Novel techniques for Audio Music Classification and Search

Kris West

School of Computing Sciences

University of East Anglia

A thesis submitted for the degree of

Doctor of Philosophy

September 2008

I would like to dedicate this thesis to my late father, Daniel West, who taught me how to work, build and to solve my problems; to my mother, Shelagh West, who has always supported me and who somehow managed to get me accepted into a good school when I was an illiterate eight year old; and finally to Eloise, Keziah and one yet to come, with whom I will spend the rest of my life and remind me every day what it is to live and be alive.

Acknowledgements

I would like to acknowledge the help and support of my Supervisor, Prof. Stephen Cox, who has helped me to define and direct my research and has been able answer nearly every one of my endless questions. I would like to thank Paul Lamere and the folks at Sun Microsystems for my internship. I would also like to thank Prof. J Stephen Downie and his group (Andreas Ehmann, Mert Bay, David Tcheng, Cameron Jones, Jin Ha Lee, Xiao Hu and the rest) at the International Music Information Retrieval System Evaluation Laboratory (IMIRSEL) for their support in the implementation of my experiments in Music-2-Knowledge (M2K), the evaluation of the performance of my techniques on an independent test collection, for the organisation of the annual Music Information Retrieval Evaluation eXchange (MIREX) and for putting up with my constant interference.

Many of the experiments in this thesis were implemented in the Data-2-Knowledge (D2K), produced by the Automated Learning Group (ALG) at the NCSA). Much of the software developed for these experiments has been contributed to the M2K toolkit. Several of the numerical procedures were implemented using Java Matrix Toolkit (JAMA) provided by the American National Institute of Standards and Technology (NIST) and several of the classification algorithms evaluated were provided as part of the Weka Machine Learning Toolkit from the University of Waikato. D2K, M2K, Jama and Weka are all implemented in Java and have been executed in Java virtual Machines (JVMs) provided by Sun Microsystems. Finally, statistical significance testing with multiple comparisons was performed using Matlab and the Statistical Toolkit.

Abstract

This thesis presents a number of modified or novel techniques for the analysis of music audio for the purposes of classifying it according genre or implementing so called 'search-by-example' systems, which recommend music to users and generate playlists and personalised radio stations. Novel procedures for the parameterisation of music audio are introduced, including an audio event-based segmentation of the audio feature streams and methods of encoding rhythmic information in the audio signal. A large number of experiments are performed to estimate the performance of different classification algorithms when applied to the classification of various sets of music audio features. The experiments show differing trends regarding the best performing type of classification procedure to use for different feature sets and segmentations of feature streams.

A novel machine learning algorithm (MVCART), based on the classic Decision Tree algorithm (CART), is introduced to more effectively deal with multi-variate audio features and the additional challenges introduced by event-based segmentation of audio feature streams. This algorithm achieves the best results on the classification of event-based music audio features and approaches the performance of state-of-the-art techniques based on summaries of the whole audio stream.

Finally, a number of methods of extending music classifiers, including those based on event-based segmentations and the MVCART algorithm, to build music similarity estimation and search procedures are introduced. Conventional methods of audio music search are based solely on music audio profiles, whereas the methods introduced allow audio music search and recommendation indices to utilise cultural

information (in the form of music genres) to enhance or scale their recommendations, without requiring this information to be present for every track. These methods are shown to yield very significant reductions in computational complexity over existing techniques (such as those based on the KL-Divergence) whilst providing a comparable or greater level of performance. Not only does the significantly reduced complexity of these techniques allow them to be applied to much larger collections than the KL-Divergence, but they also produce metric similarity spaces, allowing the use of standard techniques for the scaling of metric search spaces.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Music discovery and categorization problem

The recent growth of digital music distribution and the rapid expansion of both personal music collections and the capacity of the devices on which they are stored has increased both the need for and the utility of effective applications for organising, browsing and searching music collections and generating playlists from them.

In order to implement such applications we must first be able to estimate categorisations of tracks (for example, according to genre or mood), estimate the level of similarity between tracks and build efficient indexes based on those estimates. Systems for music similarity estimation and recommendation have been demonstrated based on social data using a technique known as Collaborative Filtering (33), hand-crafted metrics based on detailed metadata created by music experts (73), metrics based on features extracted from music scores (64) (25) and audio content-based metrics. The utility of music audio content-based metrics for estimating similarity between songs is well-known in the Music Information Retrieval (MIR) community (3) (47) (56) as they substitute relatively cheap computational resources for expensive human editors, and allow users to access the 'long tail' (1) (music that might not have been reviewed or widely distributed, making reviews, scores or usage data difficult to collect) by directly analysing the

audio of any track they are presented with.

We may also need to estimate the similarity within tracks, in order to select a preview clip or thumbnail of a music track. If this clip is representative of the whole track it can be used to aid users in browsing music collections and evaluating search results.

In the following sections we describe three applications in this area. These applications form the focus of the work in this thesis.

1.1.1 Music thumbnailing / Preview clip generation

In many music search or browsing systems, a user's first contact with the music for the tracks will be through a preview clip or thumbnail that contains an example of the music contained in the track. Hence it is important that the preview clip accurately reflects the content of the rest of the track. Unfortunately, the introductions to many tracks do not well represent the content of the main body of the track and due to variation of the content within a track, a randomly chosen clip may fall in an unfortunate position and hence poorly represent the track.

The selection of a good preview clip can be relatively easily performed by a human listener. However, the listener would be required to review the entire track, then to select a clip and finally perform some validation (checking the clip they selected is a good preview) and adjustment. For an average music track containing 3 - 5 minutes of audio, this process becomes prohibitively time consuming for a large music collection (>200 days for 50,000 tracks, >20 years for 2,000,000 tracks). Hence, previews are often selected randomly from the track. Computational methods for selecting a representative preview of a track might be able to significantly outperform random selection and complete the task in a practical time-scale.

1.1.2 Genre and mood classification

Human beings often use contextual or cultural labels when describing music. A single description might contain references to one or more genres or styles of music, a particular period in time, similar artists or the emotional content of the

music, and are rarely limited to a single descriptive label. For example the music of Damien Marley has been described as "a mix of original Dancehall reggae with an R&B/Hip Hop vibe"[1].

A traditional method of indexing music for search by a user (for example organising CDs in a record shop or tracks in an online digital music store) is to divide the music into *genres* such as Rock, Heavy Metal and Reggae. Often these purely metadata-based methods of similarity judgement and classification have to make use of metadata applied by human annotators. However, these labels introduce their own problems. Detailed music description by an annotator takes a significant amount of time, labels can only be applied to known examples (so novel music cannot be analysed until it has been annotated), and it can be difficult to achieve a consensus on music description, even amongst expert listeners.

Some applications overcome the expense of annotating data by providing interfaces to collect *social tagging data*, allowing users of the music collection to apply their own tags to the music. The most common tags applied across all users to each track can be used to search or organise the collection. However, these techniques rely on at least one user having reviewed a track and applied tags to it. Hence, new releases or relatively unknown music in the 'long tail' cannot be accessed or searched. This is known as a 'cold start problem' (61).

Content-based audio genre or mood classification systems attempt to overcome this by using features derived from the audio content to classify a track into a mood or genre, without having to collect any user data for the track or having a human listener review it. Features are computed from each track's audio and analysed by a model trained on a reference database of music features for each genre or mood that it should recognise.

Genre classification has often been used to test the performance of feature sets for the construction of music similarity estimation metrics. Given that examples of music of a particular genre are likely to be relatively similar to each other, it is assumed that features which perform well for genre classification will also perform well for similarity estimation — particularly as a genre classification model can

[1] http://cd.ciao.co.uk/

be simply implemented by creating a 'Nearest Neighbour' model based on your similarity metric. The clustering of tracks according to genre labels has also been used to pseudo-objectively evaluate the performance of music similarity metrics, as the genre classification metadata for N tracks is less problematic to obtain and use than human estimates of the $N * (N - 1)$ similarities between those N tracks. The term 'Pseudo-Objective' is used to acknowledge the fact that although the statistics maybe objective some of the metadata (e.g. genre) over which they are computed is somewhat subjective in that different listeners may have different opinions or levels of granularity in the labels they apply to tracks.

1.1.3 Music similarity estimation

In order to implement *search-by-example* queries for music databases ("find me music similar to X"), recommendation queries ("find me music similar to music I already own") and playlist generation queries ("find me music similar to these N tracks") we need to be able to estimate the similarity of one track to another. Hence, music similarity estimation algorithms try to produce a single dimensional estimate of the similarity a human would perceive between two tracks, which can be termed a *macro* similarity function. In reality, the perception of music similarity is likely to be multi-dimensional (*micro* similarities between tracks in terms of timbre, rhythm, melody, harmony, lyrics, culture). However, a single dimensional metric is both useful for search applications and duplicates a human listener's ability to decide that a track is similar or not similar to a query and to divide music into genres and moods based on some form of macro similarity.

Estimation of a single dimensional, music similarity metric has been implemented with social data by using collaborative filtering (users that played X also played Y), social tag data (comparing tags applied to tracks by users), expert metadata (applying a hand-crafted metric to metadata applied to a track by an expert using a predefined schema), through analysis of human transcribed scores and through direct analysis of the audio content.

Metrics based directly on audio content are desirable because:

- they do not require expensive human labour to create metadata for each track,
- they do not require a number of human listeners to have played or tagged a track before it can be recommended (i.e. they do not suffer from 'cold start' problems),
- they do not require a pre-transcribed score,
- and, by definition, they are the only techniques that directly analyse the documents being searched.

Music is a complex form of communication in which both artists and cultures express their ideas and identity. When we listen to music we do not simply perceive the acoustics of the sound in a temporal pattern, but also its relationship to other sounds, songs, artists, cultures and emotions. Owing to the complex, culturally-defined distribution of acoustic and temporal patterns amongst these relationships, it is unlikely that a simple low-level *audio* similarity metric will be suitable as a *music* similarity metric.

However, music classification systems demonstrate that complex relationships between audio features and higher-level characteristics of the music, such as genre, mood or key, *can* be learnt using sufficiently powerful classification techniques. We also contend that content-based music similarity estimators are not easily defined as expert systems because relationships between musical concepts that form our musical cultures are defined in a complex, ad-hoc manner, with no apparent intrinsic organising principle. Therefore, effective content-based music similarity estimators may need to reference some form of historical or cultural context in order to effectively emulate human estimates of similarity. Automatic estimators will also be constrained by the information on which they were trained and will likely develop a 'subjective' view of music, in a similar way to a human listener.

Because the processes that are used to implement music audio content analysis and comparison algorithms are computationally intensive and often scale linearly, a key challenge for music similarity estimation techniques is the building of efficient scalable indexes. Such techniques would facilitate the indexing and search

of "industrial sized" collections (500,000 to 3,000,000 tracks) with content-based techniques.

1.2 Overview of Thesis

This thesis introduces new approaches to standard feature extraction and summarisation processes for audio music classification and search that yield improvements in the performance and memory footprint of those tasks and also enable efficient scalable indexing of music collections. Specifically, an alternative approach to the extraction and summarisation of common timbral features is examined, novel features describing the rhythm of a track are introduced, classification strategies are examined and three intuitive extensions of machine-learning based music classification models to audio similarity estimation are introduced and evaluated. These music similarity estimation techniques allow the creation of music search indices which achieve a very high level of performance and efficiently scale to industrial music search problems, where standard comparison procedures do not.

In chapter 2, existing work in audio content-based music similarity estimation, classification and preview clip generation is surveyed. Some of the challenges in creating efficient scalable music audio analysis and search systems are also explored. In chapter 3 new techniques for the extraction of music audio features representing both timbral and rhythmic characteristics of the music are introduced, including an alternative to present frame-based feature processing approaches in the form of event-based parameterisations of the music stream.

In chapter 4, a selection of classification algorithms are applied to the various feature sets computed in chapter 3. Methods are reviewed for the summarisation of feature vector streams into a single vector for use in classification and alternative methods for extending traditional single feature vector classification algorithms to the handling of vector steams are introduced. Finally, A novel machine learning algorithm (MVCART), based on the classic Decision Tree algorithm (CART), is introduced to more effectively deal with multi-variate audio features and the additional challenges introduced by event-based segmentation

of audio feature streams. This algorithm achieves the best results on the classification of event-based music audio features and approaches the performance of state-of-the-art techniques based on summaries of the whole audio stream.

In chapter 5 existing approaches to music similarity estimation, based on geometric or probabilistic distance measurements between features computed from the music audio, are reviewed. These techniques utilise only the audio profiles of tracks being compared in making their similarity estimates. This thesis introduces new approaches to the calculation of music similarity estimates that use machine-learning techniques to encode cultural information about music which can then be used to enhance or scale music similarity estimates. The indexes produced are both accurate and scalable overcoming many of the drawbacks of existing techniques and allowing the extension of audio similarity estimation to much larger databases.

Both the classification and musical similarity experiments reported here have been thoroughly evaluated both statistically, and where possible, through tests involving human subjects (in the case of musical similarity measures). The evaluation results are subjected to a detailed analysis for statistical significance and conclusions are drawn about the performance, both in terms of accuracy and computational complexity, of the techniques developed in this thesis. The evaluation procedures used where developed and themselves evaluated as part of the Music Information Retrieval Evaluation eXchange (MIREX) (38) and therefore provide a suitable example for other researchers wishing to explore the performance of their techniques.

Chapter 2

Existing work in music search, classification and thumbnailing

In this chapter, different approaches to classification and search are compared, and existing works on content-based similarity estimation, classification and preview clip generation are reviewed.

2.1 Collaborative Filtering based approaches

Collaborative filtering is an information retrieval approach often used where content-based retrieval methods are weak or undefined, such as when recommending movies, music or books (33). Collaborative filtering works by collecting user ratings of items and either matching users to other users with similar tastes and recommending items from the other user's collection (users that bought A also bought B) or matching an item to other items by comparing which users positively rated both items.

Collaborative filtering has been successfully applied to searching music documents by a number of online services including Last.FM[1], InDiscover[2], OneL-

[1] http://www.last.fm
[2] http://indiscover.net

lama[1], iRate Radio[2] and MyStrands[3].

The main drawbacks of collaborative filtering based approaches are 'cold start' problems, where a track cannot be recommended until sufficient playback events or user ratings have been collected (preventing the systems from recommending new or unknown music), odd recommendations caused by the sometimes diverse tastes of users or very popular tracks and the fact that they do not directly use the music content and so may have trouble finding similar 'sounding' music.

2.2 Human-edited metadata and score based approaches

Other approaches to the estimation of music similarity have been based on detailed human-applied metadata or transcribed scores in common music notation.

Similarity estimation based on detailed metadata has been successfully applied by Pandora[4] to create a personalised internet radio services. The similarity estimates used to create the station playlists are derived by comparing very detailed metadata applied to each track according to a schema established as part of the music genome project (73). An incomplete list of the attributes that form the schema can be found on wikipedia[5].

The major drawback of using human edited metadata to estimate track similarity is that each track has to be individually reviewed. Teams of human editors are expensive and impose limits on the number of tracks that can ingested by the recommendation system. At the time of writing Pandora claims to index 'around half a million tracks' and adds around 15,000 tracks to its collection per month. This means that they are long way from being able to index other music catalogues and tracks that form part of the 'long tail' (1) of less well-known music.

[1] http://www.onellama.com
[2] http://irate.sourceforge.net
[3] http://www.mystrands.com
[4] http://www.pandora.com/
[5] http://en.wikipedia.org/wiki/List_of_Music_Genome_Project_attributes

A further drawback of human edited metadata is that a set of musically relevant attributes must be defined with which to label the catalogue. In order to acknowledge all aspects of music perception, significant work must be put into this set of attributes. Further, once a catalogue has been labelled according to these attributes it requires significant effort to modify or extend the attribute set, as every track in the catalogue must be relabelled.

Systems have also been demonstrated that retrieve specific pieces of music through indexing of monophonic scores in common music notation (often encoded as midi files) (25) and that classify scores into music genres (52) or estimate the similarity of scores for query-by-example systems (64). The approaches have the added advantage that they can accept queries based on a transcribed fragment of a melody but have the disadvantages of requiring a score to be available (or even applicable to the style of music) and abstract performance and timbral information from the query.

2.3 Audio content-based approaches

A number of content-based methods of estimating the similarity of audio music recordings have been proposed. Many of these techniques consider only short-time spectral features, related to the timbre of the audio, and ignore most of the pitch, loudness and timing information in the songs considered. We refer to such techniques as 'timbral' music similarity functions.

