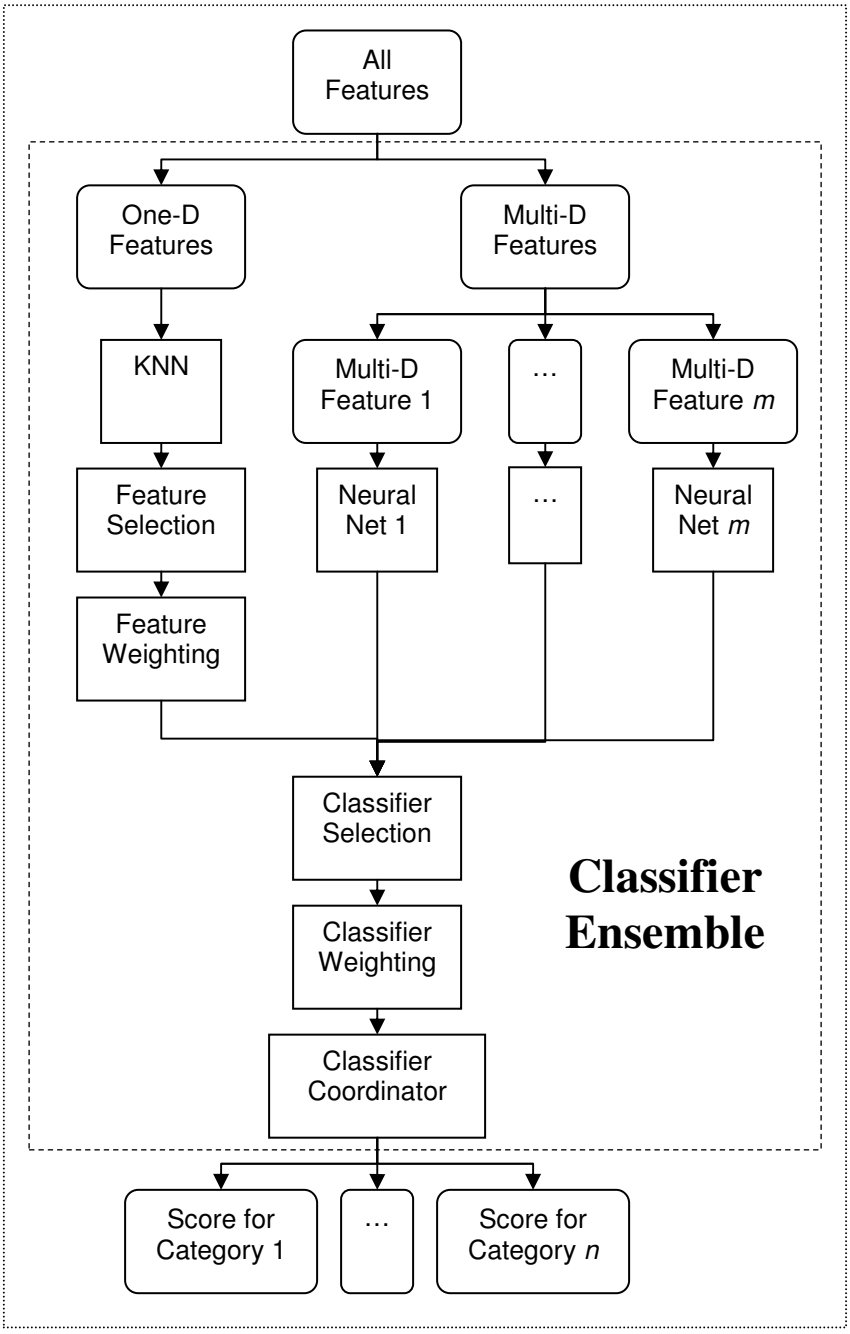

Figure 4: A single classifier ensemble with feature and classifier selection and weighting.

A learning rate of 0.1 was used as well as a momentum of 0.1 in the NNs. Initial weights were randomly set to values between 0.1 and 0.7, and bias units outputting constant values of -1 were used. A single layer of hidden units was used, consisting of a number of units equal to the square root of the sum of the number of input units and the number of output units. These network parameters were found to work well during informal experiments. As is the case with genetic algorithms, there are no reliable commonly accepted techniques for choosing the ideal values for these parameters.

Training was terminated when the absolute change in the sum of squares error across the output units fell below 10^{-7} for 500 consecutive iterations or until a set maximum of iterations was reached. The order of the training samples was randomized.

So, in summary, each ensemble of classifiers took in the complete set of features and output a non-normalized score for each candidate category from 0 to 1. Inside each ensemble, a KNN classifier (with feature selection and weighing determined by genetic algorithms) classified based on one-dimensional features and a separate NN classified each multi-dimensional feature. The final scores for each category for an ensemble were found by calculating a weighted average of the outputs of these component classifiers, with the classifier selection and weightings determined with genetic algorithms.

6.5 Coordination of classifiers

Since a hierarchical taxonomy was used in order to realistically model a way in which humans organize genres, it was decided to take advantage of this organization in order to improve classification performance by using a hierarchical classification technique. As will be seen below, this involved training a number of separate classifier ensembles on different parts of the taxonomy. Each of these classifier ensembles was of the type described in Section 6.4, but can be seen as a black box at this level of abstraction that outputs a certainty score for each candidate category.

Classifying among many categories is in general a harder task than classifying among fewer categories. Simply classifying a recording among all leaf categories (a flat classification) could be a difficult task if there are many such categories. Classifying only among root level categories, in contrast, is likely to be easier because there are fewer candidate categories and what categories there are are likely to be more easily distinguishable at such a broad level.

Hierarchical classification operates by first performing a classification to choose one or more root level categories, then classifying only among the children of those root categories selected, and so on until only a leaf level category or categories remain. This progression from initial coarse classifications to progressively finer classifications as one proceeds down the tree has the advantage that only a small subset of all possible categories must be considered at any one time, thus making the work of the classifiers much easier.

Another important advantage of this approach is that, if feature selection techniques are used, as they are here, a specialized classifier can be trained for each parent node in the taxonomy that uses specialized features to classify among its direct descendants. For example, it would be reasonable for a classifier attempting to distinguish between different types of Western classical music to base classifications partly on whether parallel fifths are present. This feature would be less useful if one is attempting to

distinguish between different types of blues music, for example, but a feature such as whether a harmonica is present or not might be useful here, whereas it would not be for classical music.

This type of hierarchical classification has two important weaknesses, however. The first is that training times are significantly increased because a separate classifier must be trained for each parent node in the taxonomy. This is not as bad as one might imagine for the implementation used in this thesis, fortunately, since each classifier only needs to be trained on the subset of recordings belonging to its particular candidate categories, which reduces training times. Also, the task of each classifier is simpler than it would be with a larger number of categories, so it is likely to converge faster than a classifier attempting to deal with a more complex task would. Finally, the use of KNN classifiers to deal with most of the features significantly cut down on training times.

The second weakness of basic hierarchical classification is that an erroneous classification near the root of the tree will cause classification failure, since an incorrect choice will lead to a path down the tree which cannot lead to a correct leaf category. This is not necessarily a disadvantage when compared specifically to flat classification, since the classifications made by such a classifier are much more likely to be incorrect than the coarse classifications near the root of a hierarchical classifier, but it is still a problem that should be dealt with.

In order to improve results, three types of classification using three different classifier ensembles or groups of ensembles were therefore performed at each level of the hierarchy as a decent was made down the hierarchy:

- **Parent classifier:** a classification was made among the direct descendants of the current node in the taxonomy (or the root categories, when a classification was first begun).
- **Flat leaf classifier:** a classification was made among all possible leaf categories. This was essentially a flat classification. The scores for leaves that were not descendants of at least one of the categories considered by the current parent classifier were discarded, and the scores for the remaining leaves were propagated up the tree until they reached a category that was a direct descendant of the current node, where they were averaged.
- **Round-robin classifier:** a separate classifier was trained for each pair of leaf categories. This was another type of flat classifier. Only the pairs that consisted of descendants of the categories considered by the current parent classifier were used. The results were propagated up the tree, as was done with the leaf classifier.

The scores for each candidate category from each classifier ensemble were then combined using a weighted sum, with the final result deciding which child or children were considered during the next stage of the decent down the hierarchy. It should be noted that the same leaf classifier and round-robin ensembles were used with every parent classifier. Figure 5 displays this classifier process graphically:

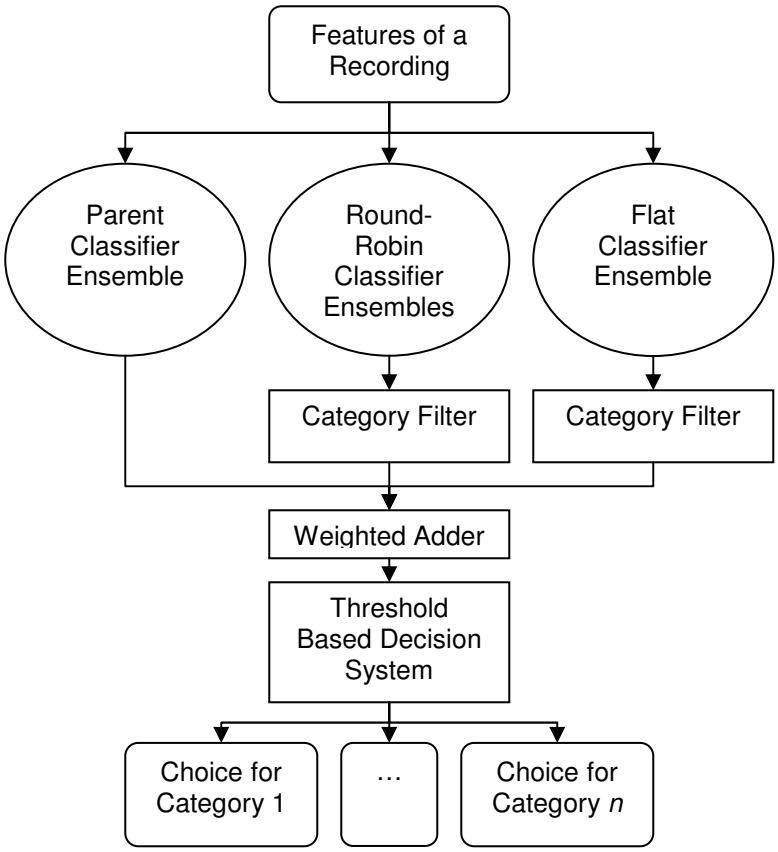

Figure 5: How a decision was arrived at as to which children of a parent category were to be explored or, if they were leaves, selected as winning categories.

This classifier coordination technique is in a sense an expert system that makes use of specialized classifiers (namely the parent classifiers) as well as the knowledge of more general classifiers (the leaf classifier and round robin classifiers). This approach does increase the amount of training time needed, particularly in the case of the round-robin classifiers, but there is good potential for increasing performance. Grimaldi et al. (2003) had some success with round robin classification, so it was decided to test their use, despite the increases in training time.

It was decided to use weightings of 0.6 for the parent classifier, 0.2 for the leaf classifier and 0.2 for the combined results of all the appropriate round-robin classifiers. These values worked well experimentally, and cut down on further training time that

techniques such as genetic algorithms would have required to calculate weightings. The emphasis was put on the parent classifiers, as they used specialized features, and it was therefore reasonable to expect them to perform better than the general-purpose leaf and round-robin classifiers.

Past results from applying hierarchical classification to musical genre have been mixed. Xu et al. (2003) had good results, but Burred and Lerch (2003) found little difference in success rates between flat and hierarchical classification methods. It was therefore decided to compare flat and hierarchal classification experimentally. The results are presented in Chapter 7. The particular hierarchal classification techniques used here are of a novel and generally more sophisticated type than have previously been investigated in relation to musical classification, and the use of many high-level features and of feature selection to train specialized classifiers makes it reasonable to hope for improved results from hierarchical classification methods here.

Each classifier, regardless of which of the three types listed above it belonged to, was an ensemble of classifiers of the type described in Section 6.4. A modular object-oriented design was used here, so that the only visible differences from one instantiation of an ensemble to another were the candidate categories and the training samples. The feature selection and training for each ensemble can be seen as having happened inside the ensemble black box, so that it simply output a score between 0 and 1 for each candidate category.

It was decided to allow classifications into more than one category, as many recordings can indeed be said to belong to more than one genre. It was also decided to allow the system to classify recordings as “unknown” if none of the scores were high enough, as in practical situations such a result can act as a flag to request human intervention, and is certainly better than simply outputting a wrong classification. Both of these decisions complicated the task of the classifiers, but this was necessary in order for the classifications to be realistic.

The potential for membership with multiple categories made it possible for multiple paths to be followed down the classification hierarchy. A result of “unknown” at any point terminated the corresponding path. If no path reached a leaf category, then a recording was classified as unknown. A recording was said to belong to all leaf categories that were reached.

It is now appropriate to explain how it was determined whether or not a given recording was said to belong to a given category based on its classification scores. If classification into exactly one category had been all that was needed, one could have simply called the category with the highest score the winner. However, this was not the case here, so a more sophisticated approach was needed.

All candidate categories whose scores met at least one of the following two conditions were considered winners:

- Score was over 0.5.
- Score was at least 0.25 and was within 20% of the highest score.

These "magic numbers" were arrived at through informal experimentation, and intuitively seem reasonable if one wants to make it possible to have a recording belong to multiple categories but does not want to have false positives. If no categories met either of these two conditions, then a recording was classified as unknown. Categories that did not meet either of the above criteria but did have a score within 30% of the highest score and had a minimum value of 0.2 were considered "secondary choices," which were output simply in order to see how close erroneous classifications were to being correct.

The process described above was applied to any set of category scores when they needed to be resolved into specific results rather than just scores, whether they came from a single classifier inside an ensemble, from the outputs of an classification ensemble as a whole or from the combined results of the three types of classifier ensembles (parent, leaf and round-robin) described above.

6.6 Synopsis of classification architecture

The system described in Sections 6.3 to 6.5 is somewhat complex. A brief summary of the system described in these sections is therefore provided here for the purpose of unifying the reader's overall conception of the system.

During training, a number of classifier ensembles are created based on the hierarchical model taxonomy provided to the software by the user. Each of these ensembles take in the full set of feature values of a recording as input and provide a score for each candidate category as output.

One leaf classifier ensemble is created that has all leaf categories as candidate categories. In addition, one round-robin classifier ensemble is created for every possible pair of leaf categories. Each round-robin ensemble has only its corresponding pair of categories as candidate categories. Finally, one parent classifier ensemble is created for each category that has child categories in the provided taxonomy. Each parent ensemble has only its direct children as candidate categories.

Each ensemble is trained using only the recordings belonging to at least one of its candidate categories. Each ensemble consists of one KNN classifier for use with all one-dimensional features and one NN for each multi-dimensional feature. The scores output by the KNN and NN classifiers are averaged in order to arrive at the final scores output by the ensemble. Feature selection is performed on the one-dimensional features during

training, followed by feature weighting on the survivors of the selection. Classifier selection and weighting is also performed on the scores output by the KNN and NN classifiers, so that each classifier has a weight controlling the impact of its category scores on the final score for each category output by the ensemble. All selection and weighting is done using genetic algorithms.

Actual classification is performed by descending down the hierarchical taxonomy one level at a time. A weighted average of the results of the flat, round-robin and parent classifier ensembles is found for each child of the parent category under consideration, and only those child categories whose scores for a given recording are over a certain threshold are considered for further expansion down the taxonomy tree. The leaf category or categories, if any, that pass the threshold are considered to be the final classification(s) of the recording.

6.7 Java and XML

The software written for this thesis was implemented in Java and made use of XML. These technologies are therefore reviewed briefly here.

Java is an object-oriented programming language developed by Sun Microsystems. Software developers write Java code using integrated development environments, or sometimes just simple text editors. Java code can then be compiled into Java Bytecode, which is a platform independent binary encoding that can be understood and run by implementations of the Java Runtime written for different operating systems. This effectively makes Java code platform independent, as Java Bytecode should be run identically by all distributions of the Runtime. In actual fact, there can be some inconsistencies, particularly relating to graphical user interfaces, but in general platform independence is successful.

Both the Java Development Kit and the Java Runtime are available free at www.java.sun.com. An important advantage of Java is that the Development Kit is distributed with a large standard class library, which includes sophisticated graphical user interface components (Swing) as well as MIDI and audio file parsing classes.

The combination of platform independence, the large standard class library and the ease with which Java code can be written, maintained and extended made it ideal for this project. Java also runs faster than code written in many other languages, making it appropriate for processing intensive tasks such as those in this project. C and C++ do in general run slightly faster than Java, but they lack many of the advantages of Java.

Gosling and Arnold's book (1996) offers an excellent introduction to the basics of Java and how the philosophy behind object-oriented program can be fulfilled with it. Horstmann and Cornell have written two books (2000 and 2001) that are good practical

guides to Java. Those looking for information specifically about Swing should consult Geary (1999) or Pantham (1999).

The software developed in this project stores a wide variety of data and configuration settings on disk using computer files. A technology called XML was used to do this. XML is a general purpose markup language that enables developers to custom design legible and robust formats for storing and transmitting data. There is a free and particularly good XML parser implemented in Java named Xerces, which was used in this thesis. Xerces is available at xml.apache.org/xerces-j/. Whitehead, Friedman-Hill and Vander Veer (2002) provide a good guide to XML in general and to using XML with Java.

6.8 Software and its interface

A modular, object-oriented software engineering approach was used to design all parts of the software, making it a simple matter to upgrade and expand the software in the future. A new feature could be added, for example, simply by writing a new class that implements the *Feature* abstract class and adding a simple reference to the new class in the *FeatureMaker* class. Similarly, new types of classifiers could be added simply by implementing the *SupervisedClassifier* abstract class and including references to the new class in the *ClassificationPanel* class.

Furthermore, the software was designed to perform classifications of any kind, not limited in scope to classifications of MIDI files by genre. Someone wishing to classify audio recordings, for example, would simply need to write an audio processing class and a class for each feature. No other modifications to the code would be necessary, as the implementation used makes no assumptions as to the type of data being processed.

If a user is concerned only with MIDI data, then the system could be used as is to perform supervised classifications of any kind simply by altering the taxonomy that is used and the model training classifications. This could be done entirely through the GUI, without and modifications at all to the code being necessary.

This flexibility and extensibility of the software is an important strength of the system. This makes the software well suited for those with some programming experience who wish to quickly build a fully functional classification system but wish to specify some key aspects to their particular needs.

The software was also written with the needs of those people with no coding experience or no inclination to write code themselves in mind. A highly flexible graphical user interface was designed that enables users to perform all tasks and customize settings within the interface itself without needing to directly modify any configuration files. The software allows users to easily:

- Custom design their own hierarchal or flat taxonomies (Figure 6).
- Control the settings for each feature and get precise descriptions of each feature (Figure 7).
- View and edit the meta-data of the loaded recordings, extract features from recordings and view feature values both before and after scaling (Figure 8).
- View and edit preferences relating to neural networks, genetic algorithms, whether or not feature and classifier selection and/or weighting is performed, what classifiers are included in classification ensembles, what types of classifier ensemble coordination are performed, what thresholds are used to terminate training and perform classifications, what information is reported in training, classification and feature selection/weighting reports, etc. (Figure 9).
- Train and classify recordings. The user has the option of training on all recordings, randomly reserving a certain percentage from each category for testing or automatically performing cross-validation tests. Formatted reports are generated giving a wide variety of statistics on training progress, feature selections and weightings and classification results, either as a whole or for individual classifiers or classifier ensembles (Figure 10).
- Save all if this information in configuration files that can be loaded and automatically parsed either individually or in project groups.
- View graphical on-line help information explaining how to use the software.

All code other than the Xerces XML parser and the *SwingWorker* class (available on the *Java Technology* web site), including classifiers, was written by the authour. Copies of the software and the source code may be acquired by contacting the authour at cory.mckay@mail.mcgill.ca.

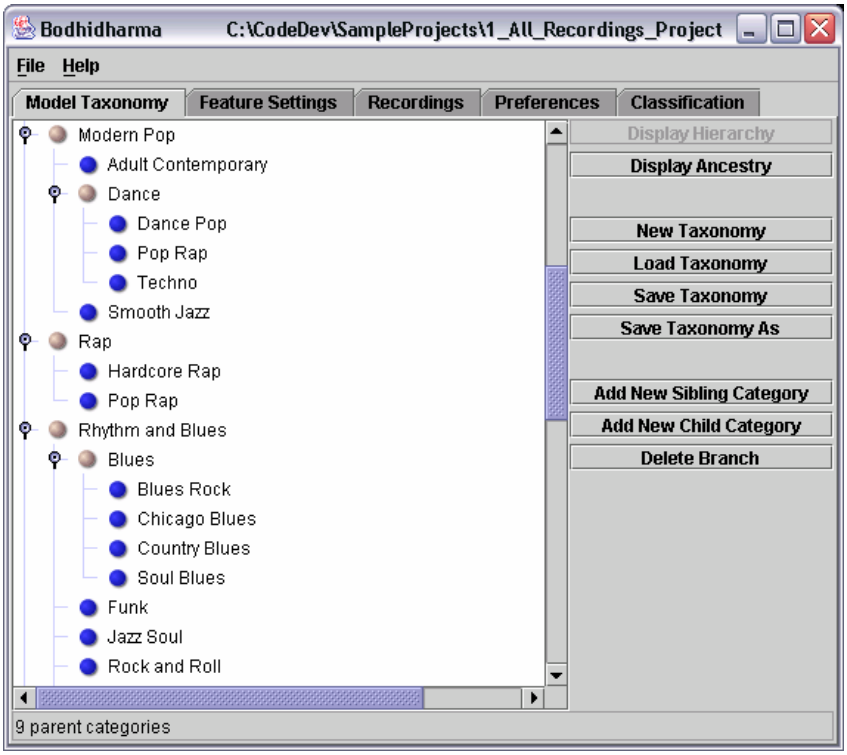

Figure 6: Component of interface used to edit and view taxonomies.

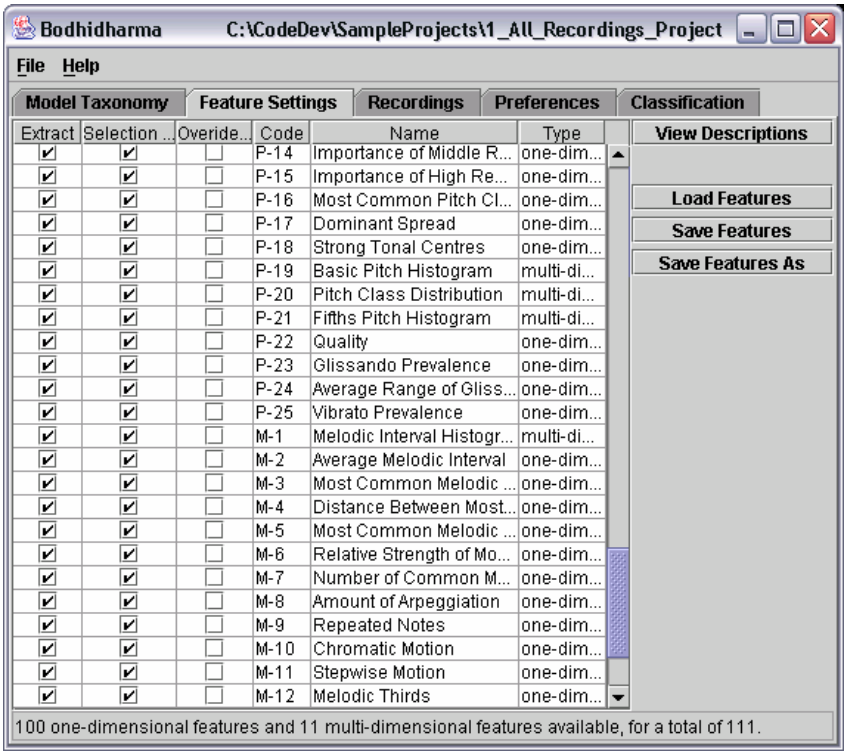

Figure 7: Component of interface used to edit and view feature settings.

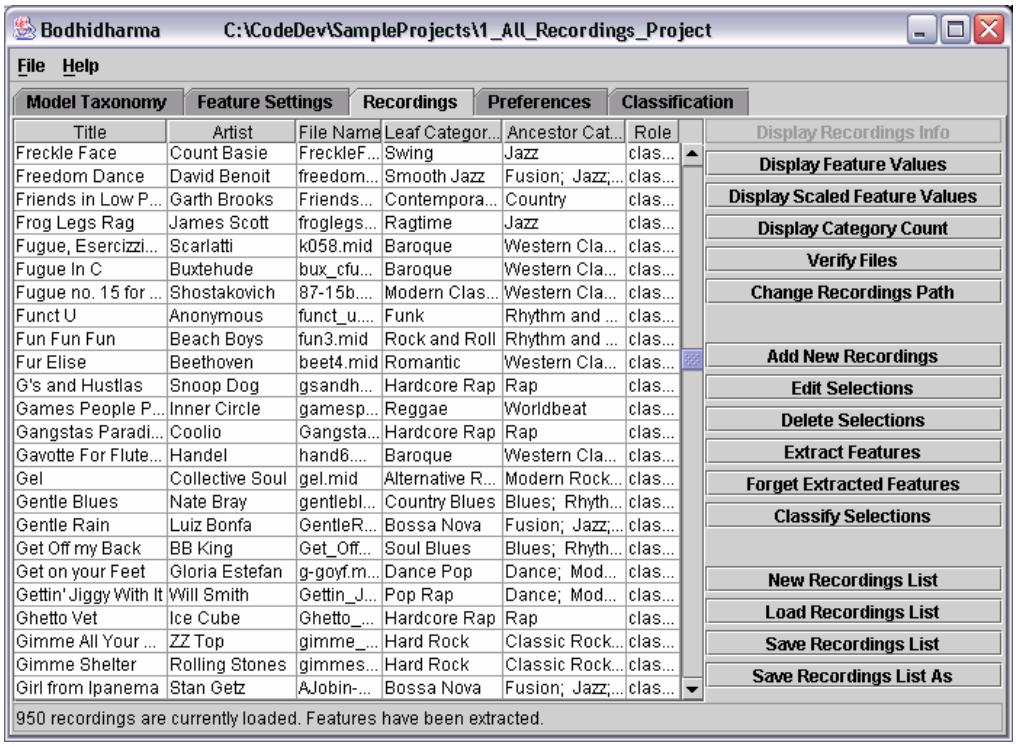

Figure 8: Component of interface used to edit and view recordings.

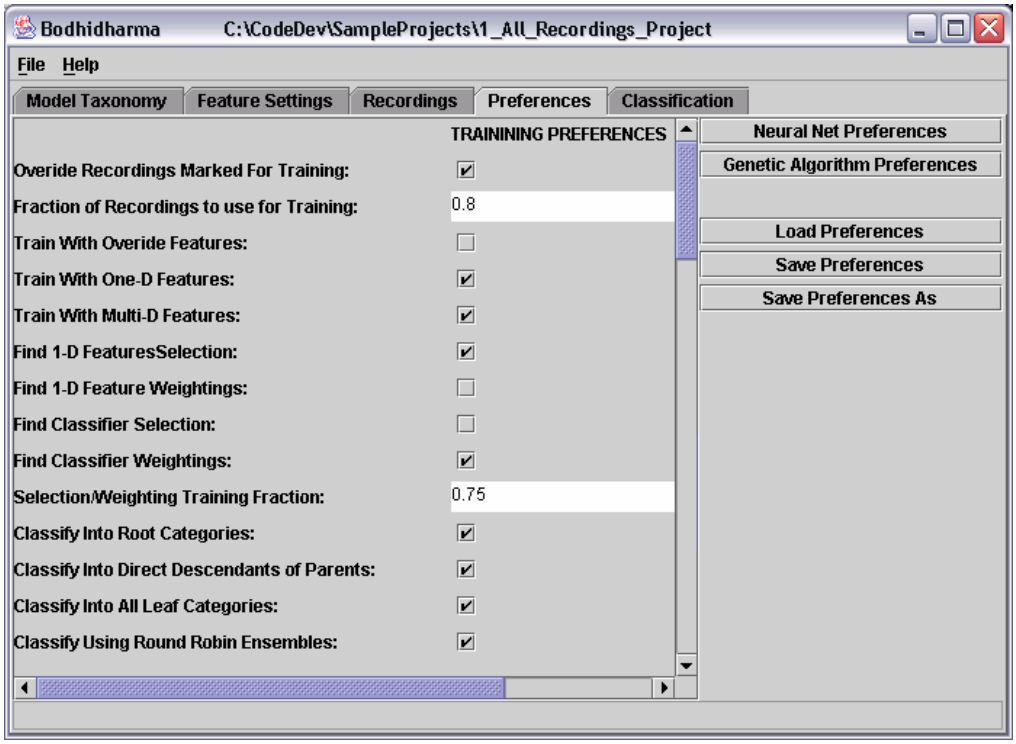

Figure 9: Component of interface used to edit and view preferences.

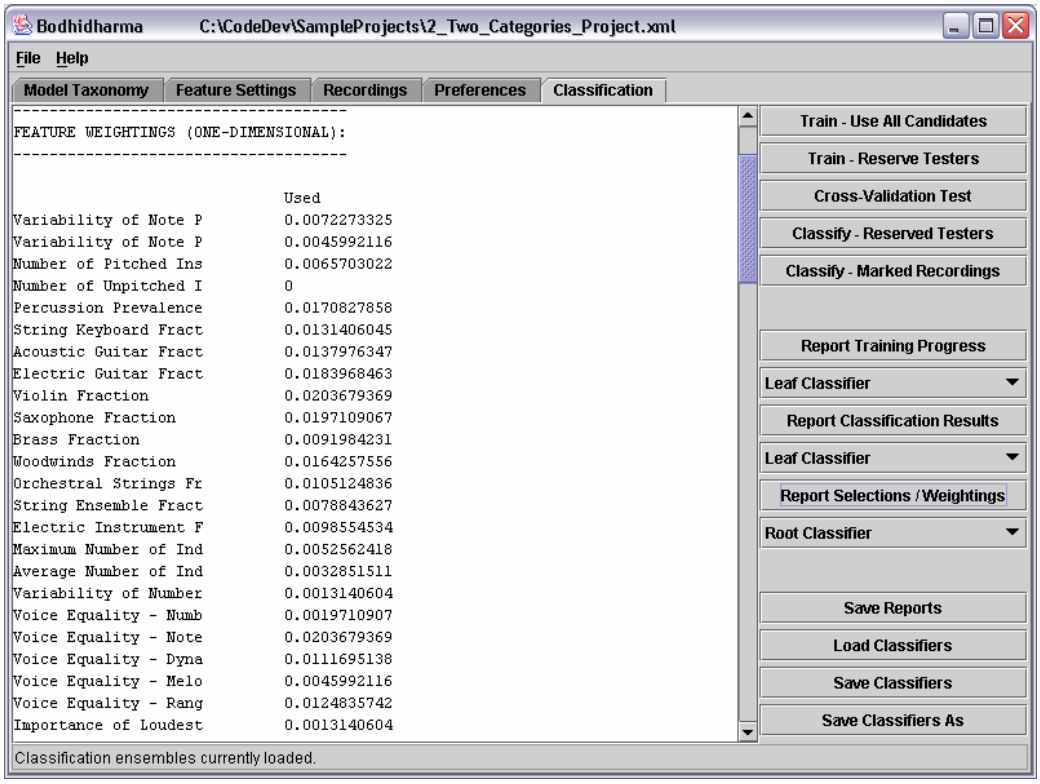

Figure 10: Component of interface used to train and classify recordings as well as see training, classification and feature selection results.

7. Experiments, Results and Discussion

7.1 Experimental methodology

The experiments discussed in this chapter were all performed using the features described in Chapter 5 and the taxonomies, library of recordings, feature selection methodologies, classifiers, classifier ensemble coordination techniques, software and interface described in Chapter 6.

All tests were performed using 5-fold cross-validation, which means that each experiment was performed five times, with each fold reserving different subsets of the available recordings for testing. 80% of the recordings were used for training and 20% for testing in each fold, with the membership of each of these groups determined randomly on a leaf genre by leaf genre basis. This means that every fold reserved 20% of the recordings belonging to each leaf genre for testing, and that each recording served as a testing sample exactly once during all of the five folds and as a training sample exactly four times. The results reported for each experiment in this chapter are all averages of the results for each of the folds in the corresponding experiment.

All training times reported in the following sections were dependant on the computer used and factors such as temperature of the room where the tests were performed. Efforts were made to keep conditions consistent across experiments, although some small variations were unavoidable. All tests were performed on a Pentium 4 2.8 GHz based machine with an 800 MHz front side bus, 1.5 GB of 400 MHz RAM and an 80 GB 7200 RPM hard drive with an 8 MB cache. The computer was running Windows XP Home, with the network cable disconnected and no applications running other than the classification system.

Some initial informal experiments were performed to explore the effectiveness of various basic classifier parameters, such as neural network learning rates and momentums. These are omitted here for lack of space, but the settings that were used are available in Chapter 6. More formal experiments were performed with regards to the use of feature selection and weighting, classifier selection and weighting, training iterations and classifier ensemble coordination methodology, however, as these parameters have some research interest. These experiments and their results are presented in the following sections.

Experiments were performed using both of the taxonomies presented in Section 6.1. The term “T-9” is used in this chapter to refer to the taxonomy shown in Figure 2 (with 9 leaf categories), and “T-38” is used to refer to the taxonomy shown in Figure 3 (with 38 leaf categories). Both figures are on page 81 As discussed in Section 6.1, T-9 was chosen in order to provide a rough means of comparing this system’s performance to that of

previous systems, as T-9 has a size and difficulty similar to that of the most difficult taxonomies used in previous research. T-38, in contrast, is a much more sophisticated taxonomy than has, to the best of the authour's knowledge, been used to test any existing automatic genre classification system. It is therefore included here to test the boundaries of this classification system beyond the scope of testing in previous research.

The majority of the experiments presented in this chapter focus on Taxonomy T-9, partly because T-9 provides a basis for comparison with previous research, and partly because T-38 requires much longer training times. Training time can increase exponentially with the number of training samples, depending on the classification configuration used. This made extensive experimentation with T-38 much more resource expensive, an important issue when it is considered that five complete folds had to be performed for each experiment.

One would ideally have liked to exhaustively experiment with every possible combination of configuration settings, but the wide range of possible configurations and the training time needed for each experiment made this an unrealistic goal in the context of this thesis. As a good second best option, experiments were performed so as to build upon each other in an incremental fashion in order to attempt to find optimal or near-optimal configurations.

The results of cheaper experiments with a small taxonomy can, to a certain extent, give indications of the best classification configuration to use for a larger taxonomy. One must be cautious on this point, however, since it will not always necessarily be correct. A flat leaf classification system, for example, might perform very well with a small taxonomy, but be unable to deal with the complexity of a more sophisticated taxonomy.

Sections 7.3 to 7.7 deal with T-9 exclusively, and Section 7.9 deals with T-38 exclusively. The experimental configurations and results of all experiments, for both taxonomies, are available below in Section 7.2.

7.2 Details of experiments and their results

The details of all of the experiments and their associated classifier configurations are given in Table 2 and Table 3. The letter codes given in the first column of each of these tables is used to identify each of the experiments through the remainder of this text. The results of these experiments can be found in Table 4 and Table 5. The remaining sections of this chapter analyze and compare the especially pertinent aspects of these experiments in a variety of contexts.

A variety of statistics were collected on the results of each experiment. The statistics presented in Tables 4 and 5 are defined as follows:

- **Training Time:** The amount of time, in minutes, needed to completely train a full classification system for a given taxonomy using all of the assigned training recordings. In other words, the time to complete one cross-validation fold.
- **Success Rate:** The percentage of recordings that were assigned the correct leaf category(ies) by the classification system. Recordings belonging to multiple leaf categories were only counted as partially correct if some categories were missed by the classification system. The success rate was calculated by assigning each recording a score corresponding to the number of correct leaf categories assigned to it by the classifier divided by the number of model leaf categories that the recording belonged to, and then averaging these fractions across all recordings and multiplying the result by 100%.
- **Secondary Success Rate:** The percentage of those recordings where at least one model leaf category was missed by the classifier, but where that category was assigned as a secondary choice by the classifier.
- **Over Select Rate:** The percentage of recordings that were assigned at least one extra (i.e. beyond the total number of model categories for the recording) leaf category by the classification system that was not one of the model leaf categories for the recording.
- **Root Rate:** The percentage of recordings that were correctly classified as descendants of the correct root category(ies). The scores for recordings belonging to descendants of multiple root categories were calculated in the same way as described for the success rate above.
- **Unknown Rate:** The percentage of missed classifications that were classified as unknown rather than being assigned an erroneous label or labels.

A particular emphasis is put in the following sections on the overall success rate and the training times. These statistics are in a sense the “bottom line,” since the ultimate goal of a classification system is to maximize the number of correct classifications while minimizing processing time. The other statistics are discussed in the following sections when they are remarkable for a particular experiment.

The uncertainties that accompany the values on Tables 4 and 5, as well as the error bars for each of the figures in this chapter, are based on the standard error (standard deviation divided by the square root of the number of trials) of measurements across all folds for each experiment. The fact that there is some variation from fold to fold is not surprising, as success rates are dependant on both the particular recordings randomly selected to be used for training and for testing and on the non-deterministic training of neural networks and genetic algorithms.