Logan and Saloman (47) present an audio content-based method of estimating the timbral similarity of two pieces of music that has been successfully applied to playlist generation, artist identification and genre classification of music. This method is based on the comparison of a 'signature' for each track with the Earth Mover's Distance (EMD), a mathematical measure of the difference between two distributions. The signature is formed by the clustering, with the K-means algorithm, of Mel-frequency Cepstral Coefficients (MFCCs) calculated for short frames of the audio signal. The proceedure for calculating MFCCs is described in section 3.2.1.

Another content-based method of similarity estimation, also based on the calculation of MFCCs from the audio signal, is presented by Aucouturier and Patchet (3). A mixture of Gaussian distributions is trained on the MFCC vectors from each song and are compared by sampling the distributions (generating random points according to one distribution and estimating their likelihood based on the other distribution) in order to estimate the timbral similarity of two pieces. Aucouturier and Pachet report that their system identifies surprising associations between certain songs, often from very different genres of music, which they exploit in the calculation of what they term an 'Aha' factor. 'Aha' is calculated by comparing the content-based 'timbral' distance measure to a metric based on textual metadata. Pairs of tracks identified as having similar timbres, but whose metadata does not indicate that they might be similar, are assigned high values of the 'Aha' factor.

It is contended that these associations are due to confusion between superficially similar timbres, such as a plucked lute and a plucked guitar string or the confusion between a Folk, a Rock and a World music track, described in (3), which all contain, for example, acoustic guitar playing and gentle male voice. A deeper analysis might separate the timbres corresponding to different genres of music and prevent errors that may lead to poor performance on tasks such as playlist generation or song recommendation. Aucouturier and Pachet define a weighted combination of their similarity metric with a metric based on textual metadata, allowing the user to adjust the number of these confusions. Reliance on the presence of textual metadata effectively eliminates the benefits of a purely content-based similarity metric.

A similar method is applied to the estimation of similarity between tracks, artist identification and genre classification of music by Pampalk, Flexer and Widmer (56). Again, a spectral feature set based on the extraction of MFCCs is used and augmented with an estimation of the fluctuation patterns of the MFCC vectors over $\tilde{6}$ second windows. These fluctuation patterns are intended to encode information about periodic changes in the timbre (such as the presence of vibrato) and longer term timbral characteristics that cannot be represented by short-time feature frames. Classification is implemented by calculating either the

EMD or comparing mixtures of Gaussian distributions of the features, using the same approach as Aucouturier and Patchet (3), and assigning to the most common class label amongst the nearest neighbours.

Tzanetakis and Cook (65) present a system for automatically classifying audio tracks into music genres based on audio content. The features used for classification in this system can be divided into three broad categories; Timbral texture features, rhythmic content features and pitch content features.

The timbral features are all based on the short-time FFT of the signal and include spectral centroid (the centre of gravity of the magnitude spectrum of the FFT), the spectral rolloff (the frequency below which 85% of the spectral energy of the audio signal is concentrated), the spectral flux (the sum of the squared differences between the current value of the normalised magnitudes of each band of the spectral distribution and its value from the last analysis frame), the zero crossing rate (a crude measure of the level of high frequency components in the signal), the first five Mel-Frequency Cepstral Coefficients (MFCCs) and the Low-energy level (the percentage of analysis frames whose RMS energy is less than the average RMS).

Rhythmic features in this system are derived from the Discrete Wavelet Transform (DWT), which is an alternative to the Fourier transformation. The DWT is designed to combat the Fourier transformation's time and frequency resolution problems through non-linear frequency resolution. The audio signal is processed by the DWT algorithm in 50% overlapping, 3 second windows and then decomposed into octave frequency bands. Each band is full wave rectified in order to extract the temporal envelope of the signal rather than its time domain representation. The envelope is low pass filtered, down-sampled and mean removal is performed to centre the signal to zero before an enhanced autocorrelation is performed on it to extract periodicities in the data. The autocorrelation is enhanced to reduce the effects of integer multiples of the basic periodicities in the data.

The first three peaks of the enhanced autocorrelation function, in the range 40–200 beats-per-minute (BPM), are selected and their peak amplitude values are added to a beat histogram. The rhythmic information captured by the beat histogram is more detailed than in most systems which only aim to provide an

estimate of the main beat in the audio signal. The following features are calculated from the beat histogram for classification: the relative amplitude of the first and second histogram peak, the ratio of the amplitude of the second peak divided by the amplitude of the first peak, the period of the first and second peak in BPM and the overall sum of the histogram. For the purposes of these calculations the DWT is performed in the three second windows, which overlap by 50%.

The final set of features used for classification is related to the pitch content of the audio signal. Multi-pitch detection is performed in this system in a similar way to beat detection, but over a much shorter time-scale of 23ms rather than 3 seconds. The signal is divided into two bands, half-wave rectified and low-pass filtered to extract the envelope of the signals. These envelopes are summed and another enhanced autocorrelation performed. The peaks in the autocorrelation function correspond to the main pitches in the audio segment. These pitches are then converted to midi note numbers, which represent their pitch, with the following equation:

$$n = round\left(12\log_2\frac{f}{440} + 69\right) \tag{2.1}$$

where f is the frequency in Hertz and n is the histogram bin or midi note number. The midi notes are added to two pitch histograms, the first is raw representation of the midi notes and the second is a *folded* version of the histogram, which maps all the notes of the histogram to a single octave, using the expression:

$$c = n \bmod 12 \tag{2.2}$$

where c is the folded histogram bin and n is the unfolded midi note number. The normal histogram encodes information about the pitch range and the folded histogram gives information about pitch classes or harmonic content of the audio. Finally, the folded histogram is converted from a semitone scale to a circle of fifths, so that adjacent bins are spaced a fifth apart rather than a semitone. This is done so that the histogram bins are better suited for expressing tonic-dominant tonal music relations, and results in better classification accuracy. The following features are calculated for classification: the amplitude of maximum peak of the folded histogram, the period of the maximum peak of unfolded histogram, the

period of the maximum peak of the folded histogram, the pitch interval between the largest peaks of the folded histogram and the sum of the histogram.

The classifiers used to evaluate these feature sets include single Gaussian models (with Mahalanobis distance measurements), Gaussian mixture models (with diagonal covariance matrices, initialised with the K-means algorithm using multiple random starting points) and a K-nearest neighbour classifier. The system was evaluated using a hierarchy of 20 music genres and 3 speech genres, each with 100, 30s excerpts used for training. The system performs with approximately 61% accuracy on the test dataset.

Lidy, Rauber, Pertusa and Iñesta (45) present a system for genre classification that combines several feature sets describing the rhythm and timbre of a piece with features estimated from an automatically transcribed score. The use of automatic transcription mitigates against the problems normally associated with needing a score to analyse.

The rhythm features include Rhythm Patterns (60 FFT coefficients for each of 24 Bark scale filter outputs transformed onto the Sone scale), Rhythm Histograms (the sum of each of the FFT coefficients computed for the rhythm patterns across each of the 24 bands) and Onset features (an onset detection function is applied to the signal to discover which frames contain the onset of an audio event and the minimum, maximum, mean, median and standard deviation of the time in frames between each pair of consecutive onsets are returned). The timbral features, referred to as Statistical Spectrum Descriptors, extracted include the mean, median, variance, skewness, kurtosis, min- and max-values of the specific loudness sensation of each of 24 Bark-scale bands.

The symbolic features are computed by applying a polyphonic transcription algorithm (which is known to produce errors as perfect polyphonic transcription has yet to be solved) to the signal. Subsequently, the following features are computed: the number of notes, number of significant silences, the number of non-significant silences, total number of Inter Onset Intervals (IOIs), the number of distinct pitch intervals, the count of the most repeated pitch interval, the sum of all note durations and the min, max, mean, standard deviation and normality

of each of the distributions of note pitches, durations, IOIs and non-diatonic notes.

Lidy, Rauber, Pertusa and Iñesta evaluate the use of different subsets of the feature sets with a first order polynomial Support Vector Machine (SVM) classifier and obtain modest increases in performance for using the symbolic and rhythm features in addition to the Statistical Spectrum Descriptors.

2.4 Preview clip generation

In (34), Huron identifies the ability for music users to hear a short representative segment of music tracks (a preview clip) as one of the most important forms of feedback in music retrieval/recommendation systems and characterises them as"a musical equivalent of the 'thumbnails' commonly used in electronic picture galleries" (34). Huron also concludes that the use of 'incipits' (the first seconds of a track) and random clips provides a far from optimal preview for most modern music.

Huron introduces a useful conceptual approach to the automatic selection of thumbnails based on prototype theory, which suggests that some objects within a class of objects are better representatives of the class in general. Further, he proposes some interesting tests based on Prototype theory (34) to evaluate supposedly prototypical thumbnails of tracks. Huron states that "prototypical musical passages should tend to be (1) more easily recalled by listeners from one day to the next, and (2) generate false-positive recognition responses (i.e., listeners should incorrectly claim to have heard the passage before, having been exposed to other excerpts from the same work)" (34). Such a test could be conducted by comparing the automatically selected clips to random or hand chosen clips.

Prototype theory also suggests that preview clips may be selected that better represent all the other possible clips or the whole track. Given a suitable parameterisation of the audio for a track, we should be able to select an appropriate, representative clip by selecting the clip that has the most similar parameterisation to the whole track or to all the other possible clips within the track.

Cooper and Foote (18) present a technique based on such an approach. The audio signal is parameterised by the computation of MFCCs for relatively long (92.87 ms) analysis frames of the signal. The MFCC vectors are used to compute a Cosine similarity matrix for each frame to each other frame. The relatively long analysis window may have been chosen to reduce the computational compexity of the the subsequent comparisons used to select the preview clip. Both dimensions of the similarity matrix are the length of the track in frames.

Regions of high self-similarity within a track will appear as square patterns along the main diagonal of the similarity matrix, while repeated sections will appear as off-diagonal squares. The more often a particular segment is repeated the more square off-diagonal patterns of high similarity it will produce. Hence, the similarity of a particular clip to the rest of the track can be computed by summing all the cells in the similarity matrix for the rows (or columns) that correspond to that particular clip of the track. To extract a fixed length clip, the summation is performed for all possible clips of the required length and the clip producing the highest total is selected.

Limited evaluation is performed by demonstrating that the important characteristics of two tracks were selected as part of the preview clip (18).

An alternative approach to preview clip selection, described by Eronen (30), uses the assumption that the chorus of a song will be the most repeated part and will likely contain the 'hook'. Therefore, a clip containing the chorus of a particular is likely to form the best preview of the track as a whole. Hence, Eronen defines a system to select the chorus regions in a track and uses this as a (variable length) preview clip.

Chorus detection is performed by calculating MFCCs and Chroma vectors for beat synchronous frames of the audio signal. Chroma vectors are computed by mapping each bin of a spectrogram to one of twelve pitch classes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B) and summing the energy. A Euclidean similarity matrix for the MFCCs and Chroma vectors is computed and the Chroma similarity matrix is enhanced to reduce the effect of slight variations in the performance and to accentuate any diagonal stripes of high similarity. The two similarity matrices are then summed to produce the final track self similarity matrix.

Next, repetitive sections within the track are selected by analysing the diagonals of the lower triangular portion of the similarity matrix. Only the diagonals representing the highest N total similarity scores are retained, smoothed and thresholded to produce a binary summary of the similarity matrix.

Finally, the chorus is selected through the application of a number of heuristics based on the position of the repetition in the similarity matrix, relationships to other repeated segments, the number of times the repetition occurred and the average energy (selecting the loudest repeated segment).

The system is evaluated on the retrieval of hand annotated chorus sections in 206 popular and Rock music tracks and achieves an F-measure of 86% (precision 89%, recall 83%), while a version of the system designed to extract fixed length clips achieves 79% F-measure (precision 70%, recall 92%), due to reduced precision caused by the fact that many choruses are shorter than 30 seconds, forcing the clip to encompass material that is not part of the chorus.

Logan (46) defines another system that uses a track's verse/chorus structure to select a good preview clip including the chorus. Again, MFCC vectors are computed from the audio signal. A bottom-up clustering technique based on a modified cross-entropy or KL divergence–based distance measure and a hand-tuned stopping threshold is applied to the vectors and the most commonly occurring cluster is assumed to be the chorus region. A fixed length clip is selected from one of the estimated chorus clusters.

The system is evaluated by asking users to rate each clip as good average or poor (3, 2 and 1 points respectively). The performance of this system compares favourably with a system that replaces the clustering process with Hidden Markov models and randomly selected clip sets.

2.5 Challenges in music similarity estimation

In this section some of the major research and experimental challenges for MIR research are examined.

2.5.1 Avoiding over-fitting

Pampalk (55) identifies the potential for the over-fitting of the characteristics of a particular artist to inflate accuracy scores in the evaluation of audio content-based genre classification systems, particularly when evaluating on small collections. If a system is queried with a track from an artist who also appears in other tracks with known genres, classification may be more accurate than for tracks from artists that don't appear in the training set because it may match the artist's other tracks on artefacts such as production characteristics rather than music characteristics. Pampalk reports experimental results with an audio content-based similarity estimator and a k-Nearest Neighbour classifier that show upto a 48 % point drop (from 76% - 28%) in the accuracy score on a 22-class genre classification problem.

Hence, when separating data into training and test sets, the data should be "artist-filtered", ensuring that no artist in the training set of a classification experiment also appears in the test set. Artist-filtering should also be applied to search results from music similarity systems (i.e. results from the same artist as the query are ignored) to prevent inflated estimates of performance in both human and statistical evaluations.

Kim, Williamson and Pilli attempt to quantify a similar effect on artist classification performance scores based on tracks form the same album (41). Often, the tracks on a particular artist's album or a compilation album of various artists' tracks will be modified in post-production to ensure a more consistent audio experience and volume level. Further, the dynamic range of the audio may also be modified to ensure playback on lower quality audio equipment does not produce distortion (41).

It is possible that these distortions could lead to over-fitting and hence observed increases in performance when material from the album from which the query was drawn exists in the training set. Kim, Williamson and Pilli demonstrate that systematic distortions can be observed in the features extracted from re-mastered versions tracks and may give rise to the increase in performance

known as the album effect, by enabling a model to learn incidental audio characteristics instead of the intended musical characteristics. A stated in section 2.5.1, such overfitting maybe avoided with genre neighbourhood clustering through the use of an artist filter.

2.5.2 Ground-truth problems for music similarity estimation

The difficulty of evaluating music similarity measures is well-known in the Music Information Retrieval community (56) as there exists no database of direct human applied similarity scores between pairs of tracks, as such measurements would be prohibitively expensive to collect for even moderate size databases. Further, we should not compare performance to similarity estimates produced by other types of similarity estimator (such as a Collaborative filtering system) as these systems only 'predict' human similarity estimates and may make there own mistakes or demonstrate their own idiosyncrasies. Hence, as we are trying to emulate the human perception of the similarity between two tracks, practical human tests must be defined. However, these are expensive and time-consuming to perform and automated procedures have also been developed.

2.5.3 Evaluating music similarity searches

Owing to the fact that music similarity estimators are attempting to mimic human perception of music similarity, evaluations of performance must be based on the measurement of similarity estimates by humans. Human testing of music similarity searches has been conducted individually by a number of authors. The collection of direct similarity ratings by human beings of pairs of tracks is too expensive to provide a useful evaluation procedure. Further, humans may not naturally estimate a uni-dimensional similarity of two tracks on a known scale, making such judgements hard to normalise. Hence, comparative evaluations of systems are conducted instead, which produce a ranking of the compared systems and may be used to identify statistically significant differences in performance between those systems.

2.5.3.1 MIREX

The International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) in the Graduate school of Library Information science at the University of Illinois conducts annual evaluations of many Music Information Retrieval (MIR) algorithms including comparative evaluation of audio music similarity estimation systems. The annual evaluations are known as the Music Information Retrieval Evaluation eXchange (MIREX) (28).

The procedure for human testing chosen by the MIREX audio music similarity task participants in 2006 and 2007 is based on the procedures used by the evaluations performed as part the Text Retrieval Conference (TREC). A set of random query tracks are chosen from the collection and the top N matches for each track is collected from each system. The results are artist-filtered to ensure that estimates of performance are made on the retrieval of similar music from *other* artists. The results from each system are collected together to form a 'retrieved-set' which is then scored by a user or users with no knowledge of which system returned the tracks. Users are presented with the entire retrieved set for each query to ensure that each result is evaluated in the same local context (as the scores applied are both subjective and arbitrary and might change in the presence of better or worse matches). The final score for each system is calculated from the individual track evaluations and the systems are ranked accordingly.