Code	Figures Used In	Classifier Ensemble Coordination	One-D Features	Multi-D Features	NN Epochs
A	12, 13, 16	Hierarchal, Flat, RR	All	None	--
B	12, 13, 16	Hierarchal, Flat, RR	Selected (50)	None	--
C	12	Hierarchal, Flat, RR	Weighted (50)	None	--
D	12	Hierarchal, Flat, RR	Selected (50), Weighted (50)	None	--
E	13	Hierarchal, Flat, RR	Selected (25)	None	--
F	13	Hierarchal, Flat, RR	Selected (90)	None	--
G	14, 15	Hierarchal, Flat, RR	All	All	1000
H	14	Hierarchal, Flat, RR	All	Selected (200)	1000
I	14, 15, 16, 17	Hierarchal, Flat, RR	All	Weighted (200)	1000
J	14	Hierarchal, Flat, RR	All	Selected (200), Weighted (200)	1000
K	15	Hierarchal, Flat, RR	All	Weighted (100)	1000
L	15	Hierarchal, Flat, RR	All	Weighted (300)	1000
M	17	Hierarchal, Flat, RR	All	Weighted (200)	500
N	17	Hierarchal, Flat, RR	All	Weighted (200)	1500
O	16	Hierarchal, Flat, RR	None	Weighted (200)	1000
P	16, 18, 19, 20	Hierarchal, Flat, RR	Selected (50)	Weighted (200)	1000
Q	18	RR (many winners)	Selected (50)	Weighted (200)	1000
R	18	RR	Selected (50)	Weighted (200)	1000
S	18, 19, 20	Flat	Selected (50)	Weighted (200)	1000
T	18, 19, 20	Hierarchal	Selected (50)	Weighted (200)	1000
U	18, 19, 20	Flat, RR	Selected (50)	Weighted (200)	1000
V	18, 19, 20	Hierarchal, RR	Selected (50)	Weighted (200)	1000
W	18, 19, 20	Hierarchal, Flat	Selected (50)	Weighted (200)	1000
Z	21, 22	Hierarchal, Flat, RR	Selected (100)	Weighted (400)	2000

Table 2: Classifier configurations for experiments involving Taxonomy T-9**.** *Code* gives the unique identifier of an experiment. *Figures Used In* gives the figures in which the values from each experiment are displayed. *Classifier Ensemble Coordination* gives the techniques that were used to combine the results of all classifier ensembles (hierarchal, flat leaf and/or round robin). *One-D Features* gives which one-dimensional features were used to perform classifications (none, all, those selected through feature selection and/or with weightings). *Multi-D Features* gives which multi-dimensional features/classifiers were used to perform classifications (none, all, those selected through classifier selection and/or with weightings). *NN Epochs* gives the maximum number of epochs used to train neural networks. Values in parentheses give the maximum number of genetic algorithm generations used in the corresponding training.

Code	Figures Used In	Classifier Enemble Coordination	One-D Features	Multi-D Features	NN Epochs
AA		Flat	All	None	--
BB		Hierarchal, Flat, RR	All	None	--
CC	23, 24	Hierarchal, Flat	All	Weighted (105)	2000
DD	23, 24	Hierarchal, Flat	Selected (35)	Weighted (105)	2000
EE	23, 24	Hierarchal	Selected (30)	Weighted (150)	3000
FF	23, 24	Hierarchal	All	Weighted (150)	3000

Table 3: Classifier configurations for experiments involving Taxonomy T-38. Columns have same meanings as in Table 2.

Code	Training Time (min)	Success Rate (%)	Secondary Success Rate (%)	Over Select Rate (%)	Root Rate (%)	Unknown Rate (%)
A	0.0 ± 0.0	75 ± 3	6 ± 3	5 ± 3	96.9 ± 1.1	0 ± 0
B	27.9 ± 0.4	80 ± 2	10 ± 6	5 ± 1	95.1 ± 1.5	0 ± 0
C	50.9 ± 0.1	80 ± 3	7 ± 3	3 ± 1	96.9 ± 0.9	0 ± 0
D	63.0 ± 0.3	81 ± 2	9 ± 4	6 ± 2	96.4 ± 1.1	0 ± 0
E	14.0 ± 0.1	79 ± 3	11 ± 1	4 ± 1	97.1 ± 1.4	0 ± 0
F	43.9 ± 0.1	78 ± 3	4 ± 2	7 ± 1	94.7 ± 1.1	0 ± 0
G	19.1 ± 0.0	85 ± 3	7 ± 7	3 ± 1	98.7 ± 0.5	0 ± 0
H	24.6 ± 0.0	84 ± 2	20 ± 5	12 ± 2	97.8 ± 0.7	0 ± 0
I	25.9 ± 0.2	89 ± 2	18 ± 6	10 ± 1	98.9 ± 0.7	0 ± 0
J	29.3 ± 0.1	87 ± 3	20 ± 11	9 ± 1	96.4 ± 1.3	5 ± 3
K	25.9 ± 0.1	88 ± 1	19 ± 11	4 ± 1	97.8 ± 0.0	0 ± 0
L	26.4 ± 0.0	83 ± 1	17 ± 7	3 ± 2	96.9 ± 1.1	0 ± 0
M	16.2 ± 0.1	83 ± 2	14 ± 2	4 ± 1	97.8 ± 0.7	0 ± 0
N	36.9 ± 0.0	86 ± 2	28 ± 12	5 ± 1	96.9 ± 1.2	0 ± 0
O	26.4 ± 0.2	83 ± 2	9 ± 5	4 ± 1	96.9 ± 1.1	2 ± 3
P	53.3 ± 0.1	85 ± 3	9 ± 6	6 ± 1	97.8 ± 0.0	0 ± 0
Q	30.1 ± 0.1	100 ± 0	0 ± 0	96 ± 2	99.6 ± 0.4	0 ± 0
R	32.2 ± 0.5	85 ± 3	0 ± 0	NA	96.9 ± 0.9	0 ± 0
S	9.7 ± 0.0	86 ± 2	30 ± 5	14 ± 2	95.6 ± 1.4	15 ± 6
T	13.6 ± 0.0	90 ± 2	24 ± 8	21 ± 3	98.2 ± 0.8	0 ± 0
U	42.0 ± 0.6	84 ± 1	0 ± 0	0 ± 0	96.9 ± 0.9	0 ± 0
V	43.6 ± 0.1	83 ± 1	12 ± 7	3 ± 1	96.4 ± 1.1	0 ± 0
W	25.2 ± 0.4	89 ± 2	54 ± 14	19 ± 2	99.1 ± 0.5	0 ± 0
Z	89.5 ± 0.3	86 ± 3	13 ± 8	4 ± 3	95.6 ± 1.9	0 ± 0

Table 4: Classification results for experiments involving Taxonomy T-9. All values are averages across all folds. Uncertainty values correspond to the standard error. See page 103 for the meaning of the columns.

Code	Training Time (min)	Success Rate (%)	Secondary Success Rate (%)	Over Select Rate (%)	Root Rate (%)	Unknown Rate (%)
AA	0 ± 0	37 ± 2	10 ± 0	7 ± 1	70.7 ± 0.6	43 ± 7
BB	0 ± 0	43 ± 2	2 ± 1	4 ± 1	63.3 ± 0.1	37 ± 6
CC	99 ± 2	57 ± 5	13 ± 1	21 ± 1	74. 9± 0.1	32 ± 1
DD	331 ± 2	52 ± 0	15 ± 0	22 ± 1	71.5 ± 2.0	29 ± 6
EE	196 ± 4	56 ± 1	11 ± 1	31 ± 2	81.2 ± 0.9	5 ± 1
FF	80 ± 1	57 ± 1	13 ± 2	31 ± 1	80.6 ± 1.1	4 ± 1

Table 5: Classification results for experiments involving Taxonomy T-38. All values are averages across all folds. Uncertainty values correspond to the standard error. See page 103 for the meaning of the columns.

It should be understood that these experiments were not intended to be rigorous and conclusive scientific studies of which classification configurations are absolutely better in general. This would have required many more trials than were performed here, with a larger number of training and testing recordings. Rather, these experiments were intended as rough examinations of how well different techniques worked in the context of taxonomies of the particular types studied here. Similar experiments can and should be performed in other research contexts in order to gain better insights on the performance of the different classifier configurations in regards to the general problem of music classification. The flexibility of allowing users to customize classifier configurations to meet different user needs is an important advantage.

7.3 Number of features

Before beginning a more detailed analysis the experiments described in Tables 2 through 5, it is appropriate to gain a baseline by first briefly reviewing some earlier experimental results that have been submitted for publication elsewhere. These studies used taxonomies almost identical to T-9 and used recordings drawn from the same pool as here.

The pilot study for this thesis (McKay 2004) used eight ensembles of neural networks combined using simple weighting to perform genre classifications of MIDI files using 20 high-level musical features. Leaf genres were correctly classified 58% of the time and parent genres were correctly classified 85% of the time.

Although these results were acceptable for the purposes of that study, it was hoped to improve these rates here. Three primary areas of improvement were identified: increasing the number of recordings, improving the classification system and increasing the number of features. All three approaches have been addressed in this thesis.

In regards to the number of recordings, the library of recordings has been expanded from 225 to 950, although this increase has had the effect of expanding the number of categories rather than the number of recordings per category, for reasons made clear in Sections 6.1 and 6.2. The classification system presented in this thesis has also greatly increased in sophistication, as can be seen in Chapter 6. As for the number of features, Chapter 5 makes it clear that many more are available now than were previously.

According to the “curse of dimensionality” (discussed in Chapter 3), too many features can make classifications harder. This is one of the reasons that feature selection has been given such an emphasis in this thesis. A study (McKay & Fujinaga 2004) was made on the effect of varying the number of candidate features from which the feature selection system could choose while keeping other classifier parameters constant. The particular features that were made available to the feature selection sub-system were randomly selected for each trial. The results are shown in Figure 11.

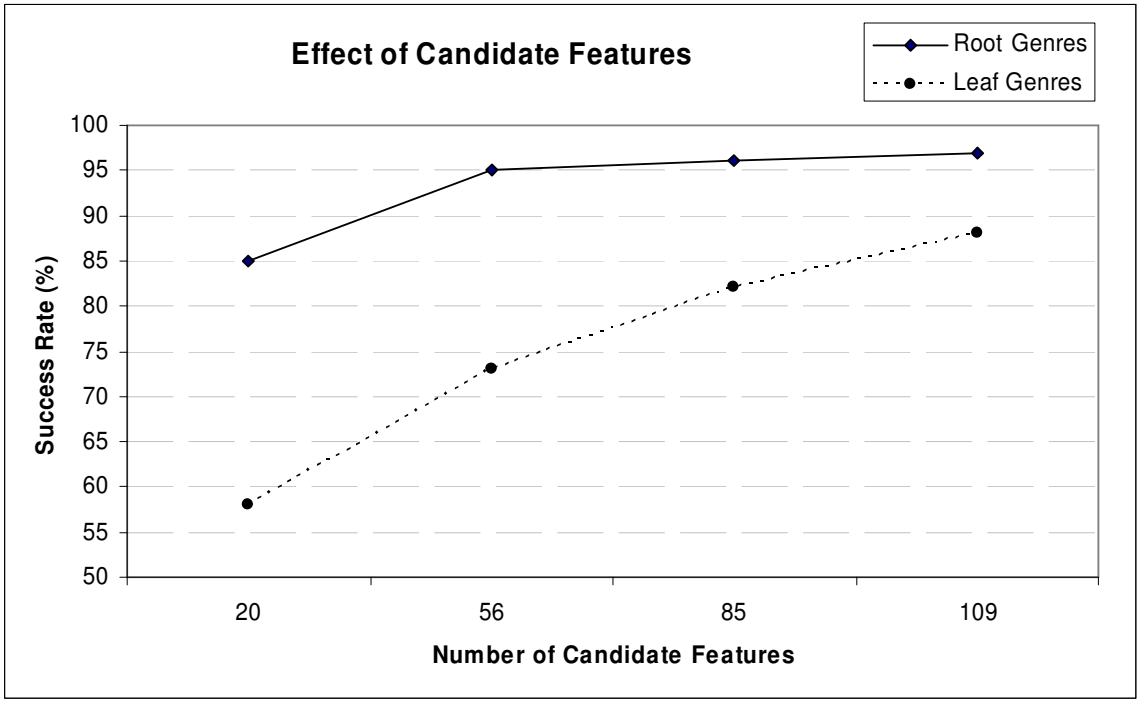

Figure 11: Effect of varying number of candidate features available to feature selection system (McKay & Fujinaga 2004).

Figure 11 makes it evident that a large number of candidate features paired with feature selection greatly increased performance. Furthermore, this classification was done using flat leaf classification only. The potential increases in performance could be even greater with hierarchal or round robin classification, as these techniques take advantage of the availability of specialized features suited to their particular candidate categories.

7.4 Feature and classifier selection methodology

Once it had been established that a large number of candidate features was indeed beneficial, the next step was to examine the relative performance of different selection techniques. One will recall from Section 6.3 that four operations of this type were performed, namely 1-D feature selection, 1-D feature weighting, classifier selection and classifier weighting. Classifications were performed with various combinations of these selection methods in order to find the ones that maximized performance while minimizing training time.

It can be seen from Figure 12 that some type of one-dimensional feature selection did improve performance over no feature selection, but it is unclear which form of feature selection was the best. Selection alone did require significantly less time than weighting or selection and weighting, however (27.9 minutes versus 50.9 and 63.0 minutes respectively), so this was therefore chosen as the best option for Taxonomy T-9.

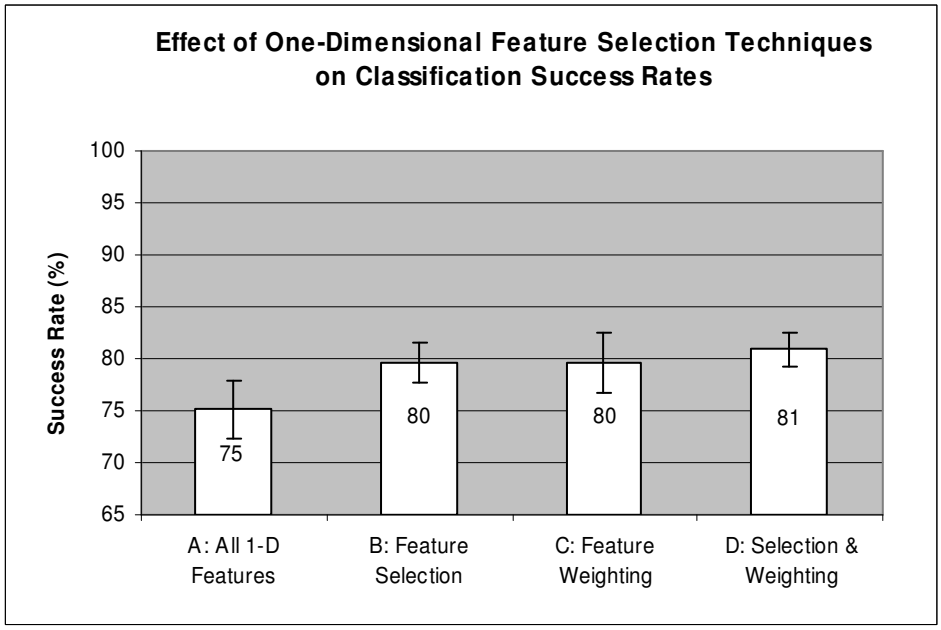

Figure 12: Effect of different one-dimensional feature selection methodologies on the classification of Taxonomy T-9. Values are averages over all folds and error bars correspond to standard error. Letter codes identify the experiment (see Table 2).

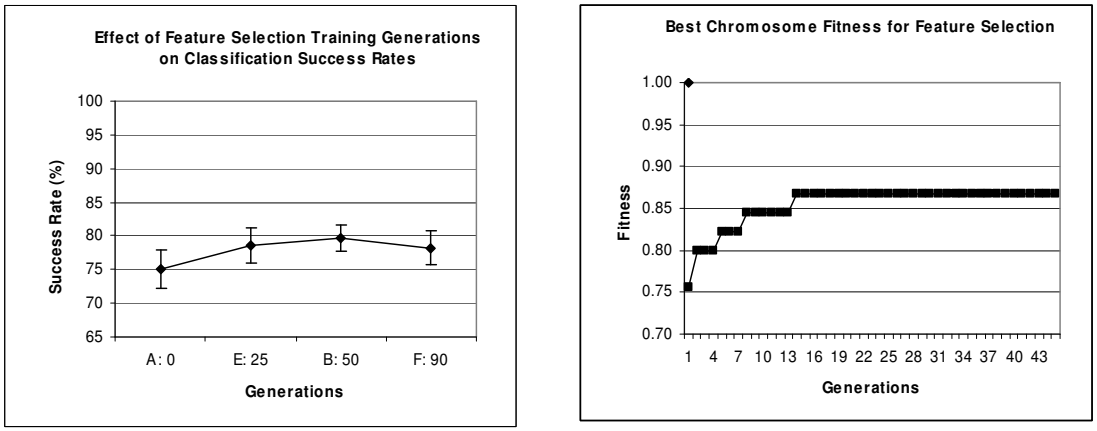

Figure 13: Effect of different numbers of feature selection training generations. The left plot gives the average success rates of Experiments A, E, B and F. The right plot gives the feature selection fitness of the best chromosome of a typical flat leaf classifier as it evolved.

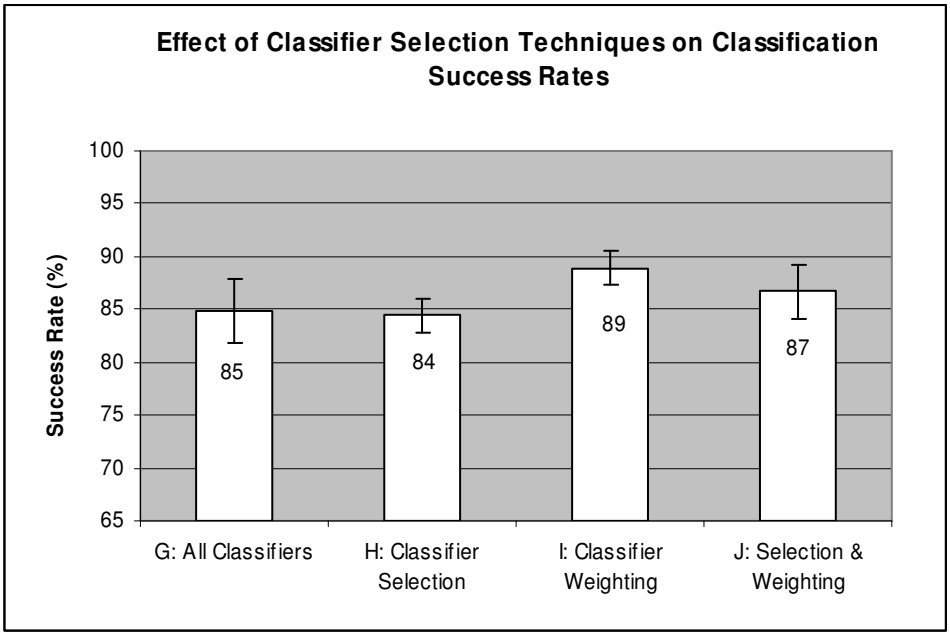

Figure 14: Effect of different classifier selection methodologies on the classification of Taxonomy T-9. Values are averages over all folds and error bars correspond to standard error. Letter codes identify the experiments (see Table 2).