Two different scoring mechanisms are used as part of the MIREX procedure, including both a three category score (similar, somewhat similar, not similar) and a fine score ranging from 0.0 - 10.0. Despite the application of different scoring schemes to the three category score (i.e. (3,2,1), (4,2,1), (1,1,0)) these different mechanisms have, so far, produced a consistent ranking of systems (38). Finally, the Friedman test or Friedman's ANOVA is applied to the results in order to conduct a statistically valid pair-wise comparison of each of the system results and to determine if any differences in ranking are significant. Friedman's ANOVA is a non-parametric test used in preference to Student's T-test as it does not assume normal distribution of the underlying data.

Unfortunately, the running of such an evaluation requires a large number of human test subjects. This is due to the fact that the number of individual similarity judgements that must be made is given by:

$$N = Q * L * E * (\alpha (1 - S) + S) \tag{2.3}$$

where N is the number of query result pairs that must be evaluated, Q is the number of queries, S is the number of systems to be compared, L is length of the result lists to examine, E is the number of evaluators that must check each query result pair and α is the average degree of overlap between the results from the different systems involved (the proportion of returned search results that have already been returned by another system). At MIREX 2006 the degree of overlap between the results of the different systems was extremely low - indicating that the worst case complexity should be considered (38).

Hence, for five systems, evaluated on sixty queries, returning five results per query and three different evaluators per query - we would require 4500 similarity judgements. Assuming a rather short one minute per similarity judgement, we would require around three hours of evaluation work from each of 25 volunteers.

2.5.3.2 Pseudo-objective evaluation of music similarity measures

Given the high cost and relatively low coverage that can be achieved using human evaluations, several authors have presented evaluations of their audio search systems based on statistics of the number of examples bearing the same label (genre, artist or album) amongst the N most similar examples to each song (*neighbourhood clustering* (47)), or on the distance between examples bearing the same labels, normalised by the distance between all examples (*label distances* (71)). In this work, the neighbourhood clustering statistics are reported as it is thought that they provide the strongest indications of the performance of each system at *recommendation*, based on the assumption that results from the same album, artist or genre and likely to be relevant recommendations. In contrast, label distances require all examples of a particular class to be tightly clustered in order

to yield good results. Within varied genres of music this may not be as appropriate a statistic as neighbourhood clustering for estimating recommendation performance.

These automated statistical evaluations are sometimes referred to as pseudo-objective statistics. The term pseudo-objective is used as the statistic most appropriate to the evaluation of music recommendation performance, genre neighbourhood clustering, is based on objective measurements of the agreement between the tracks retrieved by a system and a set of subjectively applied genre labellings. Genre neighbourhood clustering is considered to be a stronger metric than either artist or album neighbourhood clustering because of the potential for the overfitting of production characteristics, rather than the desired musical characteristics, when performing either artist or album clustering. As stated

The results from MIREX 2006 indicated that the Neighbourhood clustering metrics were correlated with the large-scale subjective human evaluation, producing the same ranking of five similarity estimation techniques from different sources (27) at 5, 10, 20 and 50 results. Hence, we report the genre, artist and album clustering at 20 results.

2.5.4 Performance of current techniques

Several authors have speculated on the possiblility of a 'glass-ceiling' on the performance of current audio music classification and similarity estimation techniques. As identified by Aucouturier (2) many of these techniques are based on 'bag-of-frames' approaches to the comparison of the audio streams. A further problem with many of these technologies is the presence of 'hubs', that is a small number of tracks that closely and erroneously match a great number of other tracks. Although there are usually few 'hubs' in the tests conducted, they adversely affect the search results of a considerable portion of the collection and increase in number as the collection size is increases. Hence, these 'bad' results will become a serious problem when producing large scale systems based on these techniques.

2.5.5 Practical considerations for large scale music similarity estimation

It is likely that different indexing techniques will be effective over small ($< 10{,}000$ tracks), medium (10,000 – 100,000 tracks) or large ($> 100{,}000$ tracks) scale music information retrieval systems. At present the largest digital music store, iTunes, contains over 6,000,000 tracks [1].

Search indices on this scale pose much greater problems for the provision for practical search times and computational resources than the small scale experiments often reported. For example, a system recording a modest 32 Kb of data per audio track would only require 312 Mb of RAM to load the features for 10,000 tracks but would require nearly 31 Gb of memory to load the features for a million tracks. Such a requirement can be satisfied by modern servers but might prove prohibitively expensive to scale when such a system is accessed by many concurrent users. Hence, these requirements should be limited, where possible, to the off-line production of an index of the collection in such a way that the index may be reloaded queried using significantly less resources.

Consideration should also be given to the size of the descriptors extracted from the audio signal. If these are extracted on a client user's machine the footprint of the extraction, extraction time and the data transmitted should all be as small as possible. Further, if any form of customisation or client side modelling of features is to be performed (e.g. the creation of ad-hoc genre, mood or playlist models), compact yet powerful descriptors for music audio will be needed.

Further, most of the similarity estimation techniques that have been demonstrated in the literature have been based on the application of distance measure to features computed from the audio tracks in the collection, leading to a linear scaling of computation time with collection size (given sufficient resources). Hence a system effectively returning results in 100 ms on a 10,000 track collection might require a full 10 seconds to conduct a search of a 1,000,000 tracks.

If we are to provide efficient music search techniques, either much faster distance computations or more efficient indexing strategies that scale sub-linearly

[1] http://www.apple.com/itunes/store/ - retrieved 1st September 2007

with collection size will have to be found. In practice, an acceptable limit for the search time, providing near instant results to the user, is around 200 ms.

Casey and Slaney (15) present a system which scales sub-linearly with collection size for a task related to generalised music similarity estimation that finds derivative works or remixes of a given track. Casey and Slaney acknowledge that finding derivatives of a track requires a more specific search than generalised music similarity search but think that their technique might be extended to also find similar tracks.

Their approach is based on technique known as Locality Sensitive Hashing (LSH) which is used to index feature vectors known as audio shingles. The audio shingles are short fixed length sequences (3 second) of chroma and MFCC-like feature frames computed from the audio file. Sequences of feature frames are used as the sequential information is essential for the finding of derivative works. Conventional hashing techniques attempt to assign similar entities to different hash bins to ensure that specific examples maybe retrieved. In contrast LSH attempts to ensure that similar examples are hashed to the same locations. The full index is built by creating an ensemble of individual hash indices based on random projections of the data. Items that are somewhat similar are expected to fall into the same hash bin in each of these hash indexes. By collecting up the number of collisions (times that a track was in the same hash bin as another track) across all the hash indexes, tracks can be ranked by their similarity to a track.

This LSH-based approach appears to accurately duplicate the performance of the full linear search of the collection with a much shorter search time (1 hour for 20 queries of a 306 track collection v.s. 7 hours for the full search). However, the technique has not been tested on a large scale collection or directly evaluated on generalised music similarity search and the reported search times are well outside the acceptable range for many music search applications.

Chapter 3

Representing Audio Music Data

In order to implement any form of audio content based similarity analysis, a suitable set of features must be calculated from the audio signal. The aim of this feature extraction is to represent the signal in a lower dimensional form that better represents the high level characteristics of the music. In this work, we use features describing the spectral envelope, primarily related to the timbre of the audio, which can be used to define a 'timbral' similarity or classification function. We also define new features that provide information about the rhythmic content of the signal. However, the classification and indexing techniques introduced in later sections could be extended to form other types of similarity function, such as melody or harmony, by simply replacing these features with other appropriate features.

3.1 Audio pre-processing

In order to calculate features from the audio signal we first pre-process the signal with fairly standard digital signal processing (DSP) techniques, commonly used in other audio analysis domains, particularly speech processing and music/speech discrimination. These techniques are based on the estimation of a magnitude spectrum for sequences of overlapping windows of the signal using the Fast Fourier Transform (FFT).

Specifically, the audio signal is first divided into a sequence of 50% overlapping frames of length 20–30 ms, over which the magnitude spectrum of the signal is

assumed to be stationary or at least varying very slowly. A Hamming window, given by equation 3.1, where w_q represents the weight applied to the q-th sample in the window and n is the length of the window, is applied to each frame using equation 3.2. Here, f_q represents the q-th sample in the frame and f'_q represents the q-th sample in the windowed frame.

$$w_q = 0.54 - 0.46 * cos\left(\frac{2\pi q}{n-1}\right) \tag{3.1}$$

$$f'_q = w_q * f_q \tag{3.2}$$

The application of a Hamming function to each window of the signal minimises the effect of the framing on the magnitude spectra estimated, which would otherwise have spurious high frequency components added to it.

The magnitude spectrum of the window is estimated using the Fast-Fourier transform (FFT) by applying equation 3.3.

$$x(i)_k = \left|\sum_{m=1}^{N} f'_m e^{-\frac{2j\pi mk}{N}}\right| \tag{3.3}$$

where $x(i)_k$ represents the k-th frequency magnitude coefficient of the i-th frame and N represents the window size (for example, 1024 samples at 44100 Hz, representing approximately 23 ms of the signal). Only the non-redundant half of the FFT magnitude spectra, representing the positive frequency values is retained.

The following sections describe the calculation of features from this spectral decomposition, including two sets of timbral features: Mel-Frequency Cepstral Coefficients (MFCCs) and Spectral distribution descriptors (SDDs); and a novel set of features intended to describe short-time rhythmic information in the signal: Rhythm Cepstral Coefficients (RCCs).

3.2 Representing Spectral and Timbral information

In this work the timbre of the audio signal is represented using a feature set similar to that described by Tzanetakis (66) (65). It is composed of the first 15 MFCCs, ignoring the zero-th coefficient (which represents the DC level of the signal and is generally assumed to be zero), and a number of coefficients that describe the distribution of the magnitude spectrum of the audio, such as the Skew or Centroid.

3.2.1 Mel-Frequency Cepstral Coefficients

Mel-frequency Cepstral Coefficients (MFCCs) are a compact representation of an audio spectrum originally formulated to capture important characteristics of speech, (20). MFCCs have (already) been used by a number of authors to parameterise a music audio stream for automatic classification and similarity estimation (10) (47) (51) (53) (55) (66).

The MFCCs are the result of a cosine transform of the real logarithm of the short-term magnitude spectrum after it has been passed through a Mel-frequency scale filterbank. The Mel-frequency scale filters are intended to duplicate the (known) distribution of the ear's critical bandwidths with frequency, using filters placed roughly linearly at low frequencies and logarithmically at higher frequencies. Figure 3.1 shows the weights applied to the non-redundant half of the FFT magnitude spectrum, in order to simulate a 24 band Mel-frequency filterbank. Humans perceive the magnitude of each of these critical bands logarithmically. Hence, the logarithm of the output of each of these filters is used.

The Cosine transform of the Mel-frequency filter outputs is used as it has been shown to provide a good approximation to the optimal de-correlation of the filter outputs produced by a Principal Components Analysis (PCA) on speech material. Further, it can be computed without needing to perform a computationally intensive Eigenvector Decomposition or Singular Value Decomposition of the covariance matrix of the dataset used to implement the PCA. Logan (48)

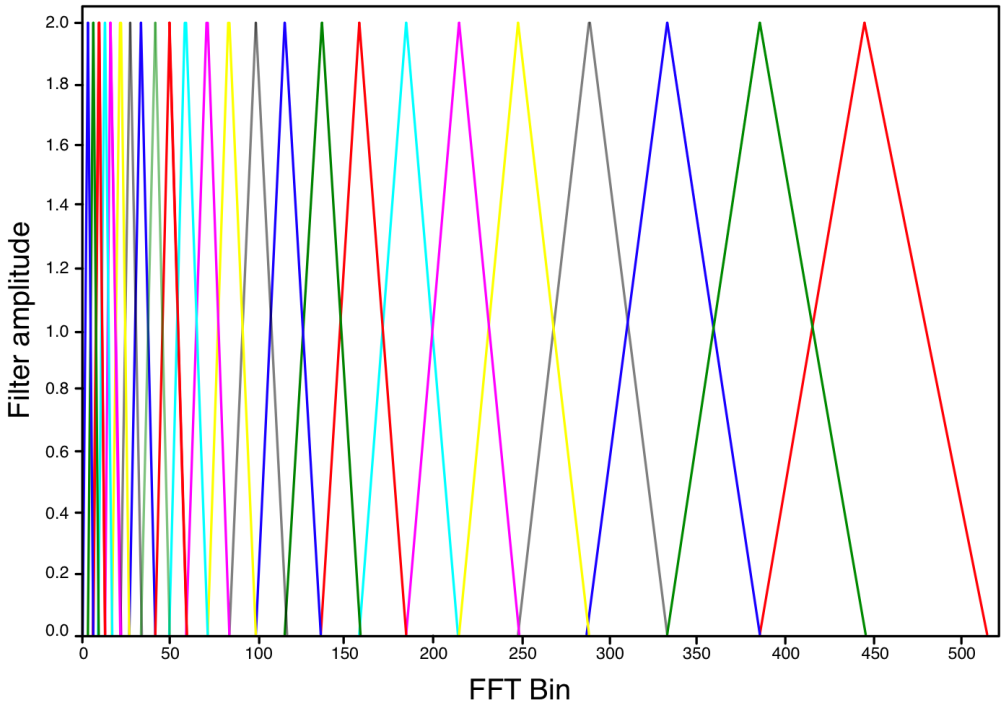


Figure 3.1: Weights applied to FFT bins in order to simulate a 24 channel Mel-frequency filterbank for 23 ms frames of a 44kHz audio signal. Each set of weights used to simulate a particular filter are shown with a different colour.

demonstrated that these properties also hold for music audio data and therefore MFCCs represent both a powerful and compact representation of the audio data.

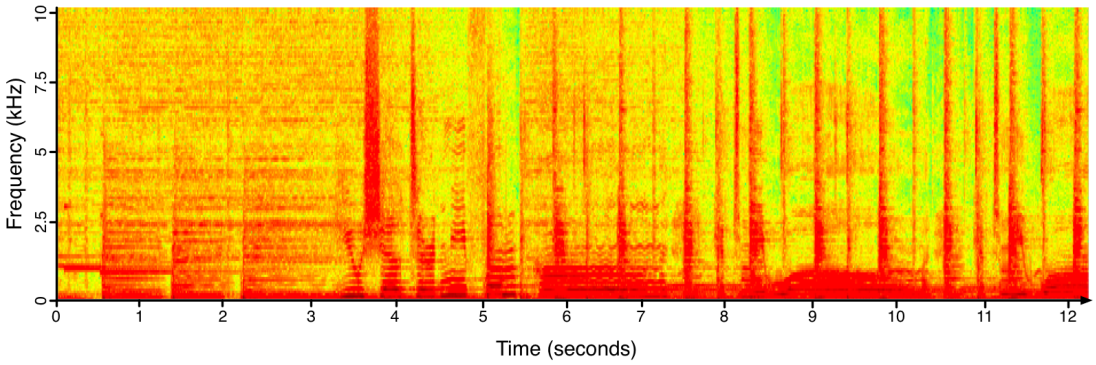
In these experiments, 40 triangular bandpass filters placed on a mel-scale ($b = 40$) are simulated by applying weights to the magnitude FFT of the audio signal calculated in equation 3.3. The logarithm of the outputs of the filters is taken to give a set of coefficients $m(i)_k, k = 1, 2, ..., b$. The MFCCs are then calculated as:

$$MFCC(i)_c = \sum_{k=1}^{40} m(i)_k cos\left(c\left(k - \frac{1}{2}\right)\frac{\pi}{40}\right), c = 1, 2, ..., C \tag{3.4}$$

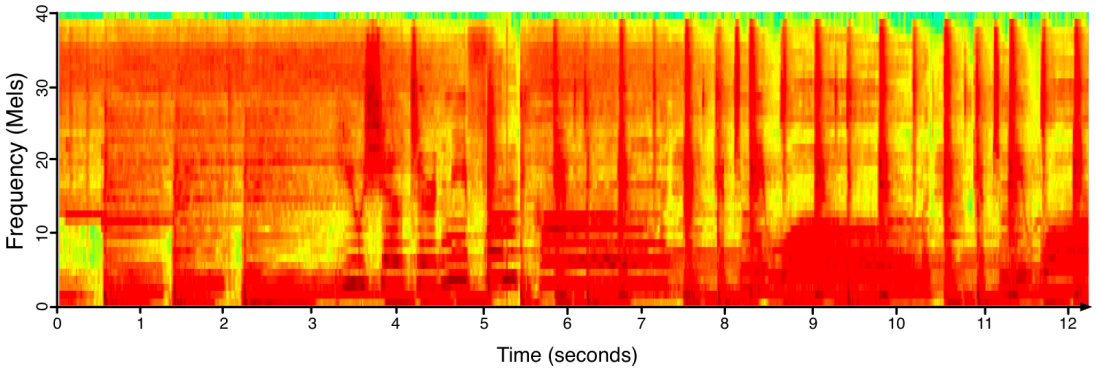
where C is the number of cepstral coefficients to be estimated. In this work we retain the first 15 MFCCs, ignoring the zero-th coefficient. Figure 3.2 shows plots of spectra produced by the calculation of Log FFT coefficients, Log Mel-filterbank coefficients and Log DCT coefficients.

In speech research it is common practice to also calculate the delta and acceleration values of the MFCC vectors (the first and second order differences between concurrent frames). In preliminary experiments, little benefit has been found for calculating the acceleration coeficients, while limited gains in performance were achieved using the delta coefficients. Hence only the raw MFCCs or raw MFCCs and delta coefficients are calculated.

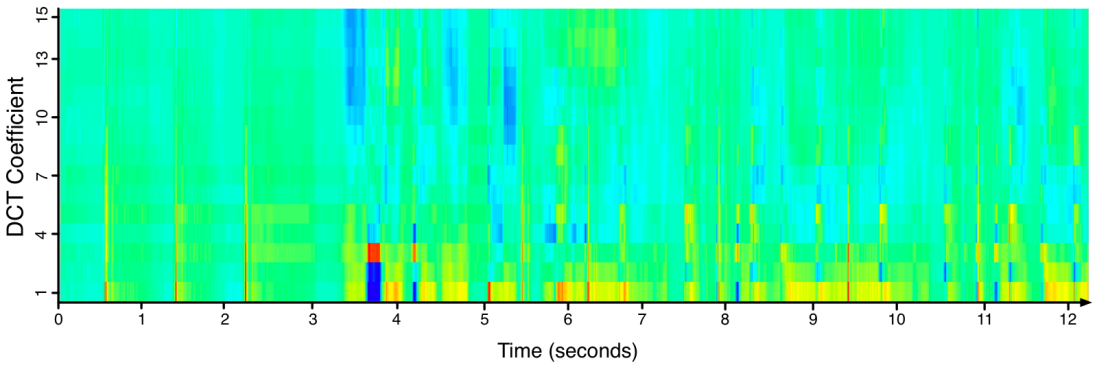
Several alternatives to Mel-Frequency coefficients for the modelling of musical timbre have been proposed, including features based on the Bark-scale, Octave-scale Spectral Contrast feature (36) and Mel-frequency Spectral Irregularities (MFSIs) (71). MFSIs are calculated from the output of a Mel-frequency scale filterbank and are composed of two sets of coefficients: Log Mel-Frequency Spectral Coefficients (as used in the calculation of MFCCs, without the Discrete Cosine Transform) and Mel-Frequency Irregularity Coefficients. Both Octave-scale Spectral Contrast features and the Mel-Frequency Irregularity Coefficients include a measure of how different the signal is from white noise in each band. This helps to differentiate frames from pitched and noisy signals that may have similar average spectral characteristics, such as certain string instruments and drum types, or to differentiate complex mixes of timbres with similar spectral envelopes. Both



(a) FFT Log Magnitude Spectrum



(b) Log Mel-filterbank coefficients



(c) MFCC coefficients

Figure 3.2: Comparative plots of FFT log-magnitude spectra, Log Mel-filterbank spectra and the first 15 MFCC coefficients for a 12 second music clip

feature sets include a significant degree of correlation between bands and must be decomposed using a PCA or similar transform.

However, experiments have indicated that these features may be particularly vulnerable to changes in the audio spectrum produced by many common codecs (including MPEG 2 Layer 3 (MP3)) which remove partials from the spectrum where they are masked by larger partials in the same critical band and are therefore not perceptually salient. This yields a systematic deviation from the original spectrum, which it is possible for a model to learn rather than the actual characteristics we are attempting to model. As it is likely that our music similarity and classification techniques will have to be applied to data from a number of sources and codecs, we have chosen to use the more robust MFCCs.

3.2.2 Spectral Distribution Descriptors

In order to enhance the description of the shape of the spectral envelope given by the MFCCs, a number of additional coefficients are calculated to describe the distribution of the spectrum and are thought to be indicative of perceptual characteristics of the audio. The use of many of these Spectral Distribution Descriptors (SDDs) was examined in detail by by Tzanetakis (65) (66). However, they have been augmented with several additional distribution descriptors.

The SDD coefficients are calculated from the magnitude frequency spectrum $\vec{x}(i)$, given by equation 3.3, and include:

Flux The difference between concurrent frames, given by:

$$Flux(i) = \sum_{k=1}^{\frac{N}{2}} |x(i)_k - x(i-1)_k| \tag{3.5}$$

where $X(i)_k$ represents the k-th FFT coefficient of the i-th frame.

Centroid the centre of gravity of the spectrum, given by:

$$Centroid(i) = \frac{\sum_{k=1}^{\frac{N}{2}} kx(i)_k}{\sum_{k=1}^{\frac{N}{2}} x(i)_k} \tag{3.6}$$

RMS The 2Norm or Euclidean norm (related to volume), given by:

$$2Norm(i) = \sqrt{\sum_{k=1}^{\frac{N}{2}} x\left(i\right)_k^2} \tag{3.7}$$

Entropy The entropy of the spectrum (higher values of which indicate a flatter spectrum which is likely to contain a higher level of noise than a spectrum with a lower level of entropy) The spectral entropy is calculated with an equally spaced 20-bin histogram and is given by:

$$Entropy(i) = -\sum_{m=1}^{20} p_m ln\left(p_m\right) \tag{3.8}$$

where p_m is an estimate of the probability of the m-th bin of the histogram of $\vec{x}(i)$:

$$P_k = \frac{x_k}{\sum_{m=1}^{20} x_m} \tag{3.9}$$

Variance The sample variance of $x\left(i\right)$, describing how the FFT magnitudes very from their mean value:

$$Var(i) = \frac{\sum_{k=1}^{\frac{N}{2}} \left(x(i)_k - \mu\left(i\right)\right)^2}{\frac{N}{2}} \tag{3.10}$$

where $\mu\left(i\right)$ represents the mean of $\vec{x}(i)$ estimated from the sample mean $\bar{x(i)}$.

Skew The sample skew, describing the asymmetry of the distribution of $\vec{x}(i)$ around its mean (μ), is estimated as:

$$Skew(i) = \frac{\sum_{k=1}^{\frac{N}{2}} \left(x(i)_k - \mu\right)^3 \sqrt{\frac{N}{2}}}{\left(\sum_{k=1}^{\frac{N}{2}} \left(x(i)_k - \mu\left(i\right)\right)^2\right)^{\frac{3}{2}}} \tag{3.11}$$

Kurtosis The sample kurtosis, which describes the extent to which the distribution of $X(i)$ is bunched around the center or spread toward the endpoints, given by:

$$Kurtosis(i) = \frac{\left(\frac{N}{2}\sum_{k=1}^{\frac{N}{2}} \left(x(i)_k - \mu\right)^4\right)}{\left(\sum_{k=1}^{\frac{N}{2}} \left(x(i)_k - \mu\left(i\right)\right)^2\right)^2} - 3 \tag{3.12}$$

Roll-off The Spectral Rolloff, which is the point on the frequency scale below which 85% of the total amplitude lies, is computed as:

$$Rolloff(i) = R \tag{3.13}$$

such that $\sum_{k=1}^{R} = 0.85 \sum_{k=1}^{\frac{N}{2}} x(i)_k$.

An overview of the basic timbral feature extraction process is given in figure 3.3, where the stream of 23ms frames of the audio signal are converted into a stream feature vectors. This process produces a stream of 38 feature values, outputting one vector every 11.5 ms.

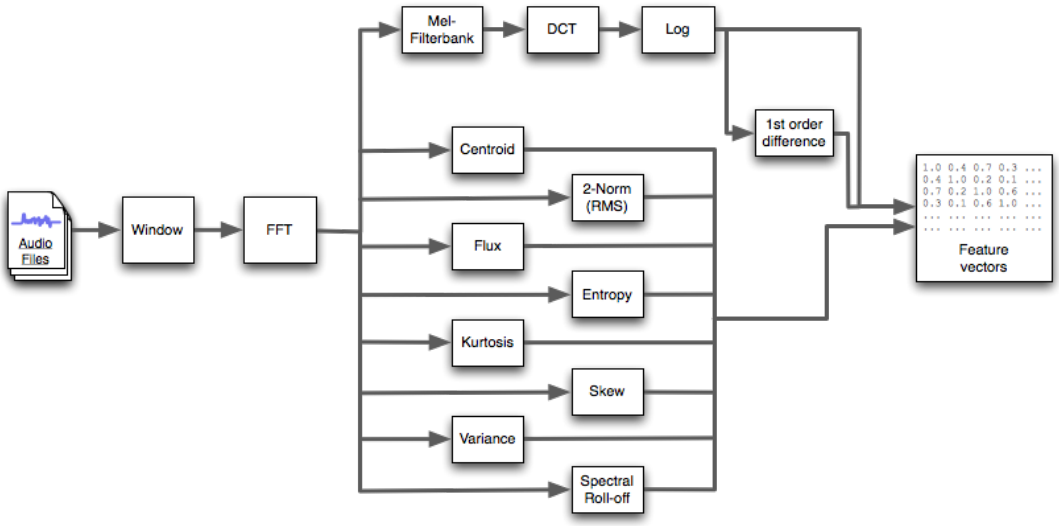


Figure 3.3: Overview of the basic feature extraction process, which outputs a stream of feature vectors each representing 23ms of audio.

3.2.3 Segmentation of feature streams

In this work audio features are calculated over short periods (23 ms with 50% overlap) within which the signal is assumed to be stationary or at least varying slowly relative to the frame size. This yields a very large number of feature vectors

for each file and in order to effectively and efficiently compare tracks these must be summarised, to produce a smaller number of more informative vectors.

One approach that has been effectively used by many authors (37) (45) (47) (49) (56) (66) (75) is to summarise the distribution of feature frames over the *whole track*. This may be performed by, for example, estimating a Gaussian distribution of their values over the track (a mean vector and covariance matrix of the features). Some authors have demonstrated systems based on mixtures of Gaussian distributions. However, experiments have shown little utility for these over a single distribution (55) despite significant additional computational load. Further, a single distribution can be efficiently represented as a single vector by concatenating the mean vector and the flattened, non-redundant portion of the covariance matrix. Hence, these summaries may be presented to distribution based comparisons (e.g. the KL Divergence), simple vector based distance measures (such as the Euclidean distance), or to a classification algorithm such as a Support Vector Machine or Decision Tree classifier).

Alternatively, feature frames may be processed individually and a method of integrating the results into a single classification or similarity estimate used. The 23 – 92 ms audio frames conventionally used to estimate audio feature values (as the signal is assumed to be stationary or at least varying slowly over this interval) are too short to be perceptually salient and so this approach would likely yield very poor performance for a high computational cost. However, a number of other segmentations of the feature stream may be used, and are explored in this section.

Several authors have used alternative segmentations of audio feature stream including the use of short segments (typically 200–500ms) (37), longer segments (30 seconds) (44) and the use of a sliding 1 second 'harmonic modelling' window which returns a 1 second window of the features around the current frame (66). Each of these segmentations requires that individual feature vectors corresponding to that segment be summarised either as the mean of the vectors or using a single Gaussian distribution with either a diagonal covariance matrix (mean and variance) or a full covariance matrix (mean and covariance). Hence, each of these segmentations returns multiple vectors of features per track which must either be further summarised or processed as a group.

Music is a stream-based phenomenom which may encompass a large number of streams within a single track, including different instrument voices, mixtures of voices and variations of each voice. Hence, compressing the stream of features containing these many modes to a single distribution over the whole file represents a drastic reduction in information. Further, the use of distributions over a whole file may preclude the use of the features in any real-time application and may be sub-optimal for the classification and comparison of arbitrary length segments of tracks. Aucouturier (2) contends that this 'bag of frames' approach (also) ignores the ordering information present in the stream of frames and leads to a 'glass-ceiling' on the performance of classification and similarity estimation techniques based on it. There may exist segments of music with substantially different characteristics within a track or other detail that a simple distribution over the whole track will be unable to capture. However, Aucouturier also experimented with a number of procedures to overcome this drawback, such as the use of Gaussian Mixture models rather than a single Gaussian distribution. However, the results of these experiments were disappointing, showing no improvement in performance for audio music classificaiton, despite gains in performance on environment sound classification and other audio-based classificaiton tasks.

3.2.3.1 Beat synchronous feature processing

Bello and Pickens describe an alternative approach to feature segmentation applied to the task of chord transcription (7) known as 'beat synchronous feature processing'. Their approach is based on the segmentation of the feature stream into windows equal to the beat period, synchronised with the beat (also known as the *tactus*). They describe the resulting features as 'a robust mid-level representation that incorporates both harmonic and rhythmic information' (7). The implementation of beat synchronous feature processing requires that the tempo or beat period be estimated and then the beat location tracked prior to the extraction of features, to ensure that the start of the feature windows computed coincide with the beat onset times.

The use of beat synchronous segmentation yields a significant improvement in performance over frame-based features for chord recognition. This increase in

performance may be because chord transcription is vulnerable to small variations caused by phrasing or ornamentation. Such variations may be disruptive to chord labelling but may be important in the estimation of musical similarity or other musical characteristics.

3.2.3.2 Event-level feature processing

Event-level feature processing, first introduced in (69), also attempts to provide a robust mid-level representation of the feature stream by summarising features over each 'event' in the audio stream instead of over the stream as a whole or at a beat level. An event is defined as any sound or concurrently occurring set of sounds in the audio stream that can be perceived by a human and localised in time. Such segmentation is performed as a relatively low-level process in real-time by the human auditory system. Further, human beings have developed music notation systems (such as Common Music Notation (CMN) or guitar tab) based on this segmentation, where each sound is encoded as note or chord. Hence, music may be transcribed to a score and reproduced at a later by another musician. Most such representations encode the music as individual events with certain characteristics, including duration, pitch/chroma, voice/timbre and volume. We may assume that each segment of the music corresponding to one of the these events is likely to be far more stationary than the stream as a whole or arbitrary windows that may contain several events. Hence, it would seem sensible to try and segment the music into individual events and to parameterise each of these. In this thesis we use the sequence of events to parameterise the stream as a whole.

Event-level segmentation of the audio stream aims to produce more informative features than those produced for individual audio frames, whilst generating significantly less data. The event-level feature vectors may then be summarised over a track or processed individually to retain much greater resolution than a track summary. In contrast to beat synchronous segmentation, event-level segmentation does not require the tactus to be estimated or tracked, which often requires a first pass over the audio and is both costly to compute and vulnerable to errors in the selection of the correct metrical level and tracking of the beat

(particularly in the presence of expressive tempo). This is due to the fact that event-level segmentation only requires the onsets of events to be identified rather than the beat itself, which may be performed both in real-time and as part of a single pass over the audio stream which both segments it and extracts features for each segment. This enables faster feature processing and a lower memory footprint for the extraction process as summary feature vectors need only be retained for each event and feature frames since the onset of the last event. Typically, event duration ranges from 127 ms to 1 second (representing a 10 to 90 fold reduction in the number of feature vectors retained compared to the full feature vector sequence). Further, the reduced number of feature vectors can significantly reduce the cost of any covariance matrix calculations used to summarise the features, offsetting the additional computational cost of computing the onset detection function.

Attempting to separate co-occuring events (events that occur at exactly the same time, such as a snare drum sound coinciding with a guitar chord) is beyond the scope of this work, but could be used to further disambiguate the audio features produced by returning a variable number of feature vectors for each event segment, containing features from a single instrument or voice.

3.2.3.3 Onset detection-based segmentation

In order to segment the audio stream into events we must first calculate a function that indicates the boundaries of each event. In order to perform several music information retrieval tasks, such as tempo estimation (22) or automated transcription, a class of algorithms known as onset detection functions are used. These functions reduce the audio stream to a single parameter per frame, indicating how likely that frame is to contain the 'onset' or beginning of an event in the audio stream (such as a single piano note, a snare drum sound or a collection of co-occurring sounds). Several common onset detection techniques are described and evaluated in (6) and (21) and the results of the annual MIREX contest provides a comparative benchmark for implementations of these and several other techniques (26).