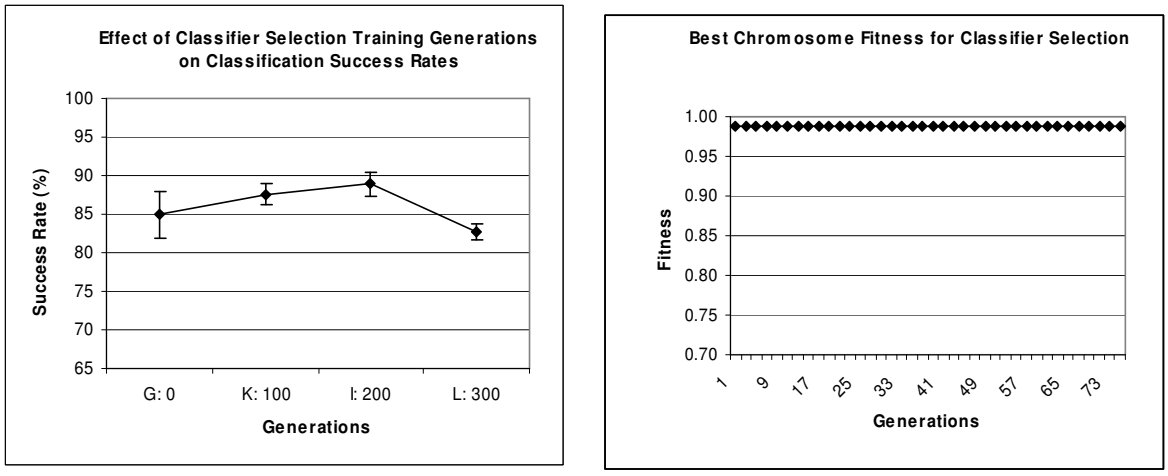

Figure 15: Effect of different numbers of classifier selection training generations. The left plot gives the average success rates of Experiments G, K, I and L. The right plot gives the fitness of the best chromosome of a typical round robin classifier as it evolved. Note that training was automatically terminated after 75 consecutive generations with no change.

Figure 13 appears to indicate that relatively few generations were needed for the genetic algorithm to converge to a good solution. Of the trials performed, the best chromosome was generally found in the first 20 or so generations, and additional evolution usually caused no improvement. This rapid convergence is good, as feature selection was a relatively time consuming process per iteration compared to classifier selection and neural network training. Of course, it is certainly possible that a superior chromosome could always appear after many generations of apparent stagnation, but apparent asymptotic behaviour and acceptable success rates nonetheless seems to make it fairly safe to say that the small potential for increased performance brought about by extended evolution has too high a training time cost associated with it for Taxonomy T-9.

The next step was to experiment with various classifier selection methodologies, namely none, selection, weighting or selection and weighting. Since classifier selection involved training one KNN classifier on all one-dimensional features and one neural network classifier for each multi-dimensional feature, it was necessary to include neural network training here.

According to the results shown on Figure 14, there did not appear to be all that much difference between the different methodologies, if one considers the error bars. Classifier weighting alone did seem to perform slightly better than the alternatives, however, and it was fairly inexpensive to use in terms of training time (25.9 minutes total training time as opposed to 19.1 minutes for no classifier selection at all). As for training generations, the classifier selection systems tended to converge very quickly in many cases, even on the first or second generation, as can be seen on Figure 15. As a side note, the relatively low performance of Experiment L was probably due to a particularly inopportune selection of training versus test recordings rather than to the classifier selection.

It was somewhat surprising that feature and classification selection did not have a more significant effect, so a closer look was taken at the individual classifiers in order to gain insights into why this was the case. It turns out that the round robin classifiers virtually always achieved perfect classification success with the training samples even with no selection, and that the hierarchal classifiers other than the root classifier did the same almost as often. This means that a chromosome with a very sub-optimal selection or weighting had perfect fitness, since it still resulted in a perfect classification of training samples, so the genetic algorithm had no reason to evolve a better selection. However, during testing, the round robin and hierarchal classifiers were exposed to recordings belonging to categories other than their candidate categories, which they had not been exposed to during training. Their behaviour was unpredictable in these cases. In essence, the classifiers performed so well during training, that they were unable to evolve good selections and weightings that would improve testing rates. This explains why feature selection had a relatively small effect in these experiments, as those classifiers that would

have benefited from it the most had perfect fitness already, as far as the genetic algorithm was concerned, so no evolution and corresponding improvement took place for them.

A potential solution to this problem would be to train the round robin and hierarchal classifiers on recordings that do not belong to their candidate categories as well as those that do. This would make the classifiers' task more difficult initially by forcing them to learn to reject more types of recordings, thereby hopefully getting non-perfect classifications with poor feature sets during training, thereby causing the genetic algorithm to evolve better feature sets. This approach would greatly increase training times, however, which could be a critical problem with round robin classification, as this requires a great many classifiers for large taxonomies.

7.5 Classifiers

The next issue to examine was the relative performance of the k-nearest neighbour (KNN) classifiers that were used to perform classifications based on one-dimensional features and the neural network (NN) classifiers that were used to classify each multi-dimensional feature. The KNN classifier had the advantage of requiring essentially no training time (although feature selection could be expensive) and the NN classifiers had the advantage of being able to model relatively sophisticated relationships between features and categories.

Figure 16 seems to indicate that the NN's alone performed slightly better than a KNN classifier alone, although the uncertainty associated with the measurements makes this difficult to say with absolute certainty. Also, the KNN and NN classifiers use entirely different features, so the results here are linked to both which classifiers were used and which features they were given to base classifications on. In any case, it does seem apparent from that the combination of a KNN classifier and NN's operating in conjunction do perform better than either type of classifier operating alone.

One surprising result was that basic KNN classification performed better when combined with NN's than KNN with feature selection combined with NN's. This result should be treated with caution, as there was a relatively large error associated with experiment P, but it still bears investigation.

This result may also be due to the effect of round robin and hierarchal ensembles immediately achieving perfect fitnesses with non-optimal feature selections during training, as discussed in Section 7.4. This would make most of the classifiers unable to benefit from feature selection. Indeed, it is even reasonable to suspect that, in circumstances such as this, performance could be poorer with feature selection that without, as perfect results during training could cause the system to settle on a selection of features less good than even the entire feature set.

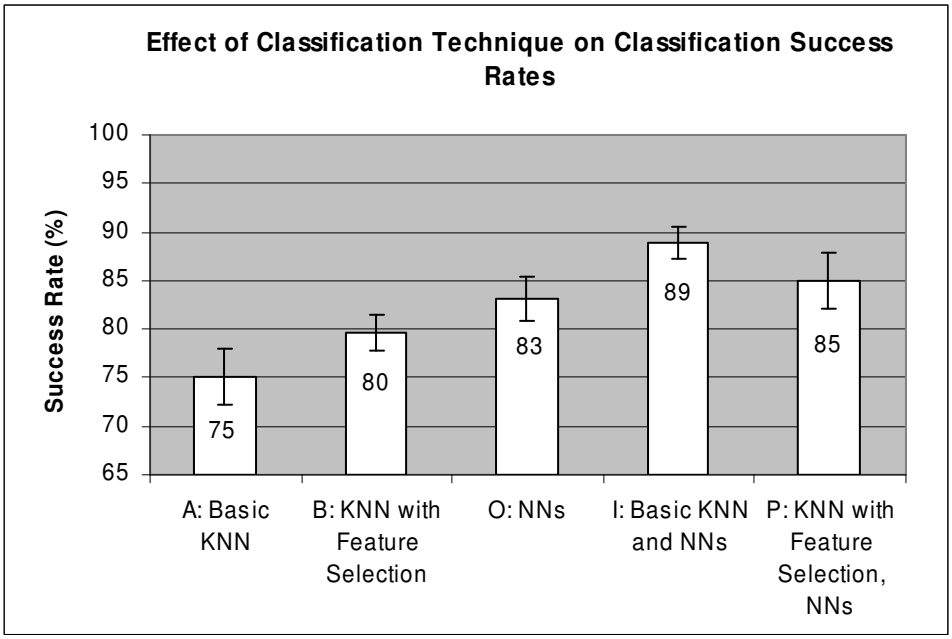

Figure 16: Effect of different classification methodologies within a classifier ensemble on the classification of Taxonomy T-9. Values are averages over all folds and error bars correspond to standard error. Letter codes identify the experiments (see Table 2).

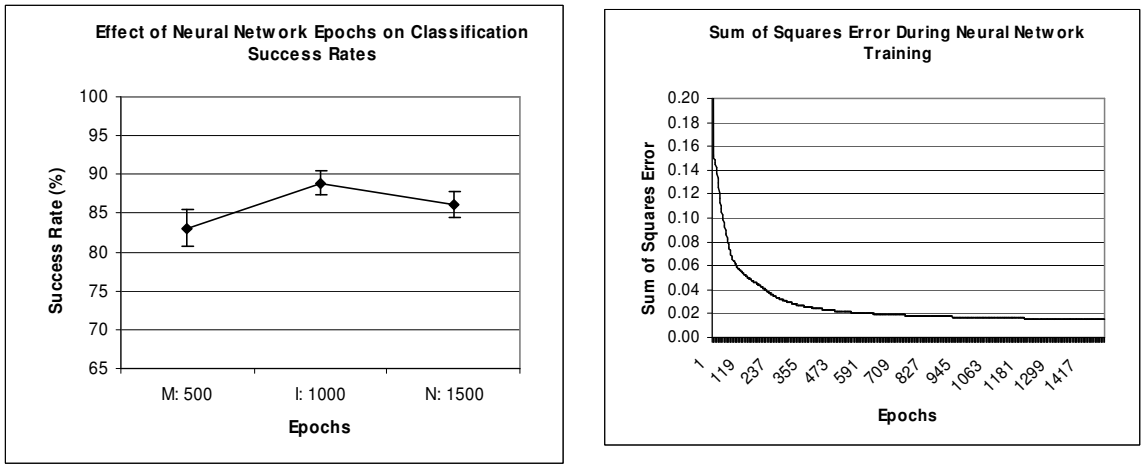

Figure 17: Effect of maximum number of neural network training epochs on success rates. The left plot gives the average success rates as a function of number of epochs of Experiments M, I and N. The right plot gives the sum of squares error of a typical *Pitched Instruments Present* NN classifier in a flat leaf classification ensemble in Experiment N.

Aside from this, it should be noted that Experiment P had a much lower over select rate than Experiment I (6 % compared to 10%), which does give at least some support for the use of feature selection. On the other hand, Experiment I only required 25.9 minutes of training time, whereas experiment P needed 53.3 minutes. In any case, it appears that either technique works well, and that it is better to combine KNN and NN's in a classifier ensemble than to limit oneself to only one classifier type. Although using a simple KNN classifier with no feature selection (Experiment A) does have the advantage of requiring essentially no training time, including NN's in the experiment as well increased the average success rate by 14%, a very significant amount, at a cost of only 25.9 minutes of training time.

Figure 17 explores the effects of varying the number of neural network training epochs on success rates. Although the plot on the left shows that the best results were achieved with 1000 epochs, there is enough uncertainty to make it possible that the classifiers trained using 1500 epochs could in fact perform better or comparably. An examination of the right plot shows that the network stopped rapidly converging after 600 or so epochs, but that small improvements were still being made right up to the 1500 epoch training termination point. This indicates that further improvements, although probably relatively minor, could be achieved by further training. This is supported by the significantly better secondary success rate and over select rate achieved in Experiment N (1500 epochs) compared to Experiment I (1000 epochs), as can be seen in Table 4. There is therefore reasonable reason to use larger numbers of training epochs, as long as training time is not critical (Experiment N took 36.9 minutes to train, but Experiment I took only 25.9 minutes).

7.6 Classifier ensemble coordination

All of the experiments discussed up to this point have used the full complement of hierarchal, flat leaf and round robin classifier ensemble coordination techniques. It is now appropriate to further examine the relative performance of these techniques while holding other classifier parameters constant. Both KNN and NN classifiers were used for these experiments, using feature selection and classifier weighting.

Figure 18 reveals some interesting results. Initially, one might be very excited by the 100% success rate achieved by the round robin only classifier in experiment Q. Unfortunately, this is accompanied with an average over select rate of 96%, which is entirely unacceptable. An additional experiment (R) was performed to further investigate this, where the round robin classifiers were only allowed to classify recordings into a single category. The resultant average success rate was 85%, which is not bad, although certainly less impressive than 100%.

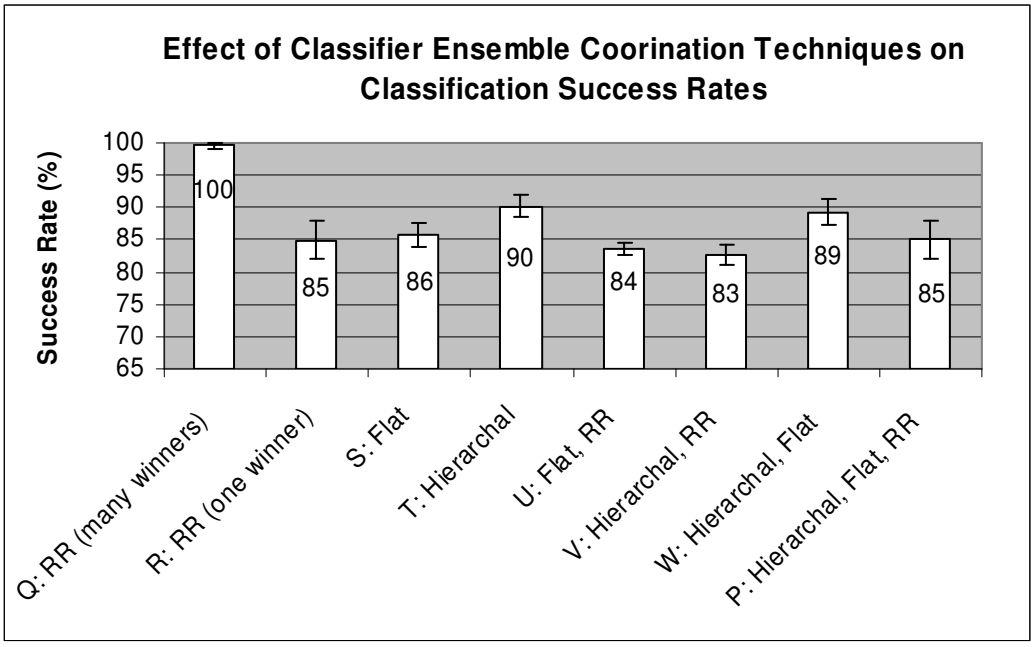

Figure 18: Effect of different classifier ensemble coordination methodologies on the classification of Taxonomy T-9. Values are averages over all folds and error bars correspond to standard error. Letter codes identify the experiments (see Table 2).

The problem here was that each round robin classifier was trained only with recordings belonging to its candidate categories, as is standard practice with round robin classification. When faced with a recording belonging to neither of its two candidate categories during test classification, a round robin classifier usually output a high score for at least one of the candidates. This is not at all a problem if one only wishes to have one winning category, since one can simply choose the category with the highest average score, without worrying about how high the other scores are. If one can have a variable number of correct classifications, however, then matters become more difficult, as one cannot assume that just choosing one winner is sufficient. The differences between the scores of all categories when the scores for all round robin classifiers for a recording were averaged were fairly small, thus making it difficult to decide how many categories should be chosen. Even though the highest score was often correct, which is why experiment R had good results, this is not much help in cases where there can be multiple correct categories or where one wishes to have the possibility of having labels of "unknown" assigned to wrongly classified recordings rather than an incorrect label.

It was therefore decided to set all scores but the highest from all round robin classifier ensembles to 0 when round robin classification was used with other techniques. This was done on all experiments presented in Tables 2 to 5 (except Q, of course). This had the effect of boosting the score of one particular category for each recording, and thus

making it more likely to be chosen by the hierarchal and flat classifiers. This leads one to suspect that this would have the overall effects of improving the number of classifications where at least one category was correct and of bringing down the over select rate. Unfortunately, this could also have the less desirable effects of lowering the number of failed classifications that were assigned unknown labels rather than erroneous labels and of causing the classification system to be more likely to miss one or more of the categories of recordings with multiple correct classifications.

So, round robin classification as implemented here is inappropriate for use by itself if one wishes to have a variable number of categories assigned to each recording, and its use in conjunction with other forms classifier ensemble coordination has both strengths and weaknesses. One possible way of getting around the problem discussed above would be to train round robin classifier ensembles on all training recordings, thus making them accustomed to giving negative results as well as positive results. This, unfortunately has the disadvantage of potentially drastically increasing training time, as a single round robin classifier would thus require almost as much training time as a leaf classifier. This is problematic for large taxonomies, as a taxonomy of n categories requires the following number of round robin classifiers, which grows quickly:

$$ {}_{n}C_{2} = \frac{n!}{2(n-2)!} \tag{14} $$

Looking at all of the experiments in Figure 18, it can be seen that hierarchal classification by itself and hierarchal classification combined with flat leaf classification had the best average success rates. Unfortunately, these two experiments also had the highest average over select rates, of 21% and 19% respectively, both significantly higher than experiments U, V and P. In fact, an examination of Figure 19 shows that those experiments that combined round robin classification (now only choosing one winning category) with other techniques had significantly lower average over select rates than experiments that did not include a round robin classifier. This confirms the suspicions expressed above that the use of round robin classification along with other techniques lowers the overall success rate (when dealing with a variable number of correct categories per recording), but also decreases the over select rate.

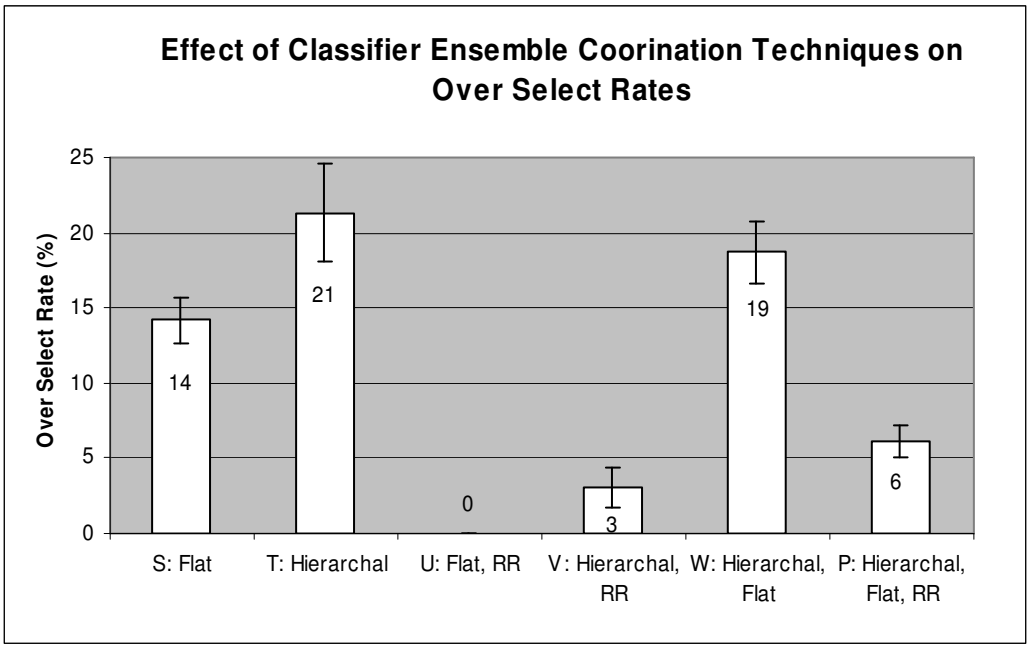

Figure 19: Effect of different classifier ensemble coordination methodologies on over select rates with Taxonomy T-9. Values are averages over all folds and error bars correspond to standard error. Letter codes identify the experiments (see Table 2).

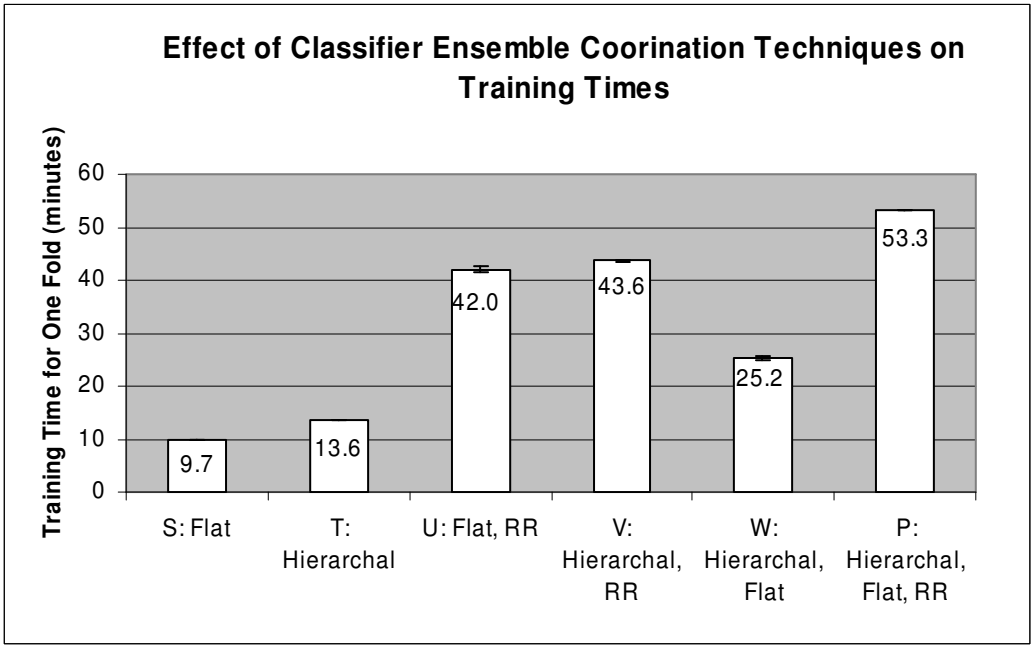

Figure 20: Effect of different classifier ensemble coordination methodologies on training times with Taxonomy T-9. Values are averages over all folds and error bars correspond to standard error. Letter codes identify the experiments (see Table 2).

Which coordination method is the best to choose likely depends on one's priorities. If one is willing to tolerate classifications that may include some false positives in addition to the correct category(ies), then simple hierarchal classification is probably the best choice, since experiment T achieved a success rate of 90%, and required only 13.6 minutes of training time. This training time is very good in comparison to other techniques, as can be seen in Figure 20. The fact that some extra categories will be found may not be as critical as one thinks, as they will be likely to be similar to the correct categories, as exhibited by the 98.2% root rate for experiment T (see Table 4). The use of combined hierarchal and flat classification may also be appropriate if one does not mind some false positives, since the success rate and over select rate were comparable for Experiments T and W, but Experiment W had an excellent secondary success rate of 54%, far higher than any other experiment. However, experiment W also required almost twice the training time of Experiment T.

If one considers a high over select rate unacceptable and does not mind some increased training time, then the combination of hierarchal, flat and round robin classification is likely the best option. Experiment P achieved a very respectable success rate of 85%, and had a relatively small over select rate of 6%. P did require an average of 53.3 minutes of training time, however, so flat classification combined with round robin or hierarchal classification combined with round robin would be good choices for someone trying to cut down a little on training time, as they also have reasonable success rates and over select rates.

As a final note, the exclusively flat leaf classification of experiment S obtained a 15% unknown rate, which is at least 3 times as high as the unknown rate for any other experiment with T-9. This seems to indicate that flat leaf only classification is an appropriate choice if one is particularly concerned about incorrect classifications and wishes to ensure that erroneous classifications are more likely to be marked as unknown rather that given incorrect labels.

7.7 Benchmark classification for 9 leaf taxonomy

Upon reviewing the above sections, it seems apparent that different classification settings are better suited to different needs. It is desirable, however, to have a single general purpose classifier to benchmark the system and present as a final overall result for Taxonomy T-9. It was decided to choose the best overall configuration, and then train it with a greater number of iterations than had been used previously in order to arrive at the benchmark. This classifier configuration is described under Experiment Z in Table 2. This particular configuration was chosen because it coincided with the classifier parameters revealed in the previous experiments to give the near best success rate while at the same

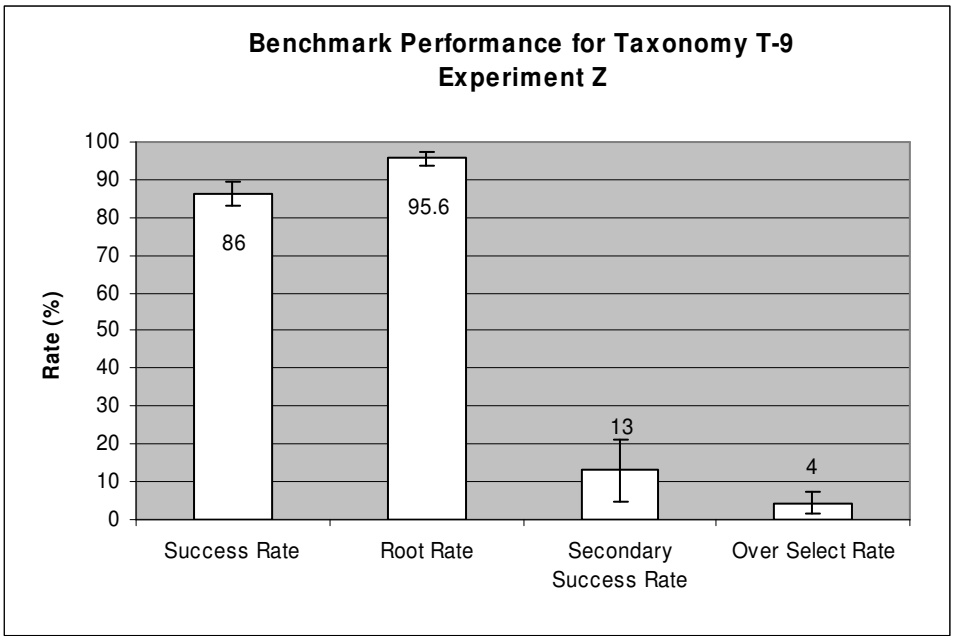

Figure 21: Performance of benchmark classifier for Taxonomy T-9 (Experiment Z). Values are averages over all folds and error bars correspond to standard error.

time minimizing the over select rate and not requiring an excessive amount of training thime. The results of this benchmark can be seen if Figure 21.

These results were quite good. Even though earlier experiments achieved superior success rates up to 90% (Experiment T), root rates up to 99.1% (Experiment W), secondary success rates as high as 54% (Experiment W) and over select rates as low as 0% (Experiment U), the results for Experiment Z characterize the best overall performance. The training time of 89.5 minutes for Experiment Z, although higher than that of previous experiments because of the additional GA and NN iterations used, is nonetheless more than small enough for practical purposes.

The particular kinds of errors made during classification are shown on Table 6, the confusion matrix for Experiment Z. Figure 22 shows the relative success rates for recordings belonging to different categories in Experiment Z. The distributions of classifications shown here are generally consistent with the distributions found in other experiments.

	Bebop	Jazz Soul	Swing	Rap	Punk	Country	Baroque	Modern Class.	Romantic
Bebop	**80±6**	16±7	0±0	0±0	0±0	2±2	0±0	2±2	0±0
J. Soul	14±4	**67±6**	0±0	12±9	0±0	10±4	0±0	4±4	0±0
Swing	6±6	3±2	**100±0**	0±0	0±0	0±0	0±0	0±0	0±0
Rap	0±0	2±2	0±0	**80±9**	0±0	0±0	0±0	0±0	0±0
Punk	0±0	2±2	0±0	0±0	**100±0**	0±0	0±0	0±0	0±0
Country	0±0	5±4	0±0	4±4	0±0	**88±5**	0±0	0±0	0±0
Baroque	0±0	0±0	0±0	4±4	0±0	0±0	**92±5**	2±2	0±0
M. Class.	0±0	0±0	0±0	0±0	0±0	0±0	0±0	**70±9**	16±12
Romantic	0±0	4±4	0±0	0±0	0±0	0±0	8±5	22±10	**84±12**

Table 6: Taxonomy T-9 confusion matrix for Experiment Z. The columns correspond to the model classifications and the rows correspond to how the test recordings were actually classified. Values are average percentages across all five folds of the experiment. Percentages are normalized for each leaf category. The boxes identify groups belonging to the same root categories.

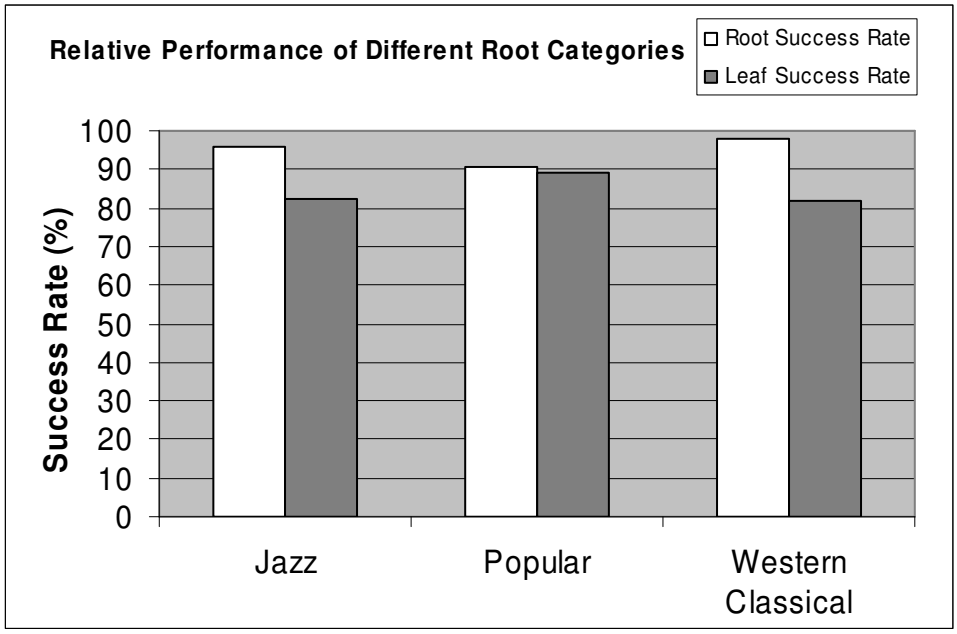

Figure 22: Success rates for Experiment Z with Taxonomy T-9**.** Root Success Rate gives how often a recording belonging to the given root category was assigned leaf label(s) with the correct root ancestor(s), and Leaf Success Rate gives how often a recording belonging to a given root category was assigned the correct leaf label(s). Values are averages over all folds.

As can be seen in Table 6 and Figure 22, recordings that were assigned erroneous labels were usually at least classified as belonging to leaf genres with the correct roots, except in the case of Popular Music. Popular Music recordings were most often assigned the correct leaf labels, however. As can be seen in Figure 22, Jazz, Classical and Popular recordings were classified with fairly similar success.

Swing and Punk recordings were unfailingly classified correctly. Jazz Soul and Modern Classical performed the worst, with success rates of 67% and 70% respectively. The performance of Modern Classical was not overly concerning, however, since 77% of the misclassified Modern Classical Recordings were classified as Romantic, which can be quite similar to Modern Classical in some cases. Although 19% of the Jazz Soul recordings were classified as Bebop or Swing, which is not too worrying, 18% of Jazz Soul recordings were classified as dissimilar categories. The relatively poor performance of Jazz Soul is perhaps not surprising, however, as this genre does have much in common with other jazz genres as well as some popular genres, so there it has a fairly high amount of overlap with other categories that could confuse the classifiers. Aside from this, however, performance was excellent overall with Experiment Z in particular and Taxonomy T-9 in general

7.8 Relative performance of different features

A detailed analysis of which features were most useful in different contexts is beyond the scope of this thesis, but could be of great musicological interest, and is therefore worth pursuing in the future. Automatically produced records were kept of the selections and relative weightings assigned to each feature and classifier in all of the experiments discussed in this chapter, and are available for future research. These may be of limited utility with respect to Taxonomy T-9, however, since, as discussed earlier in this chapter, the specialized round robin and hierarchal classifiers performed so well in training with such a wide variety of feature sub-sets that the particular ones selected were probably not near-optimal. The results for Taxonomy T-39 were slightly better in this respect, however, as the categories there were more similar, thus making feature selection more important and forcing the genetic algorithms to work harder to find good solutions. Future experiments where round robin and hierarchal classifiers are trained on all recordings, not just ones belonging to their candidate categories, could provide more musicologically interesting feature selections and weightings for both taxonomies.

In general, it can be seen from an informal examination of the feature selection and weightings discussed in the above experiments that the classifiers made significant use of features belonging to all of the feature classes. The instrumentation features were particularly useful, however, as these features were assigned 42% of the weight in , for example, experiment Z, even though they only comprised 18% of the candidate features.

7.9 Experiments with the 38 leaf taxonomy

Once it was established that the software developed here was able to perform very well with the simplified T-9 taxonomy, it was then appropriate to test it on the more difficult T-38. The experiments performed are described in detail in Table 3 (p. 104) and the results are presented on Table 5 (p.105). Experiments AA and BB were the first ones performed, as they required essentially no training time, since only simple KNN classification was used with no feature selection or weighting. Success rates of 37% with only flat classification and 43% with hierarchal, flat leaf and round robin classification were achieved, which were less than ideal. Of course, this was not surprising given the complexity of the taxonomy and the limited classification techniques used. It was therefore necessary to use more sophisticated classification methodologies.

Training time was much more of a concern here than with Taxonomy T-9, as the full library of 950 recordings was used for training and testing, thus requiring greater time for each training iteration, and the larger taxonomy also required a greater number of hierarchal and round robin classifiers (18 and 703 respectively, as opposed to 4 and 36 for T-9). Training time concerns were particularly significant, given that 5 complete folds needed to be performed for each experiment. All of this meant that the level of experimentation performed with Taxonomy T-9 was not feasible here, and that limitations had to be placed on the techniques used in order to keep training times reasonable.

Several experiments were performed with hierarchal and/or flat classification, with and without feature selection. Although the differences in GA and NN iterations in these trials limits the direct comparisons that can be made of these experiments, the differences are not so great so as to prevent some rough conclusions from being drawn. The particular classifier configurations chosen were based on the results of the experiments with T-9. Although it is not valid to say that all of the results from these experiments necessarily generalize to T-38, they did provide a good starting point.

Round robin classification was rejected immediately, as the 703 round robin classifiers took up to a matter of days to train. Furthermore, the use of round robin classifiers caused classification times, which had previously been effectively negligible, to become perceptible. In one test in Experiment BB it took almost 3 minutes to classify the 190 testing recordings, something that is less than ideal in real world situations where the system should be able to deal with multiple requests rapidly without causing users to become impatient.

Experiments CC, EE and FF all performed comparably with regard to success rate, and Experiment DD performed somewhat worse, as can be seen on Figure 23. The differences in training time were much more significant, however, as can be seen in Figure 24. Experiment DD took a great deal of time to train, but training times for CC and FF were comparatively small.

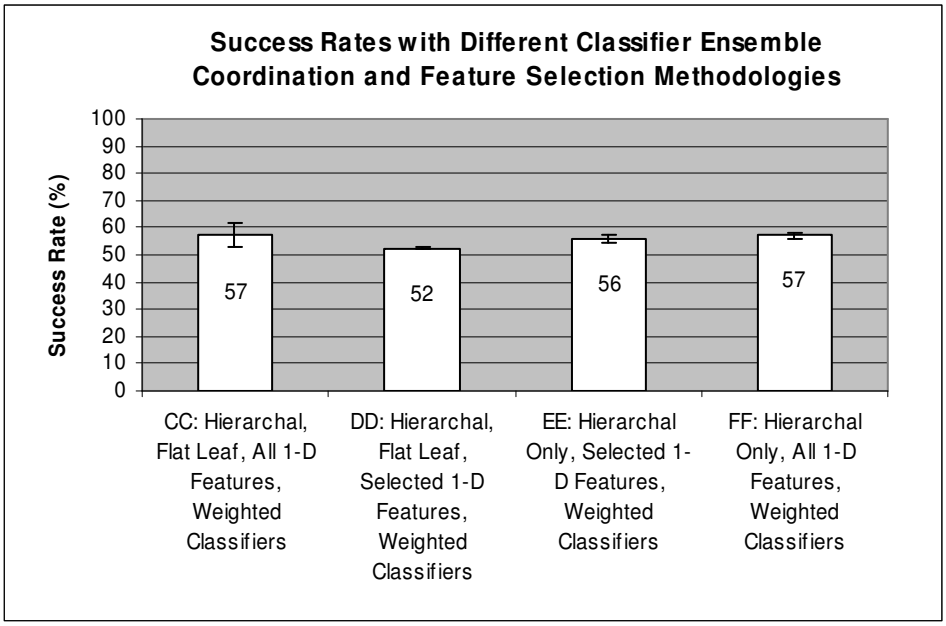

Figure 23: Effect of classifier ensemble coordination and feature selection techniques on success rates for Taxonomy T-38. Values are averages over all folds and error bars correspond to standard error. Letter codes identify the experiments (see Table 3).