In this work we use a Spectral Flux based onset detection function similar to that used in (21), except that it is based on the half-wave rectified differences of concurrent frames of the outputs of the Mel-frequency filters ($\vec{m}(i)$, also used in equation 3.4) rather than on the magnitude FFT ($\vec{x}(i)$) and is given by equation 3.14.

$$MSF(i) = \sum_{k=1}^{40} \frac{(m(i)_k - m(i)_{k-1}) + |m(i)_k - m(i)_{k-1}|}{2} \tag{3.14}$$

This function is then low-pass filtered to 20 Hz and a dynamic mean threshold is used to identify peaks that may correspond to event onsets. For a peak in the function to be identified as on onset it must be the largest peak within 60 ms either side of the current frame and must have a value greater than the mean of the last 120 ms of the function plus a small constant α and must therefore satisfy:

$$MSF(i) > \frac{\sum_{j=i-w}^{1} MSF(j)}{w} + \alpha \tag{3.15}$$

and

$$MSF(i) > MSF(j) \tag{3.16}$$

for all j such that $i - p \geq j < i + p$ and where $w = \frac{0.12fs}{0.5N}$ (yielding a 120 ms window), $p = \frac{0.6fs}{0.5N}$ (yielding a 60 ms window), fs represents the sample rate and N is the frame size.

This function achieves an average (across all audio classes) F-measure score of 0.737 on the MIREX 2006 dataset which is comparable to the scores achieved in the 2006 evaluation (0.734 (13), 0.726 (23), 0.762 (29) and 0.788 (60)). The Mel-spectral flux onset detector was chosen as it can be very efficiently computed using the same simulated Mel-filter outputs used to compute the MFCCs and therefore represents a minimal additional computational burden, whilst providing a similar level of performance to state-of-the-art onset detection tools.

An example of 6 seconds of the onset detection function, its dynamic detection threshold and the onsets selected are shown in Figure 3.4 with the audio spectrum shown for comparison.

Finally, as the transient or beginning of an event (the attack portion) may be more informative than the end (decay or release) we include the previous frame

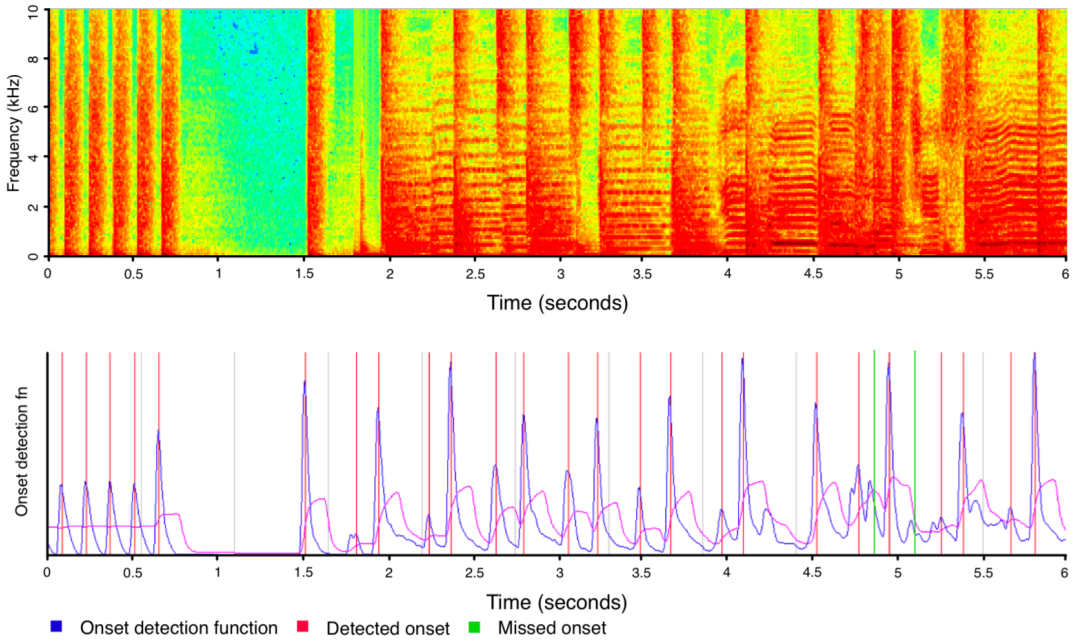


Figure 3.4: An example of 6 seconds of the audio spectrum and onset detection function (blue), the detection threshold (pink) and the onsets selected (red) for Bob Marley's 'I shot the sheriff'

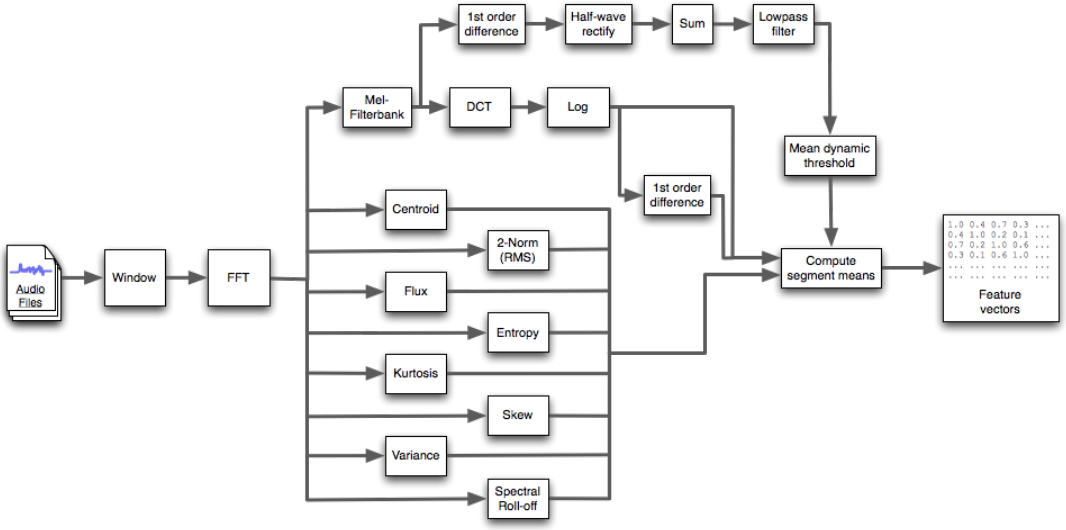


Figure 3.5: Overview of the segmented feature extraction process, which outputs a stream of feature vectors each representing a single audio event.

to ensure that it is fully captured. An overview of the segmented version of the feature extraction process is given in figure 3.5.

3.2.3.4 Benefits of event-level feature processing

As already stated, event-level feature processing attempts to encode the maximum amount of information found in frame-based features into a smaller amount of data, which may enable processing that is unfeasible on the quantity of data produced by conventional frame-based features. Event-level segmentation will ensure that timbre of extended audio events such as sustained notes or long silences are encoded in a single vector, yielding a reduction in the redundancy of concurrent feature vectors. By appending the segment length to the feature vector we can also ensure that the durations of those events are encoded, along with some limited timing information.

Figure 3.6 shows comparative plots of the first two principal components of features computed for 3 tracks from different music genres. The event-level features are obviously significantly more sparse than the frame-based features and the plot of the event-level features shows an extremely similar layout to the plot

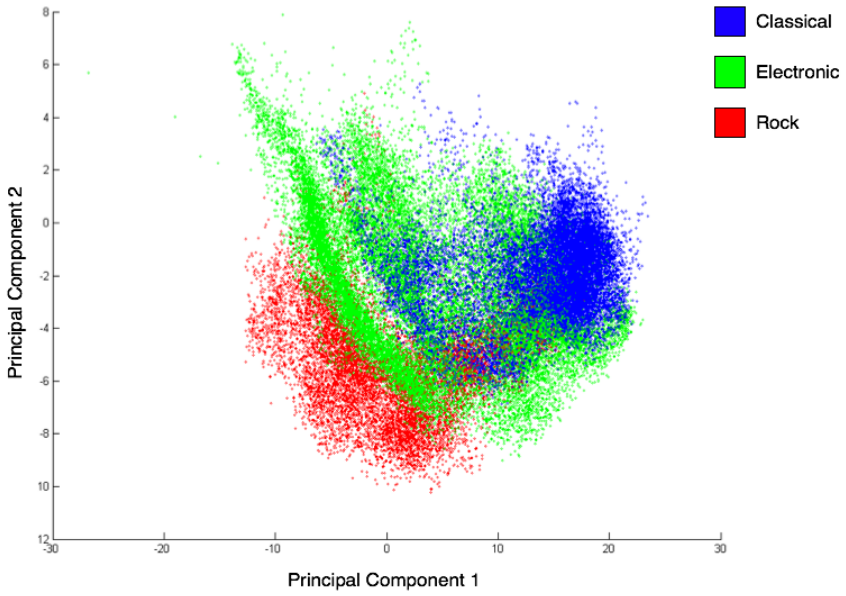
of the frame-based features, indicating that the major trends in the features have been preserved by event-level segmentation.

In order to use the features produced by event-level segmentation, methods of handling multiple vectors of features for each track must be used. The stream of features may be summarised to produce a single vector of summary features for the track, as described in section 3.2.4. An alternative approach for handling these features, without summarising the feature stream, is detailed in chapter 4 and the performance of the features produced by event-level segmentation of feature streams is compared to traditional frame-based features in section 4.3.4.

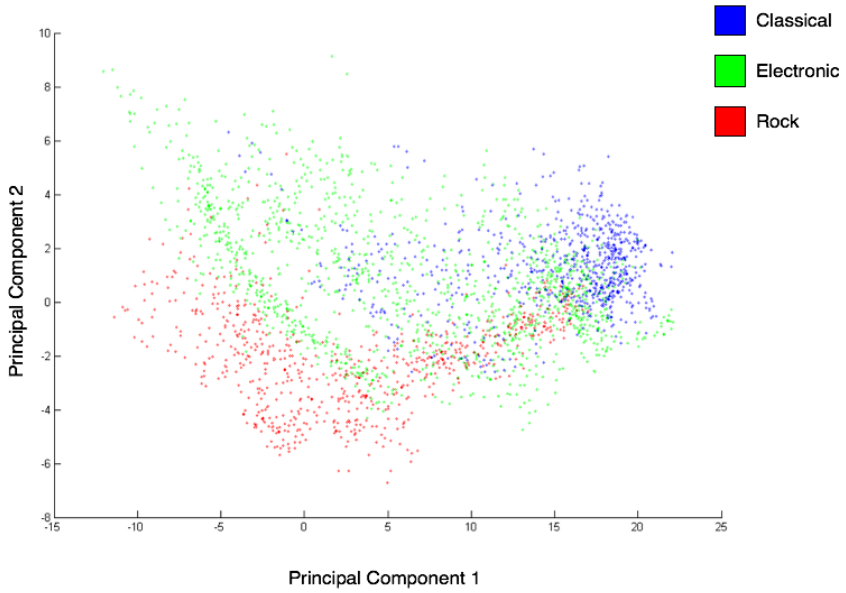
3.2.4 Summarising audio features

Whatever the segmentation chosen, the features corresponding to each segment must be summarised in some way to yield a single vector of features per segment. Despite differences in terminology, the major approaches to this summarisation in existing literature tend to be based on estimating the distribution of feature vectors over each segment. Several existing works in audio music classification, such as (66), model the features with a single Gaussian distribution with a diagonal covariance matrix (the mean and variance of the feature vectors). In (3) and (47) the distributions over whole files are modelled with Gaussian Mixture Models produced using the K-means algorithm. This yields a variable number of mixture components per file and therefore the distributions must be compared with algorithms capable of processing a variable number of features, such as the Earth Mover's Distance or Monte Carlo sampling. Later works have shown that the same level of performance can be achieved using a single Gaussian distribution with a full covariance matrix (the mean and covariance of the feature vectors)(50), (55), which allows us to take advantage of more efficient methods of comparison as the summarisation process will return the same number of coefficients for each track.

As there are often too few vectors for each event-level segment to estimate an accurate covariance matrix, we summarise each segment as either a single Gaussian distribution with diagonal covariance or simply the segment's mean vector. As the feature vectors for event-level segments tend to be far more homegenous



(a) Frame-based features



(b) Event-level features

Figure 3.6: Comparative plots of the first two principal components of the frame-based and segmented MFCC-based features for a Classical (red), Electronic (green) and Rock (blue) track

than over a whole track or fixed size segments of a track the variance is often very low. Hence we see little improvement in performance when using the variances as well as the means.

For comparative evaluation, results on full covariance Gaussian distributions of the features over whole tracks, fixed windows and segments (which are first reduced to mean vectors) are reported in sections 4.3 and 5.4. Hence, results on two systems that return a single vector of features per file (distribution of frames over whole file and distribution of segment means over whole file) and two systems that return multiple vectors of features per file (distributions of frames over fixed windows and segment mean vectors) are reported.

3.3 Representing Rhythmic information

Rhythm is the primary feature of dance music and in several sample-based electronic music cultures may be the only feature that separates different versions of a particular track into different dance genres. In many of these cultures the concept of *rhythm* must be considered independently of the *tempo*, as playback devices or performers may vary the tempo of a recorded track or performance piece, within a limited range, whilst the perceived rhythm remains the same. For example a Rhumba rhythm maybe played within a range of 110 - 150 BPM. Hence, it is our contention that any description of rhythm should be somewhat invariant of the tempo, which may form a separate feature.

In this work we are attempting to represent rhythmic characteristics of music, which would incorporate the concepts of beat, repetition and meter, rather than longer range characteristics such as a track's chorus-verse structure.

3.3.1 Existing work

A number of authors have introduced features that represent information on the rhythmic properties of the signal and use these features for the classification or similarity estimation of music.

Dixon et al. (24) address the classification of a specific subset of music, standard and Latin ballroom dance music, into ballroom dance music genres

using a classification method based only on timing information. They define two systems for extracting the periodicities in the signal, which are used to estimate the metrical hierarchy and timing patterns of each piece. The utility of each technique as input to a rule based classifier, which attempts to predict the style of the music, is examined.

The first approach to the estimation of the periodicities in an audio recording is based on the calculation of an onset detection function and the clustering of the inter-onset intervals estimated. The second approach, originally introduced in (57), uses autocorrelation on the amplitude envelopes of band-limited versions of the signal as its method of periodicity detection. The relationships between periodicities are then used to estimate the metrical hierarchy, including the tempo (at the beat level) and meter. The resulting predictions are then used to predict the style of music using a simple set of rules.

The performances of both methods are evaluated on a small test set consisting of 65 standard and Latin dance music tracks divided into 14 genres. The authors conclude that the autocorrelation method provides better performance than a system based on detected inter-onset-intervals, achieving a peak performance of approximately 80%. The authors state that the majority of these errors were caused by the incorrect selection of the metrical level.

There are two problems with this approach. The first is the necessity of pre-defining the classification rules (although the authors point out that automated learning techniques could be applied to the features generated if a sufficient test set were available for the training of a model). The second problem is that the various classes of dance music are not always separable based only on the tempo and meter estimates. These must often be augmented with an element selected from the periodicity distribution in order to achieve a reasonable classification accuracy.

Tzanetakis and Cook present another system for the classification of music into genres (65). This system leverages features that represent timbral, rhythmic and pitch related information in the signal. The individual sets of features are concatenated into a single feature vector and used as input to an automated

learning algorithm (Single Gaussian Model, Gaussian Mixture Model or K-nearest neighbour).

The features used to represent the rhythmic structure of music are based on the discrete wavelet transform, which provides low time and high-frequency resolution for low frequencies in the signal. The most salient periodicities in the signal are estimated by decomposing it into a number of octave-scale frequency bands using the DWT and computing the time domain amplitude envelope of each band (through full-wave rectification, low pass filtering, and downsampling). The envelopes of each band are then summed and the autocorrelation computed. The amplitudes of the peaks in these autocorrelation functions are accumulated over the whole file into a beat histogram.

The following features are computed from the beat histogram:

Relative amplitude the top two amplitudes divided by the sum of amplitudes of the histogram.

Amplitude ratio the amplitude of the second peak divided by the amplitude of the first peak.

Beat period the period of the first and second peak in beats-per-minute (BPM).

Sum the overall sum of the histogram, which is intended to indicate the strength of the beat.

The authors identify the strongest of the rhythmic features as the sum of the beat histogram. However, the authors also perform limited testing using each individual feature facet (pitch, rhythm and two timbre facets), and show that, though better than random, both the rhythm and pitch related features perform poorly in comparison to the timbral feature sets. However, the combined classification estimate does improve over the best performing individual facet (which is based on MFCCs).

Pampalk (55) describes a set of features known as Fluctuation patterns (FP) which describe periodicities in the modulation of the loudness of a set frequency bands in a signal. Pampalk proposes that these features represent information

in the Signal which is not available in features based on the average spectral envelopes, which are commonly used to represent the timbral profile of the signal. Given that periodicities are estimated upto 10 Hz, this additional can be thought of as pertaining to the rhythmic structure of the track analysed.

The extraction of periodicities found in the bands of both sonograms and Mel-frequency filterbanks is proposed. Pampalk indicates that the use of Mel-frequency filterbanks is preferable as the magnitude of each Mel-band is already estimated in the computation of MFCCs, which are used to represent the timbral profile of the signal. However, the 36 filter bands produced are grouped into just 12 bands because 30 frequency magnitudes are computed for each band and the use of the full 36 filters would yield an unmanageable number of features and the higher frequency bands are often highly correlated. Hence, 360 FP coefficients are computed.