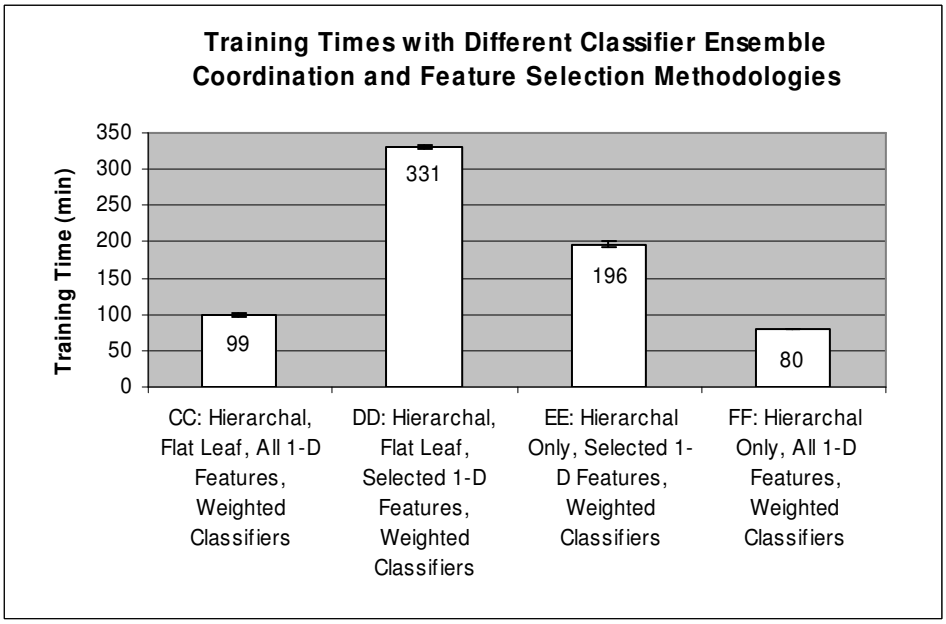

Figure 24: Effect of classifier ensemble coordination and feature selection techniques on training times for Taxonomy T-38. Values are averages over all folds and error bars correspond to standard error. Letter codes identify the experiments (see Table 3).

These results seem to indicate that the configuration used in Experiment DD is a poor choice. Beyond this, however, the other results are fairly similar in terms of average success rates. A further examination of Table 5 indicates, however, that experiments CC and DD performed much better than EE and FF in terms of their over select and unknown rates, and worse in terms of their root rates.

These overall results seem to make Experiment CC stand out as the best configuration of those experimented with. Its success rate was as good as any of the configurations and its training time was 99 minutes, a very small amount for such a large taxonomy. Furthermore, its over select rate of 21% was better than the other three options, although still uncomfortably high, and its unknown rate was the best of the group at 32%, a very good value. Although the root rate of 74.9% was less than the values of 81.2% and 80.6% for Experiments EE and FF respectively, this is a reasonable sacrifice to make in exchange for the benefits discussed above.

It is interesting to note that Experiment CC, with no one-dimensional feature selection, performed better than Experiment DD, which did use feature selection but was otherwise identical. One would expect hierarchal classifiers with feature selection to perform better, not worse. An informal examination of the training records from Experiments DD and EE, where feature selection was performed, helped to shed light on the several causes of this.

To begin with, the same problem occurred here with some hierarchal classifiers that occurred with taxonomy T-9, namely that some of the hierarchal classifiers were able to perfectly classify the training recordings belonging to their candidate categories with a wide variety of feature sub-sets, and thus had perfect fitness during training and therefore did not evolve an effective feature selection. During actual testing, these classifiers were exposed to recordings that did not in fact belong to any of their candidate categories. Ideally, in situations such as this, a good feature set would have been evolved which would reject such recordings. However, since almost any feature set worked during training, such an optimized feature set was not available, and the classifiers behaved unpredictably when given recording belonging to categories that they had not been exposed to during training.

The solution to this problem, as discussed earlier, would be to train the hierarchal classifiers with recordings of all types, not just ones that belong to their particular candidate categories. This would, for example, turn an (effectively) two-category classifier such as one with candidate categories of Bebop and Cool into a three-category Bebop / Cool / neither classifier, thereby increasing the difficulty of the classification task and hopefully forcing the evolution of a good feature selection. This would, however, have the negative effect of greatly increasing training time.

Fortunately, this problem was not as prevalent here as it was with T-9. A number of the hierarchal classifiers had fine enough distinctions between their candidate categories that distinguishing between them was somewhat difficult. This meant that a random feature subset was not sufficient to provide perfect fitness, so the GA's were forced to evolve better feature selections, which made the classifiers perform better during testing. An additional problem, however, was that there were not enough generations of evolution performed, as the fitness had often not yet converged once the maximum generations of 30 or 35 were reached. This low maximum on generations was imposed in order to cut down training times, as feature selection was the most time consuming process during training. Unfortunately, this may well have severely limited performance. This, combined with the fact that some of the more general hierarchal classifiers still suffered from too easy success during training and therefore led to incorrect classification paths down the hierarchy, probably explains why the experiments shown in Figure 24 that did not use any feature selection performed better than those that did.

If one looks at the results for T-38 in the context of those for T-9, the T-38 performances seem quite mediocre at first. The success rate did not exceed 57% for any of the experiments, and the over select rate was over 20% for all experiments that achieved a success rate over 43%. However, respectable root rates between 71.5% and 81.2% were achieved (compared to a chance rate of 11%), along with decent secondary success rates of 11% to 22% and impressive unknown rates of up to 32%. Furthermore, results of this order were achieved with training times as low as 80 to 99 minutes, which is very impressive for a taxonomy of this size.

Even the success rates of 56% to 57% are not insignificant when it is considered that the random chance of success is only a tiny 2.6% even if one is limiting oneself to one correct classification per genre. The probability of randomly guessing the correct genres when there could be more than one is significantly lower even than this. Furthermore, to put these results in context, Tzanetakis, Essl and Cooks' groundbreaking and often cited automatic genre classification research (2001) of only three years ago achieved a success rate of 62% with only 6 genres, a far easier task than dealing with 38 genres. Since no one, to the best of the authour's knowledge, has attempted to work with a taxonomy of anywhere near the size or difficulty of T-38, the fact that success rates of 56% to 57% were achieved is actually somewhat encouraging, and leads one to hope that future research could well significantly improve these results.

If one is willing to accept training times that span one or more weeks, even the system used here could potentially achieve significantly improved success rates without any modifications. Simply increasing the number of NN epochs and GA generations and/or population could cause gains. In addition, training the hierarchal classifiers on all training recordings could potentially greatly increase performance. The inclusion of round robin

classifiers, also trained using all recordings, could also significantly improve results, although this would cause training times to increase by a factor of 37 in the case of T-38.

So, the results for T-38, although not suitable for practical applications, are nonetheless quite encouraging from a research perspective, and hold a great deal of potential. The first future step towards exploring this would be to perform tests that require weeks of training time, something outside of the timeframe of this thesis, but not at all unreasonable, when it is considered that humans take years to learn the knowledge necessary to distinguish between genres, and even then they don't always perform very well. There are also numerous other techniques and approaches that could improve results in the future, as discussed in Section 8.3.

8. Conclusions and Future Work

8.1 Conclusions and summary of experimental results

Overall, the results of this study were quite encouraging, both from a purely performance based perspective as well as from a theoretical perspective. The benchmark experiment with the 9 category taxonomy was, on average, able to correctly classify recordings 86±3% of the time by leaf genre, and found the correct root genre 95.6±1.9% of the time. Even in those cases where a misclassification occurred, the correct category was given by the system as a secondary choice an average of 13±8% of them time. In addition, those errors that were made tended to be reasonable (e.g. Bebop was occasionally misclassified as Jazz Soul, but never as Baroque).

These numbers are particularly impressive, given that recordings were permitted to have a variable number of model categories. This made the classification task much more difficult than simply choosing a single winning category per recording without needing to also determine the number of categories to select as well. Even with this added difficulty, recordings were only assigned extra categories that they did not belong to an average of 4±1% of the time. Furthermore, an average of only 89.5±0.3 minutes of training time were required on a personal computer, a very small amount of time.

To put these results in perspective, it is important to realize that the average Western human would probably have difficulty achieving success rates this high. In an experiment involving listening to three seconds of audio recordings belonging to one out of ten genres (blues, country, classical, dance, jazz, latin, pop, R&B, rap and rock), college listeners were only able to perform correct classifications only 70% of the time (Perrot & Gjerdigen 1999). Although these numbers are not directly comparable to the results of the system described in this thesis, as different taxonomies were used and audio data was examined rather than high-level musical representations, this study nonetheless puts success rates in context. If relatively educated humans with training periods of years can only achieve success rates of 70%, then a success rate of 86% after only 89.5 minutes of training is exceptional.

Although direct comparisons with existing automatic classification systems that have used entirely different taxonomies, recording libraries and file formats must also be treated with some caution, some loose comparisons can help to put these success rates in context. A review of Chapter 4 will show that only one of the audio genre classification systems to date dealing with more than five candidate categories have achieved success rates above 80%. Pye (2000), who correctly classified recordings 92% of the time into one of six candidate categories, is the single exception. As far as classification systems that use symbolic data go, the best known previous results are 84% for two-way

classifications (Shan & Kuo 2003) and 63% for three-way classifications (Chai & Vercoe 2001).

When it is considered that increases in taxonomy size are well known to greatly increase classification difficulty, and that the system presented here was the only one forced to deal with the added complication of recordings having variable numbers of correct classifications, the result of 86% with 9 candidate categories is quite impressive. Of course, it is important to stress once again that one cannot make direct judgements as to whether one system is better than another without tests involving the use the same taxonomy and testing recordings. Nonetheless, it is clear that the rate of 86% achieved here is a very good result, compared to both humans and existing automatic genre classification systems.

An important implication of this success is that it clearly demonstrates the potential of high-level features, which could be applied to a wide variety of musical classification tasks beyond the scope of genre classification. High-level features could be used in conjunction with low-level features in future research to improve overall performance of audio classification systems, and could also be of great use if one wishes to classify scores for which no audio recordings are readily available.

The results for the 38 leaf category taxonomy were also encouraging. The best overall configuration of the system that was experimented with was able to correctly identify the leaf genre(s) of the 38 leaf taxonomy with an average success rate of 57±5% (with extra categories selected 21±1% of the time). This configuration also had a success rate of 74.9±0.1% when classifying recordings into one or more of the 9 root genres.

These numbers are impressive given the very large size of the taxonomy and the fact that the random chance of success is only a tiny 2.6% even if one is only limiting oneself to one correct classification per genre. The probability of randomly guessing the correct genres when there could be more than one is significantly lower even than this. Furthermore, to put these results in context, Tzanetakis, Essl and Cooks' (2001) groundbreaking and often cited automatic genre classification research of only three years ago achieved a success rate of 62% with only 6 genres, a far easier task than dealing with 38 genres. Since no one, to the best of the authour's knowledge, has attempted to work with a taxonomy of anywhere near the size or difficulty of the 38 leaf category taxonomy experimented with here, the fact that a success rate of 57% was achieved is actually quite encouraging. The subsequent improvements to Tzanetakis, Essl and Cooks' system (see Chapter 4) lead one to hope that similar future improvements will be achieved here as well.

It is also encouraging to note that the incorrectly classified recordings were assigned a label of "unknown" an average of 32±1% of the time rather than some erroneous label. This is very important, as it can act as a flag for human operators, and is much better than

just giving recordings wrong labels that can be difficult to detect. Furthermore, the correct genre was specified an average of 13±1% of the time as a secondary choice when it was missed as a primary choice. Finally, and quite impressively for such a large taxonomy, these results were achieved after an average of only 99±2 minutes of training on a personal computer.

Although the 57% success rate is still too low for practical applications, it is quite impressive from a theoretical perspective as a groundbreaking rate for such a difficult taxonomy, and holds a good deal of potential for future improvement. Even beyond the significantly increased classification difficulty of requiring that the system be able to classify recordings into a variable number of categories and be able to classify recordings as unknown in some cases rather than just giving an erroneous classification, there were a number of other factors that made the classification task performed here particularly hard.

Time limitations imposed a limit on the number of training recordings that could be collected, with the result that only 20 recordings were available per leaf category after some recordings had been reserved for testing. This is not a very large number to learn to recognize genres with, particularly considering the far larger number of recordings that humans are exposed to in their lives. The use of MIDI itself also proved to be a challenge, as important features such are lyrics are not consistently available, and therefore had to be ignored here, and most recordings are sequenced in an amateurish and inconsistent manner.

The 38 leaf category taxonomy also included more categories than the average person (although not music professional) would likely be well acquainted with, and also contained some genres where the differences can be primarily sociological for some recordings (e.g. Alternative Rock and Punk), leaving a limited amount of content-based differences for the automatic classifier to work with. Furthermore, the categories were not necessarily structured logically from a content-based perspective (e.g. Modern Classical and Medieval music should be and were grouped together, despite the fact that they have little in common from a content-based perspective), as was necessary to accurately simulate real conditions, with the result that this complicated matters significantly for the hierarchal classifiers.

Aside from the actual performance of the system, the results of the experiments described in Chapter 7 led to some interesting observations. To begin with, it was found that the inclusion of increasing numbers of candidate features for the feature selection system to choose from significantly improved classification performance. This leads to the conclusion that the implementation of even further features could be beneficial, despite the predictions of the curse of dimensionality, whose effects were probably alleviated by the feature selection.

It was also observed that the neural network classifiers and their associated multi-dimensional features worked slightly better than the KNN classifiers and their associated one-dimensional features. Overall, however, the results of both types of classifiers acting together in a classifier ensemble were better than either operating independently.

In addition, it was found that the genetic algorithms of some round robin and hierarchal classifiers converged immediately to significantly sub-optimal results during feature selection and weighting rather than gradually evolving optimal or near-optimal feature sub-sets. It was discovered that this was due to the classifiers being able to perfectly classify their training recordings using a wide variety of sub-optimal feature selections, with the result that even poor selections had perfect fitness, thus giving the genetic algorithms no reason to evolve better selections. As is typically done with round robin and hierarchal classifiers, the classifiers were only trained on examples belonging to their candidate categories. During actual testing, these classifiers were exposed to recordings that did not in fact belong to any of their candidate categories. Ideally, in situations such as this, a good feature sub-set would have been evolved which would reject such recordings. However, since almost any feature sub-set had worked during training, such a good feature sub-set was not available, and the classifiers behaved unpredictably when given recordings of types that they had not been trained on. Essentially, the classifiers were doing their job too well during training for selection and weighting to work effectively during testing.

A potential solution to this problem would be to also train these classifiers on recordings that do not belong to their candidate categories so that they could learn to reject them. This would make the classifiers' task more difficult by forcing them to learn to deal with more types of recordings, thus increasing the probability that a poor feature selection or weighting would not work as well, thereby forcing the genetic algorithms to evolve better solutions. The downside of this approach is that it could greatly increase training times, especially in cases of taxonomies with many leaf categories that would require a very large number of round robin classifiers that would now each have long training times.

It turned out that this problem was less evident with the hierarchal classifiers of the large taxonomy than it was with the smaller taxonomy. This is probably because the candidate categories for each hierarchal classifier in the large taxonomy were often quite similar, thus making the corresponding recordings more difficult to classify, thereby making the classifiers unable to successfully classify training recordings with arbitrary feature selections and therefore forcing the genetic algorithms to evolve better feature selections and weightings. In any event, it was clear that it was important to take steps to ensure that the round robin and hierarchal classifiers were not able to converge to perfect fitness with significantly sub-optimal selections and weightings.

It was also observed that round robin classifiers can be problematic in situations where recordings can belong to a variable number of candidate categories. This is because there tends to be little difference between the average scores of all categories across all round robin classifiers, thus making it difficult to decide how many are correct. This is not a problem if one only needs a single classification for each recording, as one can simply choose the category with the highest average score, which usually corresponds to the correct category, and it does not matter if the score of this category is only slightly higher than the scores of the other incorrect categories. This is a problem when dealing with a variable number of possible correct classifications, however, since the small differences between the scores of different categories produced by round robin classifiers make it difficult to decide on the correct number of categories to choose.

One solution would be to train round robin classifier ensembles on all training recordings, thus making them accustomed to giving negative results as well as positive results, thereby lowering the overall averages of incorrect categories. This, the same solution as the solution proposed above for improving feature selection, has the same disadvantage of potentially drastically increasing training time.

An alternative solution, and the one used here, was to simply choose one winner for the round robin classification, thereby bypassing this problem, but allowing the hierarchal and flat leaf classifiers to select multiple winners, if appropriate. This had the effect of lowering the overall success rate, since it made it more likely that recordings belonging to multiple categories would have fewer of these categories selected by the classifiers, but also decreased the amount of additional incorrect categories that were assigned to recordings.

A study of different classifier ensemble coordination methods also found that which method is the best to use depends on one's priorities. If one is willing to tolerate some false positives in addition to the model category(ies), then simple hierarchal classification is probably the best choice. The use of combined hierarchal and flat classification may also be appropriate if one does not mind false positives and is willing to increase the training time in exchange for a better secondary success rate. If one considers false positives to be a particularly unacceptable type of error and does not mind a significantly increased training time, then the combination of hierarchal, flat and round robin classifiers is likely the best option. Flat leaf classification alone is an appropriate choice if one is particularly concerned about false classifications and wishes to ensure that erroneously classified recordings are more likely to be marked as unknown rather that given incorrect labels, and does not mind sacrificing overall success rates a little.

Overall, testing results showed that the system performed very well with a taxonomy consisting of 9 leaf genres. The results also showed that, although the system cannot yet deal with significantly larger taxonomies well enough for practical application, some

respectable success is already possible, and there is great potential for future improvement, as discussed in Section 8.3. This research also clearly demonstrated that high-level features can be very successful as a basis for automatic classification.

8.2 Research contributions of this thesis

This thesis attempted to classify recordings into a much larger, more sophisticated and more realistic taxonomy than has been used in any known previous research on automatic genre classification. Far more categories were used than had been previously, and complications such as the possibility of recordings belonging to more than one category, recordings not belonging to any categories, and categories having more than one parent were incorporated into the taxonomy and classification methodology in order to make it more realistic. Realistic genre categories were also used, many of which were similar to one another. Realistic training examples were used as well, including ones that were in many ways atypical of the genres that they belong to. All of this made the classification done here far more difficult than any attempted previously, making this the first system to approach genre classification as more than a toy problem.

A large library of MIDI files was collected and manually classified based on genre. This library could be used in future systems as a test bed comparing different systems.

This research was the first wide-ranging investigation of the problem of musical genre classification in general based on symbolic data rather than audio data. There certainly have been some very interesting papers published on classification using symbolic data, but all of them used a much narrower range of features than those used here, and did not take full advantage of the types of features that can be extracted from symbolic data.

The feature library designed and implemented here provides a very diverse and useful resource for analyzing music, both for the purposes of classification and for other types of analysis. Although there certainly has been a significant amount of research on computerized analysis systems, this is the largest and most diverse one known to the authour that analyzes and characterizes music in an overall statistical sense rather than trying to derive meaning about the music or its structure. Many of the features that were developed are original, and a number of features that had not previously been presented in connection with one another were also collected from a variety of sources and presented here in a united form. The feature library presented here could be made use of in a diverse range of research projects.