In order to estimate the FPs, the Signal is first divided into 50% overlapped, three second windows, the spectrum is computed, passed through the Mel-filterbank and the periodicities in each filter band are estimated using an FFT. Hence, the author shows that a 2 minute audio track will generate 79 vectors of the 360-dimensional FP vectors. These may be summarised over a piece through the computation of their median values, yielding a final vector of 360 magnitudes per track, which may be compared using the Euclidean distance. However, Pampalk also shows that the median of the FP vectors may not well represent the patterns in all the vectors and tends to produce features that do not well separate many different genres of music (55).

Owing to the fact that the variable number of feature vectors produced by FPs are hard to process directly, Pampalk defines a number of simple summary features that may be computed from the FPs, including:

FP Max The highest value in the FP vectors, high values of which indicate a strong beat.

FP Sum The sum of the FP vectors, again, high values of which indicate a strong beat.

FP Bass The FP sum over the two lowest frequency bands.

FP Aggressiveness The ratio of the sum of values of the FP values for the periodicities below 1Hz to the sum of all of the FP values for all the periodicities. It is proposed that higher values of this ratio indicate relaxed pieces, while lower values indicate more aggressive pieces.

FP Domination of low frequencies The ratio between the sum of N highest frequency bands and the sum of the N lowest frequency bands, indicating whether there is a stronger beat in the lower or higher portion of the audio spectrum.

FP Gravity The centre of gravity of the FP vectors along the periodicity dimension (as opposed to along the frequency band dimension). It is proposed that lower values indicate slower tracks. However, Pampalk also notes that effects such as vibrato or tremolo may also be reflected in this value and hence what it indicates is not necessarily clear.

FP Focus The mean of all values in the FP vectors. It is proposed that this indicates whether energy is focused in small regions of the FP feature space, or spread widely. This may provide information as to the complexity and/or variation of the rhythmic structure as a 'focus' on small areas would indicate a simple structure, whereas widely spread energy could indicate a complex pattern or variations on a pattern.

Pampalk shows that the distance matrices produced by many of these features, when compared with the Euclidean distance, are highly correlated with each other. For example, FP Max is correlated with FP Sum and FP Aggressiveness and the median FP vectors are correlated with the FP Max, FP Sum, FP aggressiveness and FP Bass. The increase in performance of a set timbral genre classification algorithms is evaluated for each of these features and it is shown that the median FP vectors and the FP Gravity improve the performance of the combined system by significantly more than the other features. However, Pampalk also notes that overall, the increases in performance are minimal.

3.3.2 Rhythm Cepstral Coefficients (RCCs)

In this work we use a novel set of features known as Rhythm Cepstrum Coefficients (RCCs) to describe the short-time rhythmic information within a track. These coefficients are calculated in a similar way to MFCCs and share several important properties. RCCs make use of very similar audio pre-processing techniques to those used in the estimation of tempo and meter.

RCCs are based on the power spectral density (PSD) of the onset detection function given in equation 3.14. The onset detection function allows us to extract the timing information in the signal and abstracts any short-time spectral information. The PSD of this function can therefore be used to estimate the strength of any periodic repetitions present in the music. Due to the abstraction of spectral information, the repetition of particular sounds will not be encoded. However, the strength and shape of each onset in the onset detection function is reflected in the PSD. Were the sequence of detected onsets to be used instead, this information would be ignored and errors in the detection of onsets in the function might disrupt the estimation of the 'shape' of the rhythm. For example, a stronger onset every fourth beat might help to indicate the meter of a 4/4 rhythm. Similar conclusions were arrived at by Dixon, Pampalk and Widmer (24).

In order to calculate RCCs, the onset detection function is divided into 50% overlapping, 9 second segments and the PSD estimated. The onset detection function has already been estimated using 1024 frames of 44100 Hz signal, overlapped by 75%. This yields a sample rate for the onset detection function of 172.27 Hz. Hence, we are able to extract periodicities within this Signal up to approximately 86 Hz. As the onset detection function has a non-zero average power, the Fourier transforms may not exist (in such a situation they would have to be estimated by the averaging of many highly overlapped frames). This is highly computationally expensive, but fortunately both conventional Hamming windowing, as given in equation 3.1 and 3.2, and the Einstein–Wiener–Khinchin theorem (54) provide alternative methods of estimation. The Einstein–Wiener–Khinchin theorem states that the power spectral density of a wide–sense–stationary random process is the

Fourier Transform of the corresponding autocorrelation function (54). Hence, the PSD maybe calculated as follows:

$$Au(i)_m = \sum_{f=1}^{N} (MSF(f)MSF(f-m)) \tag{3.17}$$

$$PSD(i)_k = \left| \sum_{m=1}^{N} Au(i)_m e^{-\frac{2j\pi mk}{N}} \right| \tag{3.18}$$

where $Au(i)_m$ is the m-th autocorrelation coefficent of the i-th onset detection frame and N is the length of the onset detection function frame.

However, Hamming windowing of the onset detection function and a single FFT is less computationally intensive than applying an autocorrelation first and the PSD spectra estimated are very similar.

In order to improve both resolution and efficiency, the autocorrelation frame is zero padded to a power of two length before computation of the FFT. Hence, 1024 positive frequency points are computed over the range 0 - 86Hz, with a resolution of 0.084 Hz. A window size of 9 seconds was chosen to provide a sufficiently high resolution at the low frequency end of the spectrum.

In order to estimate tempo or meter, the next step would be to select peaks in the PSD that represent strongly occurring event intervals. Instead, the PSD coefficients of the onset detection function are passed through a simulated filterbank that is initially linearly scaled up to 2 Hz and then logarithmically scaled at higher frequencies (analogous to the Mel-scale filterbank used in the calculation of MFCCs). This ensures that the maximum frequency resolution is retained at the low frequency end of the spectrum, the less rhythmically relevant high frequency components are compressed, and that shifts in tempo do not distort the shape of the PSD-envelope excessively.

In the experiments reported here the transistion frequency (between linear and logarithmic scaling) of the filterbank is varied between 0 Hz (a fully logarithmic scale) and 4Hz. The logarithm of the outputs of the b simulated filters,

$L(i)_k, k = 1, 2, ..., b$, are used to calculate the RCCs by performing a discrete Cosine transform as follows:

$$RCC(i)_c = \sum_{k=1}^{b} L(i)_k cos\left(c\left(k - \frac{1}{2}\right)\frac{\pi}{b}\right), c = 1, 2, ..., C \tag{3.19}$$

where C is the number of cepstral coefficients to be estimated. These coefficients roughly encode the 'shape' of the short-time rhythm and are somewhat invariant of limited changes in tempo. The DCT also helps to decorrelate the individual dimensions of the RCC vectors as it does in the computation of MFCCs. In the experiments reported here the first 18 DCT coefficients are retained.

Spectral plots of an example of the Power Spectral Density (PSD) of the onset detection function, Log frequency scale PSD and Rhythm Cepstral Coefficients computed for a 30 second audio clip are given in figure 3.7.

As a final step the entropy, as given in equation 3.8, of the PSD coefficients is calculated to estimate the strength and complexity of the beat. Lower values indicate a strong regular beat and high values indicate a weaker or more erratic beat. A diagrammatic overview of the RCC extraction process is given in figure 3.8.

RCCs can encode interesting rhythmic information about a track. However, they do stop short of encoding specific rhythmic patterns as they abstract any information on the relative phase of the periodicities extracted from the onset detection function. Representation of specific short-time rhythmic patterns might be essential to the classification of certain types of music such as African drum rhythms.

Multiple vectors of RCCs will be returned for each track. These vectors may be summarised to produce a single vector or processed individually. As the number of vectors is likely to be quite low it is unlikely that we will be able to estimate a full covariance Gaussian distribution. Hence, the RCC vectors will need to be summarised as a mean vector or diagonal covariance Gaussian distribution.

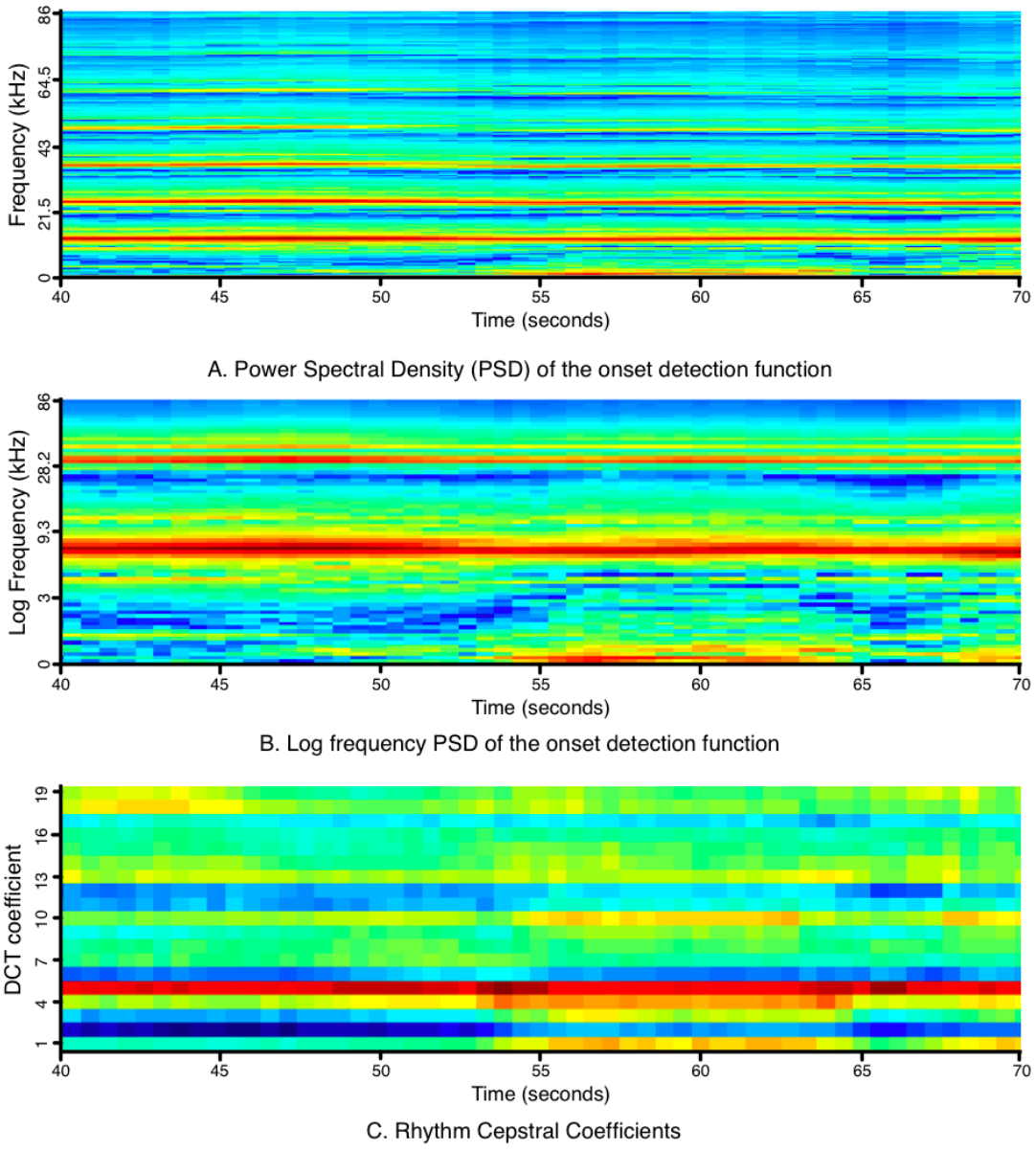


Figure 3.7: Comparative plots of the Power Spectral Density (PSD) of the onset detection function, Log frequency scale PSD and Rhythm Cepstral Coefficients computed for a 30 second audio clip from 'Bob Marley - No Woman No Cry'

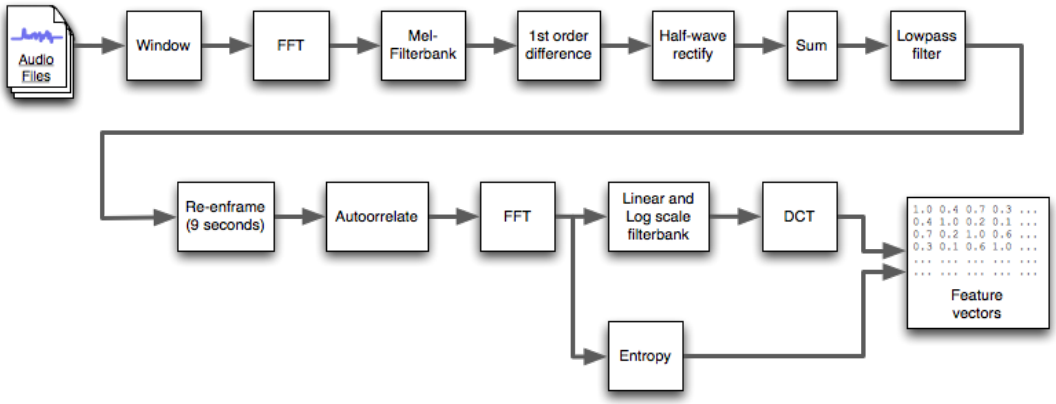


Figure 3.8: Overview of the rhythm feature extraction process, which outputs a stream of feature vectors each representing 9 seconds of audio.

Chapter 4

Automatic Classification of Music Audio

In this chapter the audio representations developed in chapter 3 are applied to the classification of music audio. The primary music audio classification task is the categorisation of music into music genres. However, the same techniques and systems have been successfully applied to a number of other music audio classification tasks; for example, at MIREX 2007, several systems were applied to the classification of music into genres, moods, composers and recording artists.

A number of different segmentations of the stream of timbral features, as described in section 3.2.4, are compared and a novel classification model is introduced that can handle multiple vectors of features per track. Existing approaches to the classification of music audio were surveyed in section 2.3.

4.1 Existing approaches to audio music classification

Audio music classification has been implemented by a number of authors through the application of classification algorithms to both automatically extracted high-level characteristics of audio (such as the tempo and meter (24)) and to low-level characteristics such as the features described in chapter 3.

In other fields, such as speech recognition or text document classification, the most successful approaches make use of mid-level representations, e.g. phones in speech recognition, words or N-grams of words in document classification. The obvious choices for a mid-level representation of music are the extraction of a score in common music notation (CMN) or a MIDI transcription. However, current systems for the automated transcription of polyphonic music do not yet provide a consistently high level of performance and remove a significant amount of useful information from the signal. Hence, systems that directly model low-level music features currently provide the highest level of classification performance on music with an unknown score.

A number of different classification algorithms have been applied to low-level features calculated from music, including: Single Gaussian (GS) classifiers (65), Gaussian Mixture Models (GMMs) (65) (3) (53), Nearest Neighbour (NN) algorithms (with Euclidean distance measurements (56), Earth Mover's distance measurements (56) and KL-Divergence based distance measurements (55)), Fishers Criterion Linear Discriminant analysis (LDA) (68), Neural Networks (NNet) (62), Hidden Markov Models (HMMs) (4) (39), Decision Tree (DT) algorithms (68), classifier ensembles (such as AdaBoost) (11) and Support Vector Machine (SVM) classifiers (49) (75) (45). Hierarchical classification approaches have also been applied, which use a known tree structure linking the lowest-level genres into higher level meta-genres (66) (14). This approach is often implemented by classifying a track into a limited number of high-level genres and then recursively applying classifiers intended to divide the music into the child genres of that higher-level genre.

Attempts have also been made to improve the performance of these techniques through combination with symbolic music genre classification algorithms, whose use is enabled by pre-processing the audio with an automated transcription algorithm (45). Limited gains in performance are achieved for the combined classification system over the individual models.

High-level features automatically estimated from audio, such as the tempo and meter of a dance music track, have also been used for classification using a

rule based model (24).

Direct comparison of music classification techniques is often challenging as real-world music databases are composed of copyrighted material, preventing distribution of common test databases on which performance may be compared. Further, as stated in section 2.5.3, music similarity information from humans is difficult to collect in quantity, making direct evaluation impossible. Hence, the annual Music Information Retrieval Evaluation eXchange (MIREX) (28) often includes a number of tasks used to compare audio music classification algorithms. For example, MIREX 2007 included genre classification, mood classification, classical composer identification and artist identification tasks. In most cases, each participating algorithm was evaluated on all four tasks. Audio genre classification tasks were run at both MIREX 2005 and 2007.