Feature selection and weighting techniques were used to experimentally determine which features were most significant in different classification contexts. When combined with the hierarchical classification approach that was used, this makes it possible to study the importance of different features with respect to differentiating between different categories, something which is of great musicological interest.

This is the first known implementation of on automatic genre classification that includes a significant and serious discussion of the musicological and theoretical issues relating to musical genre taxonomies rather than treating the task primarily in terms of being an engineering problem only. This thesis also contains the most complete and multidisciplinary known survey of automatic musical genre classification systems to date.

The software was designed to have an easy-to-use and flexible graphical interface and an easily extensible program design. This allows users and programmers of different skill levels to easily adapt the software to their own classification needs with little learning time. The software was also carefully coded and well documented in order to allow more in-depth modifications by those with coding experience. This system was designed so that it could be used for a large variety of musical classification tasks well beyond the scope of musical genre classification, with little or no modification being necessary.

A novel classification system was used here that made simultaneous use of hierarchical classification, flat leaf category classification and round robin classification. The separation of features into one and multi-dimensional types so that ensemble classifiers could be used that took advantage of the relative strengths of KNN and neural networks where they were needed the most was also a novel technique.

8.3 Future research

The genre classification system that was developed here could easily be adapted to tasks such as composer identification, performer identification, mood classification or classification based on time period simply by changing the classification taxonomy and training data. Future experiments with this expansion of scope could explore the multiple ways in which this system could be used.

More sophisticated methods could be developed for displaying the results of classifications in order to make the positions of each recording relative to different categories and of different categories relative to one another intuitively apparent. This has been discussed by Pampalk (2001), Pampalk, Rauber and Merkl (2002) and Pampalk, Dixon and Widmer (2003). Interfaces of the type described by Tzanetakis, Essl, and Cook (2001) could also be investigated further and implemented as part of this system.

The field of literary studies is perhaps the area in which the most work has been done on identifying and classifying genres, and the adaptation of applied research from this field into a musical context could prove useful. Rauber and Müller-Kögler (2001), to cite just one example among many, have done interesting research into automatically analyzing and classifying text documents and then presenting the results using an intuitive GUI.

Further theoretical study of genre in areas outside of music could provide insights as well. The collections of papers edited by Duff (2000) and Grant (2003) are excellent

resources on literary genre and film genre respectively. An expanded study could also be made in the field of classification and labelling theory itself. Lakoff's book (1987) provides an excellent start in this direction, and further research in the fields of psychology and library science could also be useful. Even the way that recipes are categorized in a cook book, for example, or the ways that biologists organize types of life, could be revealing in respect to how humans assign labels and organize categories. Research in this field could help one to construct improved musical genre taxonomies.

The system could be expanded to deal with other types of file formats. A module for analyzing KERN, MusicXML or GUIDO files, or perhaps translating them to MIDI, would expand the types of recordings that could be classified. Further research into extracting high-level information from printed scores and audio data would also be useful.

The system could also be expanded to include features extracted from low-level audio data directly that are not necessarily directly translatable to symbolic terms, but nonetheless help one to distinguish between genres. The work covered in Section 4.2 would certainly prove useful in this respect, as would more general work such as that by Scheirer (2000). Audio data contains the information that most humans actually use to perform content-based classifications, so it would be advantageous to make use of these low-level cues as well as the higher-level musical awareness that trained musicians possess.

Using both symbolic and audio data would make it possible to take advantage of score alignment research in order to match scores to audio recordings and remove noisy transcription and performance errors. This would also give a measure of the amount of deviation from the score in a performance, which could be a useful feature in itself. Perhaps the ideal would be the exploitation of music formats, such as that proposed in the MPEG-21 standard, which can package audio data together with symbolic data describing the score and production parameters of the music. This would make an extremely wide range of features available all in one package.

Data mining techniques could be implemented in order to collect sociocultural features about the performers and audiences of different genres. Metadata such as country of origin, date of composition, composer, performers who have recorded a piece, ethnicity of performers, age of performers, fan demographics, etc. could all prove to be highly revealing. The emergent approach suggested by Aucouturier and Pachet (2003) holds a great deal of promise in theory. The work of Whitman and Smaragdis (2002) would provide a good starting point for this. A module to derive features from lyrics, including meaning, vocabulary used, rhyming scheme and syllabic structure could prove very useful.

As suggested by Tekman and Horascsu (2002), emotional feeling and mood could provide useful features, albeit potentially hard ones to extract. Research on the KANSEI system (Katayose & Inokuchi 1989) as well as other recent research in this area, such as that by Liu, Lu and Zhang (2003) and Li and Ogihara (2003), could prove useful in this respect.

Further high-level features could be implemented as well, including those that were presented in Chapter 5 but have not yet been implemented. Despite the limitations of sophisticated theoretical analytical techniques in relation to genre classification, as discussed in Section 5.1, one should still use whatever information is available. Features derived from these types of analyses could in fact be highly useful for a certain limited number of genres. Harmonic analysis could, for example, be useful for distinguishing between types of Classical music, but one would need a separate module to first, for example, measure the degree of tonality of a recording in order to decide whether it would be appropriate to use such an analysis. A hybrid expert system / supervised learning system could be developed in order to take advantage of sophisticated analysis techniques while avoiding the weaknesses of expert systems discussed in Section 1.5.

If it turns out that sophisticated analytical techniques are too problematic to implement, it could still be useful to implement a relatively rudimentary harmonic analysis system. Work such as that by Raphael and Stoddard (2003) and the techniques used by Rowe (2001) could be useful in this respect. Features derived from chord progressions based on Roman numerals or from chord voicings / inversions could be extracted, for example. Although these analyses might certainly contain errors due to the difficulty and subjectivity of harmonic analyses, and would have limited value in and of themselves, features derived from such analyses could still provide a rough but effective way of distinguishing between genres. Simple statistics based on the intervals of notes above the bass note could prove to be revealing in the context of known chord progressions. The ability to take arpeggios into account as well as vertical chords would also be valuable.

More sophisticated statistical analyses could also be applied to the histogram features described in Chapter 5. The calculation of higher order moments, skew, scatter and excess, for example, could all be useful. Gabura's paper (1965) provides a good starting point for this approach. A more in depth study of the techniques used by ethnomusicologists to compare different types of music could also provide more ideas. It might also be beneficial to use alternative ways of representing pitch, as suggested by Chai and Vercoe (2001).

Further research on more sophisticated features based on sequential information could be useful as well. Phrasing and repetition, in terms of melodies, chord progressions and rhythms, are very important. The degree and regularity at which such patterns are

repeated, as well as their general character, length and register could all be useful features. It would also be good to collect features related to transposition, decoration and inversion of motifs. In order to extract features related to these characteristics, however, it would first be necessary to have an effective phrase detection and segmentation system, something which would be a valuable research contribution beyond just the scope of just classification systems. It would also be useful to make use of a system that could automatically segment different lines in order to derive features related to melodic contour.

The problem of segmentation in general is highly relevant to genre classification, as a single recording can have different parts that are characteristic of different genres. It would be beneficial to have a system that could properly segment such pieces so that each segment could be classified separately, rather than having the features averaged out, which could result in features that are not indicative of any of the correct genres.

There has been some interesting research done on models of how humans code sequential musical information as well as on applications of generative grammars to music, which is one way in which to think of phrases. Stevens and Latimer (1997) present some references on these areas that could provide a good starting point for adapting this research to the purposes of music classification, as well as some further references on the limitations of these approaches. Research on detecting and processing repeating musical patterns, such as that by Hsu et al. (2001), Typke et al. (2003) or Lartillot (2003), could be taken advantage of. Works such as those by Tseng (1999) or Foote (1999) could also be useful in devising a system to extract melodies and segment recordings. An alternative and potentially very interesting approach to extracting features from phrases would be to characterize melodies by fitting them to functions, as was done by Laine and Kuuskankare (1994), in order to search for patterns and then apply functional data analysis techniques. In the case of audio segmentation, Dannenberg and Hu (2002) present several techniques for searching the underlying structure of music that could be used to search for different types of repetition in the music.

Existing research on query by humming systems could also provide a useful resource for the extraction of features based on sequences and phrases. Features could be extracted by collecting and analyzing n-grams relating to melodies, rhythms and chord progressions. There are a number of resources that could be useful in this respect (Agrawal & Srikant 1995; Hsu, Liu & Chen 2001; Selfridge-Field 1998; Uitdenbogerd & Zobel 1998). The work of Shan and Kuo (2003) could also be of use, as it considers the problem in the context of style classification. The work of Yip and Kao (1999) also provides some useful background on melody-based features.

All of the features that were used in this thesis were extracted from entire MIDI files, in order to take advantage of the full data that was available. Given that it is possible for

humans to make genre identifications based on only short segments of data (Perrott & Gjerdingen 1999), it should be possible for similar classifications to be made by computers. It would therefore be interesting to conduct further studies using only short segments of MIDI recordings to see if the system could still perform well. This would be useful from the perspective of real-time classification. It should be noted, however, that humans may make these quick classifications based on timbral data that is not available in symbolic musical representations such as MIDI.

The use of alternative classifiers could also improve results. Support vector machines, for example, would be particularly well suited to the binary round-robin classifiers, and some success has been had with this approach in the past (Xu et al. 2003). Hidden Markov models could also be useful for classifying features dealing with sequential events, such as melodies or chord progressions. Neural networks with feedback could also be used to process sequential features.

There have been a number of studies on the benefits of combining neural networks in ensembles of different types using a variety of methods to coordinate the results (Granitto et al. 2002; Hansen & Salaman 2001; Wanas & Kamel 2001; Wanas & Kamel 2002; Zhou et al. 2001). Network ensembles such as these could be experimented with for the purposes of musical genre classification, and some of the coordination techniques could be adapted to the types of classifier ensembles already used here. Stevens and Latimer (1997) and Crump (2002) present good surveys of literature relating to the application of neural networks to music that could be consulted in order to devise more sophisticated networks.

Non-metric methods such as induction trees could also be used, as they offer the significant advantage of revealing how classifications are performed. Inductive learning could also help eliminate unneeded features. Some interesting work relating to this has been done by John et al. (1994).

Alternatives to genetic algorithms for feature selection could also be experimented with. For example, principal component analysis could be used to reduce the dimensionality of multi-dimensional features, since it is not as important that one be able to judge the musicological importance of their components as it is for entirely different features.

Existing research into alternative ways of deciding which classifiers to group into ensembles, such as that by Zenobi and Cunningham (2001) could be further explored in order to gain ideas as to alternative types of classifier ensembles. Alternative techniques could also be used to coordinate the ensembles as well. Studies such as that by Tax et al. (2000) could be useful in this respect. Blackboard systems are one particularly promising approach, where different classifiers or ensembles of classifiers could be treated as knowledge sources. A good coverage of such systems is presented in Jagannathan et al.'s

book (1989). The advantages of both expert systems and supervised learning could be taken advantage of with this approach. Opitz and Maclin (1999) review and evaluate some promising techniques for different ways in which neural networks and decision tree classifiers can be combined.

A system could be implemented that punishes certain kinds of errors more than others. For example, a misclassification between two similar sub-genres would not be as bad as confusion between Gangsta Rap and Baroque organ fugues, for example. This could be considered in the evaluation of the system as well, rather than only looking at potentially deceptive overall error rates, which may be as influenced by the choice of genre taxonomy as by problems in the systems itself.

Probably the best way to improve performance would be to acquire a larger and better classified library of training and testing recordings. An automated system for finding and downloading files and the acquisition of access to a large database of easily accessible music would be of great use here.

Techniques for automatically selecting the "best" training samples and eliminating inappropriate ones could be applied. Hartono and Hashimoto (2002) have already done some research on this topic, and there is certainly room for more exploration here. Care must be taken not to leave out atypical members of a genre that nonetheless clearly do belong to the genre, however. This means that this approach would require a great deal of training data with well balanced representatives from all sub-types of each genre.

More categories and training samples could also be used to further expand the scope and effectiveness of the system in general. More attention could be given to the selection of the taxonomy and model examples as well. These were, of necessity, designed and chosen by the authour personally. Although external resources were certainly consulted, better results could probably be achieved by forming a panel of experts that could come up with their own taxonomy and model examples. The correlations between the judgements of different panel members could also provide interesting psychological data.

The results of such a study could be used as a large, diverse and standardized testbed for other automatic musical genre classification systems, something that is sorely missing at the moment, making it difficult to compare different systems. The work of Goto et al. (2003) provides the beginning of work in this direction. Research into developing a standardized way of studying and measuring musical similarity (Ellis et al. 2002; Berenzweig et al. 2003) could also be adapted to genre-based applications.

In regard to expansions to the taxonomy, the sub-genres of techno are one area of particular muscicological interest, for which Reynolds' work (1998) could be of particular use, although it is becoming somewhat dated due to rapid rate of change in these types of music. The taxonomy could also be expanded to include the ways in which blues music has interfaced with other types of rhythm & blues music, such as jump blues and uptown

R&B. The addition of genres such as Rockabilly, Surf, early British Invasion and Folk Rock would help to make the taxonomy more complete. Folk musics of all kinds could also be added, including American Old-Time, as well as more World Pop. The limited availability of these recordings is a problem if one limits oneself to MIDI recordings, but the inclusion of an optical music recognition sub-system that could translate scores into MIDI would greatly improve matters.

The issues raised by Pachet and Cazaly (2000) and by Aucouturier and Pachet (2003), in terms of the needs of a large global database, could also be considered when designing an alternate taxonomy. A system could be implemented that classifies the genre of artists rather than individual recordings, for example. The emergent approach proposed by Aucouturier and Pachet (2003) of deriving taxonomies by applying data mining techniques to resources such as radio play lists and CD track listings could also provide a powerful alternative.

It could also be useful to train the system with custom recordings of musicians asked to play music in specific genres. This would have the advantage of using "idealized" prototypical training data rather than training data that may be "contaminated" with irregularities. This would, of course, carry the risk of making the system unable to classify recordings that are not purely typical of a genre. Perhaps it would be a good compromise to use training data comprised of both custom recordings and normal recordings.

Aside from improvements that could benefit the practical performance of the system, there are also several important areas of musicological research that could be investigated. Unsupervised learning could be applied to the features that were extracted from recordings in order to arrive at similarity measurements. The resultant categories could be compared to the genres used by humans in order to get a feeling of the relative "objective" similarity of music that humans put in the same or different categories. The resulting similarity measurement system would also have numerous applications independent of genre. Some of the research discussed in Chapter 4 would be of use here, and work such as that of Dittenbach et al. (2000) would be of particular use, as it allows the formation of hierarchical taxonomies which could be compared to existing hierarchical taxonomies.

Psychological experiments could be performed in order to gain a more precise idea of the extent to which humans agree with each other when they classify music. Experiments could be done where subjects come up with their own categories as well as experiments where subjects fit recordings into supplied categories. This would help to give a quantitative measure of human consistency that would put automatic classification success rates in better context. Subjects could also be asked why they classify certain recordings in certain ways, and the results could be compared to the particular features

that the automatic system used for different groups of genres. It would also be interesting to see if there is a difference in how well human subjects classify MIDI recordings compared to audio recordings.

The features extracted from different recordings and the results from the feature selection process for different groups of categories could be studied from a musicological perspective in order to attempt to derive meaning from the classifications that were performed and to help understand how different genres of music are related to particular features. Music theorists could also use the feature extraction system developed here to collect data that could be used to develop or enhance music theories in relation to different genres of music.

9. Bibliography

Aarden, B., and D. Huron. 2001. Mapping European folksong: Geographical localization of musical features. *Computing in Musicology* 12: 169–83.

Abdi, H., D. Valentin, and B. Edelman. 1999. *Neural Networks*. Thousand Oaks, CA: Sage Publications.

Adams, C. 1976. Melodic contour typology. *Ethnomusicology* 20 (2): 179–215.

Adeli, H., and S. L. Hung. 1995. *Machine learning: Neural networks, genetic algorithms and fuzzy systems.* New York: Jowh Wiley & Sons.

Agrawal, R., and R. Srikant. 1996. Mining sequential patterns: Generalizations and Performance improvements. *Proceedings of the International Conference on Extending Database Technology.* 3–17.

Ash, T. 1989. Dynamic node creation in backpropagation networks. *Connection Science* 1: 365–75.

Aucouturier, J. J., and F. Pachet. 2003. Representing musical genre: A state of the art. *Journal of New Music Research* 32 (1): 1–12.

Bengio, Y. 1996. *Neural networks for speech and sequence recognition*. London: International Thomson Computer Press.

Berenzweig, A., B. Logan, D. P. W. Ellis, and B. Whitman. 2003. A large-scale evaluation of acoustic and subjective music similarity measures. *Proceedings of the International Symposium on Music Information Retrieval.* 99–105.

Brackett, D. 1995. *Interpreting popular music.* New York: Cambridge University Press.

Brackett, D. 2002. *(In search of) musical meaning: Genres, categories and crossover.* London: Arnold.

Briscoe, G., T. Caelli. 1996. *A compendium of machine learning volume 1: Symbolic machine learning*. Norwood, NJ: Ablex Publishing.

Broere, B. J. 1983. El Chambú – A study of popular music in Nariño, South Colombia. In *World music, politics and social change*, edited by S. Frith. New York: Manchester University Press.

Brown, J. C. 1993 Determination of meter of musical scores by autocorrelation. *Journal of the Acoustical Society of America* 94 (4): 1953–7.

Burred, J. J., and A. Lerch. 2003. A hierarchal approach to automatic musical genre classification. *Proceedings of the International Conference on Digital Audio Effects*.

Chai, W. and B. Vercoe. 2001. Folk music classification using hidden Markov models. *Proceedings of the International Conference on Artificial Intelligence*.

Cook, N. 1987. *A guide to musical analysis.* London: J. M. Dent & Sons.

Cooper, G., and L. B. Meyer. 1960. *The rhythmic structure of music.* Chicago: University of Chicago Press.

Cope, D. 1991a.Computer simulations of musical style. *Computers in Music Research.* 15–7.

Cope, D. 1991b. *Computers and musical style*. Madison, WI: A-R Editions.

Cope, D. 1996. *Experiments in musical intelligence*. Madison, WI: A-R Editions.

Crump, M. 2002. *A principle components approach to the perception of musical style.* Honours thesis. University of Lethbridge, Canada.

Cumming, J. E. 1999. *The motet in the age of Du Fay.* Cambridge, UK: Cambridge University Press.

Dannenberg, R. B., B. Thom, and D. Watson. 1997. A machine learning approach to musical style recognition. *Proceedings of the International Computer Music Conference.* 344–7.

Dannenberg, R. B., and N. Hu. 2002. Pattern discovery techniques for music audio. *Proceedings of the International Symposium on Music Information Retrieval*. 63–70.

Deshpande, H., U. Nam, and R. Singh. 2001. Classification of music signals in the visual domain. *Proceedings of the Digital Audio Effects Workshop*.