At MIREX 2005 the best performing classification algorithm was based on an ensemble technique (AdaBoost) (11) that combines classifications from a large number of weak classifiers and aggregates classifications over multiple short segments (13.9 seconds) from each track. Other high performing algorithms included SVM based classifiers (49). Nearly every entry to the MIREX 2007 genre classification task was based on an SVM classifier. Unfortunately, the MIREX 2005 results are not comparable with the 2007 results due to the use of different test collections and the use of an artist-filter in 2007. As stated in section 2.5.1, all genre classification results on small test collections should be artist filtered or over-optimistic estimates of the performance of an algorithm may be made. Further, different algorithms may be affected differently. Hence, the comparative evaluation of genre classifiers at MIREX 2005 may not be wholly trustworthy.

4.2 Classification algorithms and approaches

In this section each classifier that has been applied to the parameterisations of the audio stream detailed in chapter 3 is described.

4.2.1 LDA

A binary linear discriminator forms a projection of multi-dimensional sample data on to a single dimension, along which the data is best separated. A classifier may be formed for a two-class problem by defining a threshold of this projection which will form the decision boundary between the classes. Fisher's criterion is used to find the linear combination of variables that provides the greatest separation between the two classes and is defined as the ratio of between class to within-class scatter (variances) (67). By finding the projection, $\mathbb{W}$, that maximises this ratio, we can discover the direction in feature space along which the two classes are maximally separated. This procedure can be generalised to multi-class problems where a total of $v-1$ projections are used to maximally separate all v classes of data.

For the multi-class problem, the within class scatter matrix ($\vec{\mathbb{S}_W}$) is given by:

$$\mathbb{S}_W(i,j) = \frac{1}{v}\sum_{x=1}^{v}\Sigma_x(i,j), i \in Y, j \in Y, x \in X \tag{4.1}$$

where Y is the set of all feature dimensions, X is the set of all classes, v is the number of classes and $\Sigma_x(i,j)$ represents the covariance of feature i and j for the training data in class x.

The between class scatter matrix ($\vec{\mathbb{S}_B}$), is given by:

$$\mathbb{S}_B(i,j) = \sum_{x=0}^{v}\frac{N_x}{N}(\mu_{x,i}-\mu_i)(\mu_{x,j}-\mu_j), i \in Y, j \in Y, x \in X \tag{4.2}$$

where N is the total number of examples, N_x is the number of examples in class x, μ_i represents the mean of feature i in the training data and $\mu_{x,i}$ represents the mean of feature i for all members of class x in the training data.

The optimal linear projection, $\vec{\mathbb{W}}$, of the data is given by the eigenvectors of the class separation (Fisher's criterion) matrix, $\vec{\mathbb{S}_W}^{-1}\vec{\mathbb{S}_B}$. The most significant $v-1$ eigenvectors of criterion matrix are retained and used to project an observed feature vector, $\vec{u}$, into the new discriminant space as follows:

$$\vec{v} = \vec{u}\cdot\vec{\mathbb{W}} \tag{4.3}$$

The classification of an observed vector, $\vec{u}$, is assigned to the class that minimises the distance of the transformed vector, $\vec{v}$, to the transformed class mean, $\vec{\mu}_x$ as given by:

$$d\left(u,x\right) = |\vec{u}\cdot\vec{\mathbb{W}} - \vec{\mu}_x\cdot\vec{\mathbb{W}}| = |\vec{\mathbb{W}}||\vec{u}\cdot\vec{\mu}_x| \tag{4.4}$$

where $\vec{\mu}_x$ represents the mean vector for class x.

In real situations the true mean and covariance of the data set is not known and is always estimated from the training dataset.

Linear Discriminant Analysis (LDA) may be used as either a linear classifier or as a dimensionally reduction technique for use with other classification algorithms. Further, LDA eliminates correlation's between dimensions of the input feature vector, producing $v-1$ orthogonal dimensions, which can be useful when applying other classification techniques.

4.2.2 Simple Gaussian

A Simple Gaussian classifier assumes that the data points for each class in a classification problem are normally distributed and can therefore be modelled as a mean vector ($\vec{\mu}_x$) and covariance matrix (Σ_x). Again the true mean and covariance of each class is not known and will be estimated from the training dataset. An observed vector of features is classified by comparing it to each class template using the Mahalanobis distance and assigning it the label of the class to which it is closest (67).

The Mahalanobis distance between an observed vector and the normal distribution representing a class is given by:

$$d_M\left(\vec{u},x\right) = \sqrt{(\vec{u}-\vec{\mu}_x)^T\vec{\Sigma_x}^{-1}(\vec{u}-\vec{\mu}_x)} \tag{4.5}$$

The data used to train and apply the simple Gaussian classifier may be preprocessed using the LDA or Principal Component Analysis (PCA) transforms. In the later case, a full covariance matrix need not be estimated as the resulting features will be independent and will therefore have a diagonal covariance matrix.

4.2.3 CART

Breiman et al. (12) introduce a classifier known as a Classification and Regression Tree (CART) which belongs to a class of algorithms known as decision trees, and has formed the basis for many other decision tree algorithms including the well-known C4.5 classifier. Decision tree classifiers recursively partition the training dataset using a threshold of a single variable or weighted combination of variables in order to build a tree structure where the leaf nodes provide significantly better estimates of the classification likelihood of an example than the prior probabilities of the dataset (the probability at the root node). Breiman et al. show that any N-way split using a single variable may be represented as a binary tree, which significantly simplifies the evaluation and selection of splits. Hence, the CART algorithm produces a binary tree.

A CART tree is built by finding the best *split* of the root node, based on the thresholding of any variable or linear combination of variables, into two leaf nodes that have the lowest combined *impurity*. The splitting process is repeated recursively until no split of a node can be found that produces leaf nodes with a lower combined impurity value than the parent node, or the number of examples at the node is less than a specified threshold.

A completely pure node contains only examples of a single class and a maximally impure node would contain an equal number of examples from all classes. There are a number of functions available for estimating the impurity, $\Phi(t)$, of a node, which often lead to trees of very different shapes. One such function is the entropy of the number of examples of each class at the node, given by:

$$\Phi_e(t) = -\sum_{x=1}^{v} p(x|t) log\left(p(x|t)\right) \tag{4.6}$$

where $p(x|t)$ represents the prior probability of class x at node t and v represents the number of classes.

Where a number of classes are present at a node, the entropy-based splitting criterion will tend to divide nodes into two roughly equal groups (42). An alternative to the entropy-based splitting criteria is the Gini index of diversity (12),

which is given by:

$$\Phi_g(t) = 1 - \sum_{j=1}^{v} p(j|t)^2 = \sum_{j \neq k} p(j|t)p(k|t) \tag{4.7}$$

where t is the current node, $p(j|t)$ and $p(i|t)$ are the prior probabilities of the left and right child nodes of node t.

The Gini criterion will choose splits that isolate a single class from the rest of the data. Experiments have shown that the entropy-based splitting criteria tends to slightly outperform the Gini index on our datasets.

The change in impurity, $\Delta\Phi(s,t)$, yielded by a split s of node t into left and right child nodes, whose impurities are $\Phi(t_L)$ and $\Phi(t_R)$ respectively, is given by:

$$\Delta\Phi(s,t) = \Phi(t) - (p_L\Phi(t_L) + p_R\Phi(t_R)) \tag{4.8}$$

where p_L and p_R are the proportion of examples in the child nodes t_L and t_R respectively.

Figure 4.1 provides an example of a CART tree grown to classify data into three classes (Rock, Classical and Electronic). Initially a root node is formed and the probability of any example chosen from the node being in each class calculated. The node is then split according to the procedure outlined above and the feature and decision point providing the best reduction in impurity are chosen and used to create two child nodes as shown. The splitting procedure has been repeated on the right child node to form two further nodes and will be repeated until the resulting nodes are too small to split or a split, that produces nodes with a lower combined impurity than the parent node, can't be found.

In (12) it is shown that defining a rule to stop splitting nodes in the tree, when it is large enough, is less successful than building a maximal tree, which will over-fit the training set, and then pruning the tree back to a more sensible size. A tree which over-fits the training data set is assumed to produce less accurate likelihood profiles for novel feature vectors at a greater computational expense than a smaller tree which does not over-fit the training data. The maximal tree

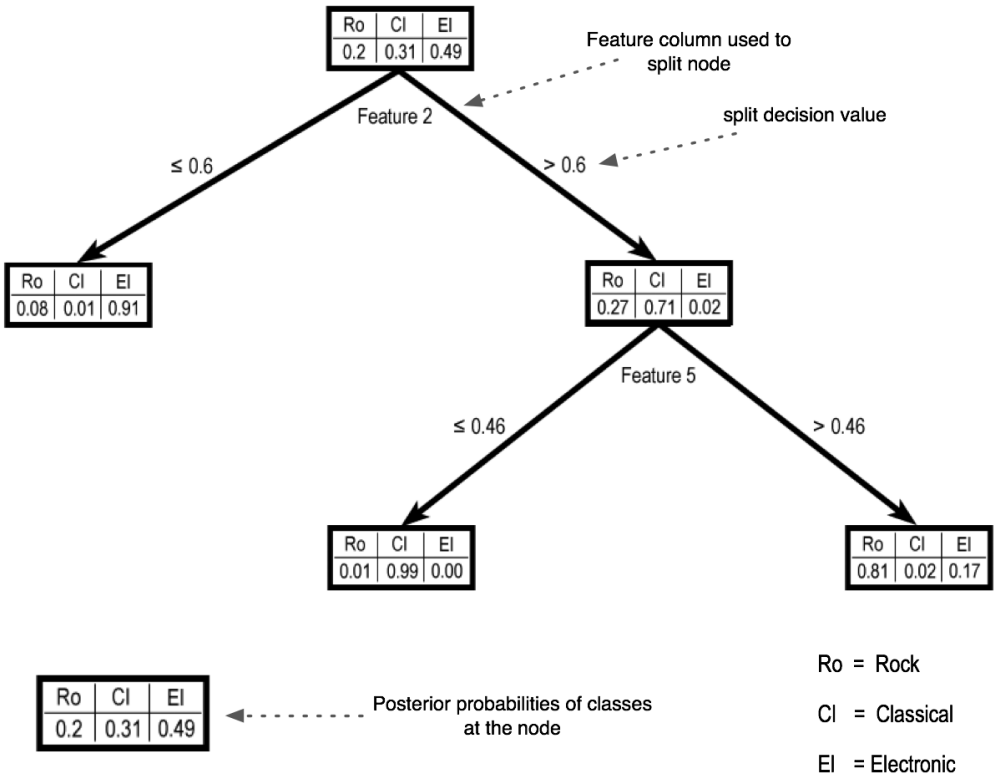


Figure 4.1: An overview of a 3-class CART tree after the first two splits have been selected, demonstrating the refinement of the posterior probabilities of each class as the tree is grown.

is pruned by selecting the weakest non-terminal node in the tree and removing its subtrees. The weakest link in the tree is selected by calculating a function G for each non-terminal node in the tree. G is formulated as follows:

$$G(t) = \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1}, t \notin \tilde{T} \tag{4.9}$$

where $R(t)$ is the re-substitution estimate (the accuracy when classifying the training data present at the node by majority vote) of node t as a leaf node, $R(T_t)$ is the re-substitution estimate of the tree rooted at node t, $\tilde{T}_t$ is the set of all terminal or leaf nodes in the tree T_t and $|\tilde{T}_t|$ is the number of leaf nodes in the tree T_t. The node that produces the lowest value of G in the tree is identified as the weakest link, the whole tree is duplicated and the child nodes of the weakest node are removed. This process is continued until the root node is reached, yielding a finite, nested sequence of pruned trees, ranging from the maximal tree to the tree containing only the root node.

Once a finite, nested sequence of pruned trees has been produced, each tree is evaluated against an independent test sample, drawn from the same distribution as the training data. This allows us to identify trees that over-fit their training data, as they should return a higher misclassification rate than the "right-sized" tree. Initially the tree with the lowest test sample estimate of G is selected. In order to reduce instability in the selection of the right sized tree from a series of trees that may have very similar test sample estimates, the standard error (SE) of the test sample estimate is calculated and the simplest tree (smallest number of leaf nodes) within 1 standard error of the lowest scoring tree is selected as the output tree. The standard error or binomial confidence interval is calculated as follows:

$$SE = \sqrt{\frac{R_{ts}(T)(1 - R_{ts}(T))}{N}} \tag{4.10}$$

where $R_{ts}(T)$ is the independent test sample estimate of the misclassification cost of tree T and N is the number of examples in the test set.

In small tests, the removal from the training set of the additional data required for evaluating the pruned trees can reduce performance. Further, Kuncheva (42) suggests that better likelihood estimates are produced by the maximal tree, particularly when combining results from a number of models or as part of a classifier ensemble.

4.2.4 Handling multiple vectors of features per example

As stated in section 3.2.4, several of our feature extraction systems return sequences of vectors of features per example track. Further, the number of vectors returned is variable and depends on the length of the track or on properties of the audio sequence, such as the number of detected onsets. Hence, the vectors cannot be concatenated to form a single fixed length vector of features.

As most classification algorithms, with the notable exception of Hidden Markov Models, are designed to deal with single, fixed-length vectors of features, a method of handling the multiple and variable number of vectors must be implemented. One approach is to summarise the multiple vectors, for example by estimating a Gaussian distribution (i.e. computing their mean and variance or mean and covariance). This approach is applied to the event-level feature segments as they provide sufficient vectors to estimate a non-singular covariance matrix. An alternative approach is to is to make the (poor) assumption of independence of concurrent feature vectors and to combine their classification likelihood profiles.

Most classification algorithms can be used to estimate a classification likelihood profile (the posterior probabilities of classification). The final classification chosen is usually the class with the highest likelihood, although alternative strategies are available (one such strategy will be explored in section 4.4). *Likelihood profiles* are combined by taking the product of the profiles to be combined. However, a likelihood of zero for a particular class will completely eliminate all other likelihoods for that class. Hence, the likelihood profiles must be smoothed prior to combination. For this we use a generalisation of Laplacian smoothing known as Lidstone's smoothing, which, for classifiers based on counts such as decision trees, is given by:

$$p_{\mathrm{li}}\left(x;\delta\right) = \frac{\left(c\left(x\right)+\delta\right)}{\left(N+\left(\delta * v\right)\right)} \tag{4.11}$$

where p_{li} is the smoothed likelihood of class i given the smoothing parameter δ, $c\left(x\right)$ is the count of examples belonging to class x that the classifier has identified as relevant (for example the count of class x vectors at the leaf node identified for a vector in a decision tree), N is the total number of relevant vectors (i.e. the total number of vectors at the identified leaf node in a decision tree) and v

is the number of classes, given by $v = |X|$, where X is the set of all classes. δ is a small value chosen within in the range $0:1$; it controls the level of smoothing performed (i.e. the amount of the probability space given over to unseen events).

For classifiers that are not based on counts but instead directly return a classification likelihood (for example, distance based classifiers such a single Gaussian classifier with Mahalanobis distance measurements), Lidstone's smoothing can be implemented as follows:

$$p_{\text{li}}(x;\delta,\theta) = \frac{(\theta p(x) + \delta)}{(\theta + (\delta * v))} \tag{4.12}$$

where $p(x)$ is the probability of class x and θ is a large constant used to convert the probability into a count. The level of smoothing performed depends on the combination of δ and θ and hence equation 4.12 can be reformulated as:

$$p_{\text{li}}(x;\gamma) = \frac{(p(x) + \gamma)}{(1 + (\gamma * v))} \tag{4.13}$$

where γ represents $\frac{\delta}{\theta}$.

To avoid underflow in the computation of the product of many vectors with values in the range $0:1$, the log sum is used (equivalent to the product). The final classification likelihoods may be estimated through a normalisation of the resulting vector. Hence the final likelihood for class x ($p_{\text{norm}}(x)$) is given by:

$$p_{\text{norm}}(x) = \frac{Ls(x) - \min\limits_{y \in V}(Ls(y))}{\sum_{y=0}^{v} Ls(y)}, y \in X \tag{4.14}$$

where $Ls(x)$ is given by:

$$Ls(x) = \sum_{i=0}^{I} log(p_{\text{li}}(x,i)) \tag{4.15}$$

where I is the number of vectors for a particular example.

Because the normalisation is a relatively simple operation, it may be performed on the fly, allowing the resulting classification algorithm to return likelihood profiles for a track in real-time as feature vectors become available, gradually refining the likelihood profile. A windowed version may also be implemented by

subtracting the log likelihoods from the accumulated log sum vector (p_{Ls}) as they fall out of the analysis window. Again, this is a simple arithmetic operation, and could be used to implement a system for segmenting a continuous audio stream into different genres.