Dittenbach, M., D. Merkl, and A. Rauber. 2000. The growing hierarchical self-organizing map. *Proceedings of the International Joint Conference on Neural Networks.* VI-15 – VI-19.

Duda, R.O., P.E. Hart, and D.G. Stork. 2001. *Pattern classification*. New York: John Wiley & Sons Inc.

Duff, D., ed. 2000. *Modern genre theory.* New York: Longman.

Ellis, D. P., B. Whitman, A. Berenzweig, and S. Lawrence. 2002. The quest for ground truth in musical artist similarity. *Proceedings of the International Symposium on Music Information Retrieval.*

Fabbri, F. 1981. A theory of musical genres: Two applications. In *Popular music perspectives*. edited by D. Horn and P. Tagg. Göteborg: IASPM.

Fabbri, F. 1982. What kind of music? *Popular Music* 2: 131–43.

Fabbri, F. 1999. Browsing music spaces: Categories and the musical mind. *Proceedings of the IASPM Conference*.

Fausett, L. 1994. *Fundamentals of neural networks: Architectures, algorithms and applications*. Englewood Cliffs, NJ: Prentice Hall.

Foote, J.T. 1999. Methods for the automatic analysis of music and audio. *FXPAL Technical Report FXPAL-TR-99-038*. FXPAL. Palo Alto, CA.

Frith, S. 1996. *Performing rites: On the value of popular music.* Cambridge, MA: Harvard University Press.

Frühwirth, M., and A. Rauber. 2001. Self-organizing maps for content-based music clustering. *Proceedings of the Italian Workshop on Neural Nets*.

Fujinaga, I. 1996. Exemplar-based learning in adaptive optical music recognition system. *Proceedings of the International Computer Music Conference*. 55–6.

Fukunaga, K. 1972. *Introduction to statistical pattern recognition.* New York: Academic Press.

Gabura, A. J. 1965. Computer analysis of musical style. *Proceedings of the ACM National Conference*. 303–14.

Gallant, S. I. 1993. *Neural network learning and expert systems.* Cambridge, MA: MIT Press.

Gardner, H. 1973. Children's sensitivity to musical styles. *Merrill-Palmer Quarterly* 19: 67–77.

Geary, D. M. 1999. *Graphic Java 2: Mastering the JFC.* Palo Alto, CA: Prentice Hall.

Gosling, J., and K. Arnold. 1996. *The Java programming language.* Don Mills, Canada: Addison-Weley.

Goto, M., H. Hashiguchi, T. Nishimura, and R. Oka. 2003. RWC music database: Music genre database and musical instrument sound database. *Proceedings of the International Symposium on Music Information Retrieval.* 229–30.

Granitto, P. M., P. F. Verdes, H. D. Navone, and H. A. Ceccatto. 2002. Aggregation algorithms for neural network ensemble construction. *Proceedings of the Brazilian Symposium on Neural Networks.* 178–83

Grant, B. K., ed. 2003. *Film genre reader III*. Austin, TX: University of Texas Press.

Grimaldi, M., A. Kokaram, and P. Cunningham. 2003. Classifying music by genre using a discrete wavelet transform and a round-robin ensemble. *Work Report.* Trinity College, University of Dublin, Ireland.

Hallinan, J. 2001. Feature selection and classification in the diagnosis of cervical cancer. In *The practical handbook of genetic algorithms applications,* edited by L. Chambers. New York: Chapman & Hall.

Hansen, L. K., and P. Salamon. 1990. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Learning* 12 (10): 993–1001.

Hargreaves, D. J., and A. C. North. 1999. Developing concepts of musical style. *Musicae Scientiae* 3: 193–216.

Hartono, P., and S. Hashimoto. 2002. Adaptive neural network ensemble that learns from imperfect supervisor. *Proceedings of the International Conference on Neural Information Processing.* 2561–5.

Horstmann, C. S., and G. Cornell. 2000. *Core Java Volume II – Advanced Features.* 2nd edition. Palo Alto, CA: Prentice Hall.

Horstmann, C. S., and G. Cornell. 2001. *Core Java Volume I – Fundamentals.* 2nd edition. Palo Alto, CA: Prentice Hall.

Hsu, J. L., C. C. Liu, and A. L. P. Chen. 2001. Discovering nontrivial repeating patterns in music data. *IEEE Transactions on Multimedia* 3 (3): 311–25.

Hussein, F., R. Ward, and N. Kharma. 2001. Genetic algorithms for feature selection and weighting, a review and study. *International Conference on Document Analysis and Recognition*. 1240-4.

Jagannathan, V., R. Dodhiawala, L.S. Baum, eds. 1989. *Blackboard architectures and applications.* New York: Academic Press.

Jain, A. K., and D. Zongker. 1997. Feature selection: Evaluation, application and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (2): 153–8.

Jain, A. K., R. P. W. Duin, and J. Mao. 1999. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (1): 4–37.

James, M. 1985. *Classification algorithms.* London: William Collins and Sons.

Jennings, H. D., P. C. Ivanov, A. M. Martins, P. C. da Silva, and G. M. Viswanathan. 2003. Variance fluctuations in nonstationary time series: A comparative study of music genres. Elsevier Science preprint: retrieved May 27, 2004 from http://xxx.lanl.gov/abs/cond-mat/0312380.

Jiang, D.N., L. Lu, H.J. Zhang, J.H. Tao, and L. H. Cai. 2002. Music type classification by spectral contrast feature. *Proceedings of Intelligent Computation in Manufacturing Engineering*. 113–6.

John, G. H., R. Kohavi, and K. Pfleger. 1994. Irrelevant features and the subset selection problem. *Proceedings of the International Conference on Machine Learning.* 121–9.

Kandel, A., and H. Bunke, eds. 2002. *Hybrid methods in pattern recognition.* London: World Scientific.

Katayose, H., and S. Inokuchi. 1989. The KANSEI music system. *Computer Music Journal* 13 (4): 72–7.

Karpov, I. 2002. Hidden Markov classification for musical genres. *Course Project*, Rice University.

Kirby, M. 2001. *Geometric data analysis: An empirical approach to dimensionality reduction and the study of patterns*. New York: John Wiley & Sons.

Kosina, K. 2002. Music genre recognition. *Diploma thesis*. Technical College of Hagenberg, Austria.

Laine, P., and M Kuuskankare. 1994. Genetic algorithms in musical style oriented generation. *Proceedings of the IEEE Conference on Evolutionary computation.* 858–62.

Lakoff, G. 1987. *Women, fire, and dangerous things: What categories reveal about the mind*. Chicago: University of Chicago Press.

Lambrou, T., P. Kudumakis, R. Speller, M. Sandler, and A. Linney. 1998. Classification of audio signals using statistical features on time and wavelet transform domains. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*: 3621–4.

Lartillot, O., S. Dubnov, G. Assayag, and G. Bejerano. 2001. Automatic modeling of musical style. *Proceedings of the International Computer Music Conference.* 447–54.

Lartillot, O. 2003. Discovering musical patterns through perceptive heuristics. *Proceedings of the International Symposiumon Music Information Retrieval.* 89–96.

LaRue, J. 1992. *Guidelines for style analysis*. Warren, MI: Harmonie Park Press.

Li, T., and G. Tzanetakis. 2003. Factors in automatic musical genre classification of audio signals. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.* 143–6.

Li, T., and M. Ogihara. 2003. Detecting emotion in music. *Proceedings of the International Symposium on Music Information Retrieval.* 239–40.

Li, T., M. Ogihara, and Q. Li. 2003. A comparative study on content-based music genre classification. *Proceedings of the ACM SIGIR Conference.* 282–9.

Liu, D., L. Lu, and H. J. Zhang. 2003. Automatic mood detection from acoustic music data. *Proceedings of the International Symposium on Music Information Retrieval.* 81–7.

Lomax, A. 1968. *Folk song style and culture.* Washington: American Association for the Advancement of Science.

Maclin, R., and J. Shavlik. 1995. Combining the predictions of multiple classifiers: Using competitive learning to initialise neural networks. *Proceedings of the International Joint Conference on Artificial Intelligence.* 524–30.

Manuel, P. 1988. *Popular musics of the non-western world*. Toronto: Oxford University Press.

Matityaho, B., and M. Furst. 1995. Neural network based model for classification of music type. *Proceedings of the Convention of Electrical and Electronic Engineers*. 4.3.4/1–5.

McKay, C. 2004. Automatic genre classification as a study of the viability of high-level features for music classification. Accepted for publication in the *2004 Proceedings of the International Computer Music Conference.*

McKay, C., and I. Fujinaga. 2004. Automatic genre classification using large high-level musical feature sets. Accepted for publication in the *2004 Proceedings of the International Symposium on Music Information Retrieval.*

McKinney, M. F., and J. Breebaart. 2003. Features for audio and music classification. *Proceedings of the International Symposium on Music Information Retrieval.* 151–8.

Michalski, R. S., I. Bratko, M. Kubat, eds. 1999. *Machine learning and data mining: Methods and applications.* Toronto: John Wiley & Sons.

Middleton, R. 1990. *Studying popular music*. Philadelphia: Open University Press.

Middleton, R. 2000. Popular music analysis and musicology: Bridging the gap. In *Reading pop: Approaches to textual analysis in popular music,* edited by R. Middleton. New York: Oxford University Press.

MIDI Manufacturers Association. 2001. *Complete MIDI 1.0 detailed specification v96.1*. Los Angeles: International MIDI Association.

Mitchell, T. M. 1997. *Machine learning*. New York: McGraw-Hill.

Moore, A. F. 2001. Categorical conventions in music discourse: Style and genre. *Music & Letters* 82 (3): 432–42.

Nam, N., and J. Berger. 2001. Addressing the same but different-different but similar problem in automatic music classification. *Proceedings of the International Symposium in Music Information Retrieval*. 21–2.

Negus, K. 1999. *Music genres and corporate cultures.* New York: Routledge.

Nettl, B. 1990. *Folk and traditional music of the western continents*. Englewood Cliffs, NJ: Prentice-Hall.

North, A. C., and D. J. Hargreaves. 1997. Liking for musical styles. *Music Scientae* 1: 109–28.

Opitz, D., and R. Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research* 11: 169–98.

Pachet, F., and D. Cazaly. 2000. A taxonomy of musical genres. *Proceedings of the Content-Based Multimedia Information Access Conference*.

Pachet, F. 2002. The Continuator: Musical interaction with style. *Proceedings of the International Computer Music Conference.* 211–8.

Pampalk, E. 2001. Islands of music: Analysis, organization and visualization of Music Archives. *Master's thesis*. Vienna University of Technology, Austria.

Pampalk, E., A. Rauber, and D. Merkl. 2002. Content-based organization and visualization of music archives. *Proceedings of ACM Multimedia*. 570–9.

Pampalk, E., S. Dixon, and G. Widmer. 2003. Exploring music collections by browsing different views. *Proceedings of the International Symposium on Music Information Retrieval.* 201–8.

Pantham, S. 1999. *Pure JFC Swing.* Indianopolis, IN: SAMS.

Perrott, D., and R. O. Gjerdingen. 1999. Scanning the dial: An exploration of factors in the identification of musical style. *Research Notes*. Department of Music, Northwestern University, Illinois, USA.

Ponce de Leon, P. J., and J. M. Inesta. 2002. Musical style identification using self-organising maps. *Proceedings of the International Conference on Web Delivery of Music.* 82–89.

Pudil, P., F. Ferri, J. Novovicova, and J. Kittler. 1994. Floating search methods for feature selection with nonmonotonic criterion functions. *Proceedings of the IEEE International Conference on Pattern Recognition*. 279–83.

Pye, D. 2000. Content-based methods for the management of digital music. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. 2437–40.

Rabiner, L., and B. H. Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine* 3 (1): 4–16.

Rabiner, L. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77 (2): 257–86.

Raphael, C. and J. Stoddard. 2003. Harmonic analysis with probabilistic graphical models. *Proceedings of the International Symposium on Music Information Retrieval.* 177–81.

Rauber, A., and M. Frühwirth. 2001. Automatically analysing and organising music archives. *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries*. 402–14.

Rauber, A., and A. Müller-Kögler. 2001. Integrating automatic genre analysis into digital libraries. *Proceedings of the ACM-IEEE Joint Conference on Digital Libraries.* 1–10.

Rauber, A., E. Pampalk and D. Merkl. 2002. Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarity. *Proceedings of the International Symposium on Music Information Retrieval*. 71–80.

Reti, R. 1951. *The thematic process in music.* New York: Macmillan.

Reynolds, S. 1998. Generation ecstasy: Into the world of techno and rave culture. Boston: Brown.

Rosch, E. 1975. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General* 104: 192–233.

Rothstein, J. 1995. *MIDI: A comprehensive introduction.* Madison, WI: A-R Editions.

Rowe, R. 2001. *Machine musicianship.* Cambridge, MA: MIT Press.

Russell, S., and P. Norvig. 2002. *Artificial intelligence: A modern approach*. Upper Saddle River, NJ: Prentice Hall.

Ryan, M. L. 1981. Introduction: On the why, what, and how of generic taxonomy. *Poetics* 10: 109–26.

Sakawa, M. 2002. *Genetic algorithms and fuzzy multiobjective optimization.* Norwell, MA: Kluwer Academic Publishers.

Scheirer, E.D. 2000. Music-listening systems. *Doctoral thesis*. M.I.T., MA.

Selfridge-Field, E., ed. 1997. *Beyond MIDI: The handbook of musical codes.* Cambridge, MA: MIT Press.

Selfridge-Field, E. 1998. Conceptual and representational issues in melodic comparison. In *Melodic similarity: Concepts, procedures, and applications*, edited by W. B. Hewlett and E Selfride-Field. Cambridge, MA: MIT Press.

Shan, M. K., and F. F. Kuo. 2003. Music style mining and classification by melody. *IEICE Transactions on Information and Systems* E86-D (3): 655–9.

Siedlecki, W., and J. Sklansky. 1989. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters* 10 (5): 335–47.

Society of Motion Picture and Television Engineers. 1994. *ANSI/SMPTE 268M-1994, SMPTE standard for file format for digital moving-picture exchange (DPX), v 1.0, 18*. White Plains, NY: Snell & Wilcox.

Soltau, H., T. Schultz, M. Westphal, and A. Waibel. 1998. Recognition of musical types. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing.* 1137–40.

Stevens, C., and C. Latimer. 1997. Music recognition: An illustrative application of a connectionist model. *Psychology of Music* 25 (2): 161–85.

Tagg, P. 1982. Analysing popular music: Theory, method and practice. *Popular Music* 2: 37–67.

Tarasti, E. 2002. *Signs of music: A guide to musical semiotics.* New York: Mouton de Gruyter.

Tax, D., M. van Breukelen, R. Duin, and J. Kittler. 2000. Combining multiple classifiers by averaging or by multiplying? *Pattern Recoginition* 33 (9): 1475–85.

Taylor, J. R. 1989. *Linguistic categorization: Prototypes in linguistic theory*. Oxford: Clarendon Press.

Tekman, H. G., and N. Hortacsu. 2002. Aspects of stylistic knowledge: What are different styles like and why do we listen to them? *Psychology of Music* 30 (1): 28–47.

Temperley, D. 2001. *The cognition of basic musical structures.* Cambridge, MA: MIT Press.

Toynbee, J. 2000. *Making popular music: Musicians, creativity and institutions.* London: Arnold.

Towsey, M., A. Brown, S. Wright, and J. Diederich. 2001. Towards melodic extension using genetic algorithms. *Educational Technology & Society* 4 (2): 54–65.

Tseng, Y.H. 1999. Content-based retrieval for music collections. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 176–82.

Typke, R., P. Giannopoulis, R. C. Veltkamp, F. Wiering, and R van Oostrum. 2003. Using transportation distances for measuring melodic similarity. *Proceedings of the International Symposium on Music Information Retrieval.* 107–14.

Tzanetakis, G. 2002. Analysis and retrieval of audio signals. *Doctoral dissertation*. Princeton University, USA.

Tzanetakis, G., and P. Cook. 2002. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing* 10 (5): 293–302.

Tzanetakis, G., A. Ermolinskyi, and P. Cook. 2002. Pitch histograms in audio and symbolic music information retrieval. *Proceedings of the International Symposium on Music Information Retrieval.*

Tzanetakis, G., G. Essl, and P. Cook. 2001. Automatic musical genre classification of audio signals. *Proceedings of the International Symposium on Music Information Retrieval*. 205–10.

Uitdenbogerd, A. and J. Zobel. 1998. Manipulation of music for melody matching. *Proceedings of the ACM International Multimedia Conference.* 235–40.

Vafaie, H., and I. Imam. 1994. Feature Selection Methods: Genetic Algorithms vs. Greedy-like Search. *Proceedings of the International Conference on Fuzzy and Intelligent Control Systems.*

de Villiers, J., and E. Barnard. 1992. Backpropagation neural networks with one and two hidden layers. *IEEE Transactions on Neural Networks* 4 (1): 136–41.

Wanas, N. M., G. Auda, M. Kamel and F. Karray. 1998. On the optimal number of hidden nodes in a neural network. *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering.* 918–21.

Wanas, N. M., and M. S. Kamel. 2001. Decision fusion in neural network ensembles. *Proceedings of the International Joint Conference on Neural Networks.* 2952–7.

Wanas, N. M., and M. S. Kamel. 2002. Weighted combination of neural network ensembles. *Proceedings of the International Joint Conference on Neural Networks*. 1748–52.

Whitehead, P., E. Friedman-Hill, and E. Vander Veer. 2002. *Java and XML.* Mississauga, Canada: Wiley Publishing Inc.

Whitman, B., and P. Smaragdis. 2002. Combining musical and cultural features for intelligent style detection. *Proceedings of the International Symposium on Music Information Retrieval*. 47–52.

Xu, C., N. C. Maddage, X. Shao, F. Cao, and Q. Tian. 2003. Musical genre classification using support vector machines. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. Vol. V, pp. 429–32.

Yip, C. L., and B. Kao. 1999. A study on musical features for melody databases. *HKU CSIS Technical Report TR-99-05.* Hong Kong University, Department of Computer Science and Information Systems, Hong Kong.

Zenobi, G., and P. Cunningham. 2001. Using diversity in preparing ensembles of classifiers based on different subsets to minimize generalization error. *Proceedings of the European Conference on Machine Learning*. 576–87.

Zhou, Z. H., J. X. Wu, Y. Jiang, and S. F. Chen. 2001. Genetic algorithm based selective neural network ensemble. *Proceedings of the International Joint Conference of Artificial Intelligence,* 797–802.

All classical guide. Retrieved November 6, 2003, from http://www.allclassical.com.

All music guide. Retrieved November 6, 2003, from http://www.allmusic.com.

Amazon.com. Retrieved November 6, 2003, from http://www.amazon.com.

Java technology. Retrieved May 5, 2003, from http://www.java.sun.com.

MIDI Manufacturers Association. Retrieved Sept. 4, 2003, from http://www.midi.org.

Xerces Java Parser. Retrieved January 7, 2004, from http://xml.apache.org/xerces-j.