4.2.5 MVCART

For many classification problems, decision tree algorithms that split nodes on single variables seem appropriate. For example, in a dataset of statistics about people, such as height or weight, producing univariate splits of people into groups based on height alone may be appropriate. However, our feature set is likely to contain a very large number of complex patterns expressed in many closely related dimensions. The heavily modified version of the CART algorithm, detailed in West and Cox (69) and West and Lamere (71) and known as Multivariate CART (MVCART), is intended to address this concern by implementing multivariate splits of the data at each node.

A Multivariate Classification and Regression Tree is grown by forming a root node containing the entire training dataset and recursively splitting according to the normal CART training algorithm (including the use of an entropy-based split evaluation metric). However, in order to split a node, Fisher's criterion multiclass linear discriminant analysis is used to produce a linear transformation of the data into a space providing maximal separation between the classes. This produces a new $N' - 1$ dimensional space, where N' represents the number of classes that had examples at this node. Should the transform training fail, i.e. if the covariance matrix is singular and cannot be inverted, the algorithm will 'fail over' to a diagonal covariance LDA transform, or to the raw feature values. Singular covariance matrices may be the result of too few training vectors at the node or an excessively pure node. As this process is repeated recursively, projections on different branches of the tree will evolve very differently in order to separate different subsets of the music.

In order to implement a binary split, all the combinations of the available classes of data into two groups (a positive and negative class) are enumerated,

without allowing repetition, permutation or reflection. For example, in a dataset consisting of examples in three classes $\{A, B, C\}$ the splits $A|BC$, $AB|C$ and $AC|B$ are all evaluated. A pair of Gaussian distributions are then estimated, representing the positive and negative classes, to try and duplicate each split on novel data. Finally, the training data is passed through the resulting classifier and divided over the two potential child nodes, as it is likely that the classifier will imperfectly reproduce the division of the data into the child nodes, and the split yielding the maximum reduction in the entropy of the classes of data at the node (i.e. the combination that produces the most 'pure' pair of leaf nodes) is selected as the final split.

A diagrammatic overview of a simple 3-class MVCART tree is given in figure 4.2, demonstrating how the classification likelihoods have been refined by each partitioning of the dataset. In figure 4.2 a root node has been formed with roughly equal numbers of example vectors from each class, yielding approximately the same classification likelihood For each class. The root node has been split using the process described above (i.e. an LDA has been trained, pairs of Gaussian distributions fitted to all the possible divisions of the classes into two groups and the pair yielding the greatest reduction in the entropy of class membership in the child nodes selected). Hence, the root node is a non-terminal (non-leaf) node and is shown with two ellipses, representing the binary Gaussian classifier trained in LDA transform feature space. The splitting process is repeated recursively. In this example the left child node could not be split effectively, however, the right child node has been split yielding a further pair of child nodes. Terminal or Leaf nodes have not or could not be split and, in figure 4.2, they have been left blank. Figure 4.2 demonstrates the splitting process, however, a real MVCART tree is likely to be trained on significantly more classes of data and would be composed of many more nodes than in this example.

A maximal tree is trained by recursively splitting until a node is too small to be split or is pure. A sensible threshold for number of examples that may be used to create effective splits is roughly twice the number feature dimensions, as this will usually allow for the estimation of a non-singular covariance matrix. In most experiments the tree produced by the MVCART algorithm does not benefit from pruning, as this operation often cuts the tree immediately back to the root.

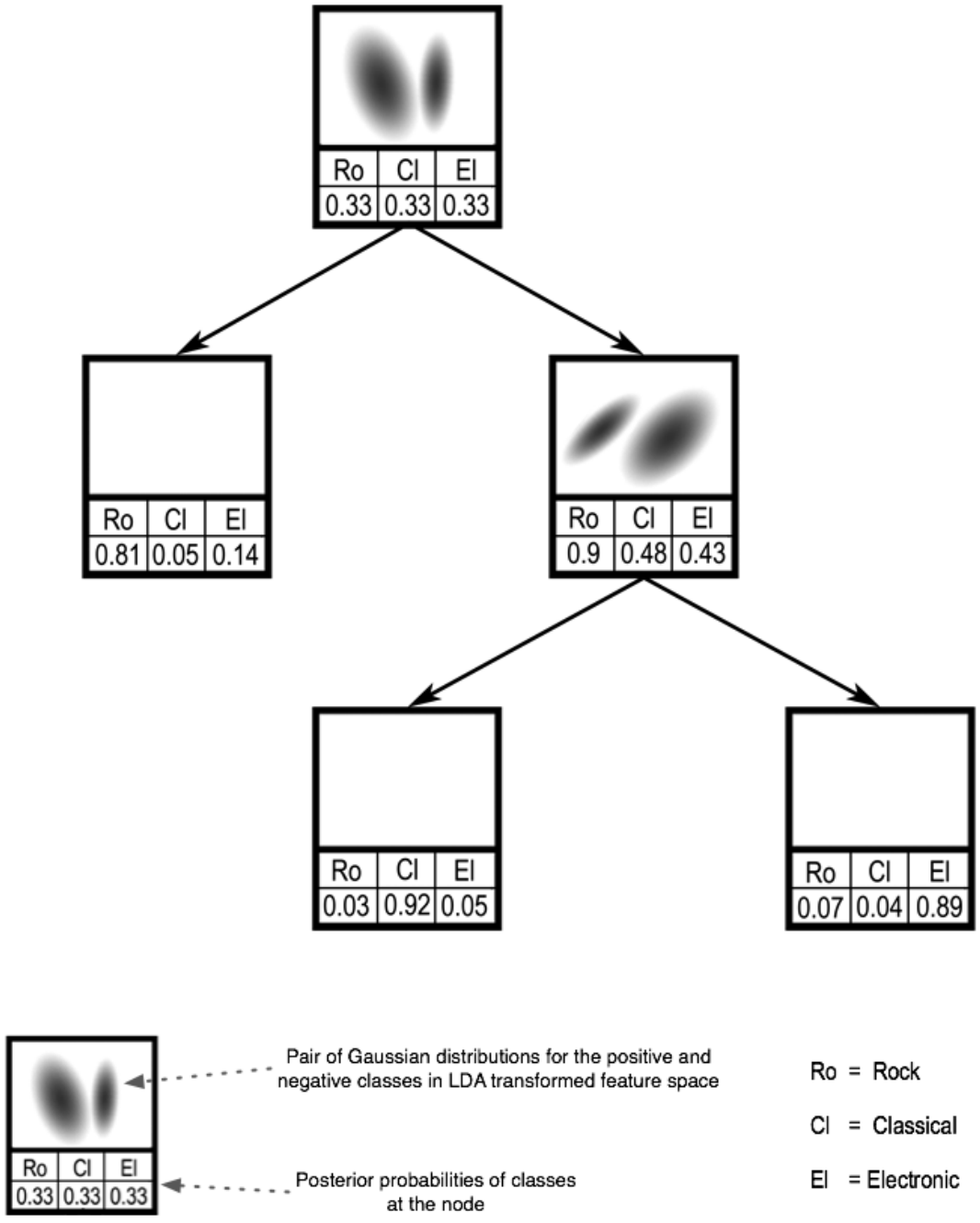


Figure 4.2: An overview of a 3-class MVCART tree after the first two splits have been selected, demonstrating the refinement of the posterior probabilities of each class as the tree is grown. Note that nodes which have been split (non-leaf nodes) are shown with ellipses representing the Gaussian distributions trained in the LDA-projected feature space. Leaf nodes have not been split and are blank.

Hence, pruning is not attempted.

The computational complexity (runtime) of building an MVCART tree is higher than that of a conventional CART tree, but not significantly so. In most tests the difference in runtime for the training process is not noticeable and often lower than that of some of the linear classifiers. Further, the application of the MVCART tree at runtime is also very fast as the complexity of applying a decision tree scales logarithmically with increases in the number of its leaf nodes (i.e. the trees are much 'wider' than they are 'high', so descending through the decision tree is relatively fast).

However, the number of splits attempted by MVCART is not related to the number of feature dimensions and vectors as it is in the classic CART algorithm. It is instead controlled by the number of classes in the dataset. Given that repetition, permutations and reflections of the grouping of the classes into a pair of meta-classes is not allowed, the number of splits that must be evaluated at each node is $\sum_{k=0}^{v-2} 2^k$ where v is the number of classes. Hence, for large v this can rapidly grow to be an unmanageable number of Gaussian distributions to compute. To reduce this complexity, a hierarchical genre taxonomy may be used to 'back-off' to a smaller number of meta-classes for the initial m splits. Given m rounds of significantly reduced split evaluation complexity, one or more classes may be rapidly eliminated entirely from a particular branch of the tree or reduced to a negligible proportion of the population that may be ignored when enumerating the possible splits. Further, the significantly reduced node population will ensure that splits maybe much more rapidly evaluated.

If a multi-level hierarchical genre taxonomy is available, the'back-off' procedure may be performed in stages. This enables the MVCART training algorithm to handle combinations of relatively large datasets and wide genre taxonomies. This procedure is closely related to hierarchical classification strategies used by some authors (e.g. the system described by Tzanetakis and Cook (65)), which divide the feature vectors into higher-level meta-classes and then apply classifiers designed to classify the example only into the child classes of that meta-class.

The MVCART tree is applied to feature sets containing multiple vectors of features per track, by making a making an assumption of independence of the vectors and combining their likelihood profiles, as described in section 4.2.4. A diagrammatic overview of this process, for event-level feature vectors, is given in figure 4.3.

4.2.5.1 Correcting probabilities generated by variable length sequences

Because our audio pre-processing front-end provides us with a *variable* number of vectors per example and not a single feature vector per example or even fixed number of vectors per example, we need to normalise the likelihood of classification for each class by the total number of vectors for that class in the training set. Particularly with event-level feature processing, different genres of music may produce significantly different numbers of vectors. For example fast dance music will contain many more audio events of shorter duration than tracks from other genres and hence will produce many more vectors. Hence, to avoid outputting over-optimistic likelihoods for the best represented classes, normalisation is performed as follows:

$$p'(x|t) = \frac{\frac{1}{v}p(x|t)}{Pr(x)} = \frac{\frac{1}{v}\frac{Nv_{x,t}}{Nv_t}}{\frac{Nv_x}{Nv}} \tag{4.16}$$

where $p(x|t)$ represents the likelihood of class x given node t, $Pr(x)$ represents the prior probability of example vectors for class x in the dataset, v is the number of classes, Nv is the total number of example *vectors* (Note: this will be significantly greater than the number of *tracks*), Nv_x is the total number of vectors for class x and $Nv_{x|t}$ is the total number of vectors for class x at node t. The $\frac{1}{v}$ term assumes an equal prior for all classes that will be substituted. Alternatively, this term could be replaced with the actual prior of the whole dataset, $\frac{N_x}{N}$.

As the normalisation is performed on the smoothed and accumulated log-likelihood profile equation 4.15 may be reformulated as:

$$Ls(x) = \frac{1}{v} + \left(\sum_{i=0}^{I} log\left(p_{\text{li}}(x|t,i)\right) - log\left(Pr(x)\right)\right) \tag{4.17}$$

to efficiently implement this normalisation.

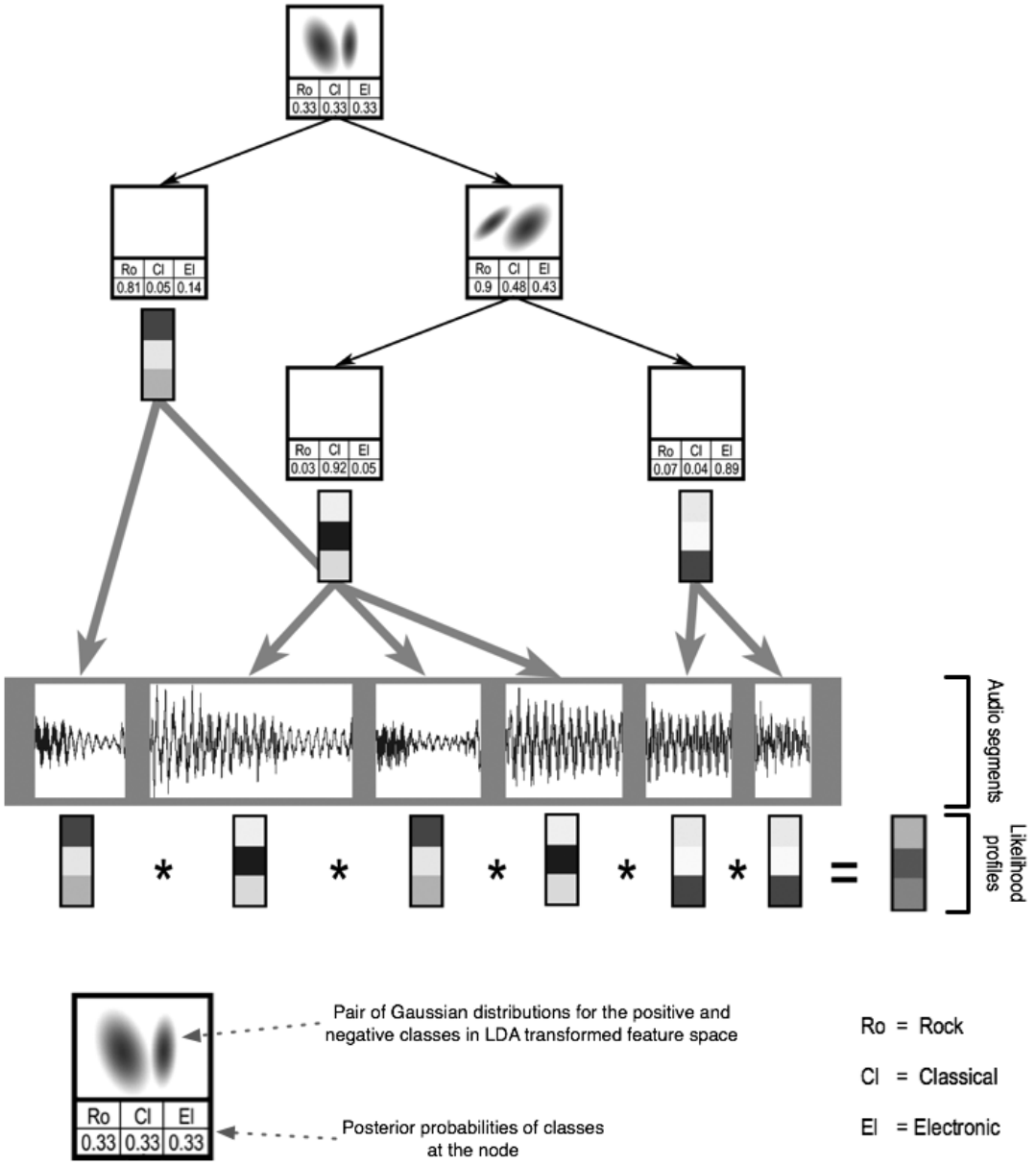


Figure 4.3: Application of simplified MVCART tree to event-level feature vectors of a track and the combination of likelihood profiles to produce a final classification profile.

Despite the fact that the MVCART tree is attempting to partition the feature space into smaller regions, reducing the effect of the prior probability based on the number of event vectors, the performance of both MVCART and conventional CART trees at classification is improved by the normalisation of the likelihood profiles produced by the model.

4.2.5.2 An alternative method of handling multiple feature vectors

As stated in section 4.2.4, an independence assumption is used to combine the classification likelihoods of individual vectors to generate a likelihood profile for the whole track, and this assumption is a poor one for music. Through the use of text processing techniques, this may be avoided for Decision Tree classifiers such as the CART or MVCART trees.

The Decision Tree is viewed as a hierachical taxonomy of the event-level audio segments in the training database, where each taxon is defined by its explicit differences and implicit similarities to its parent and sibling taxons. The leaf nodes of this taxonomy can be used to label a sequence of input frames or segments and provide a 'text-like' transcription of the music. It should be stressed that such 'text-like' transcriptions are in no way intended to correspond to the transcription of music in any established notation and a specific taxonomy will only be produced by a specific model and training set. An example of this process is shown in figure 4.4. This transcription can be used to index, classify and search music using standard retrieval techniques.

To demonstrate the utility of these transcriptions, we have implemented a basic Vector model text classifier, where the transcription is converted into a fixed size set of term weights, which are subsequently used to train a classification algorithm. In a conventional text-based information retrieval system the *terms* for each *document* are the individual words. In our system, the "documents" are the audio tracks and the leaf node in the Decision tree for each event-level vector is considered a "term". The weight for each term t_i can be produced by simple term frequency (TF) (5), as given by:

$$t_i = tf_i = \frac{n_i}{\sum_k n_k} \tag{4.18}$$

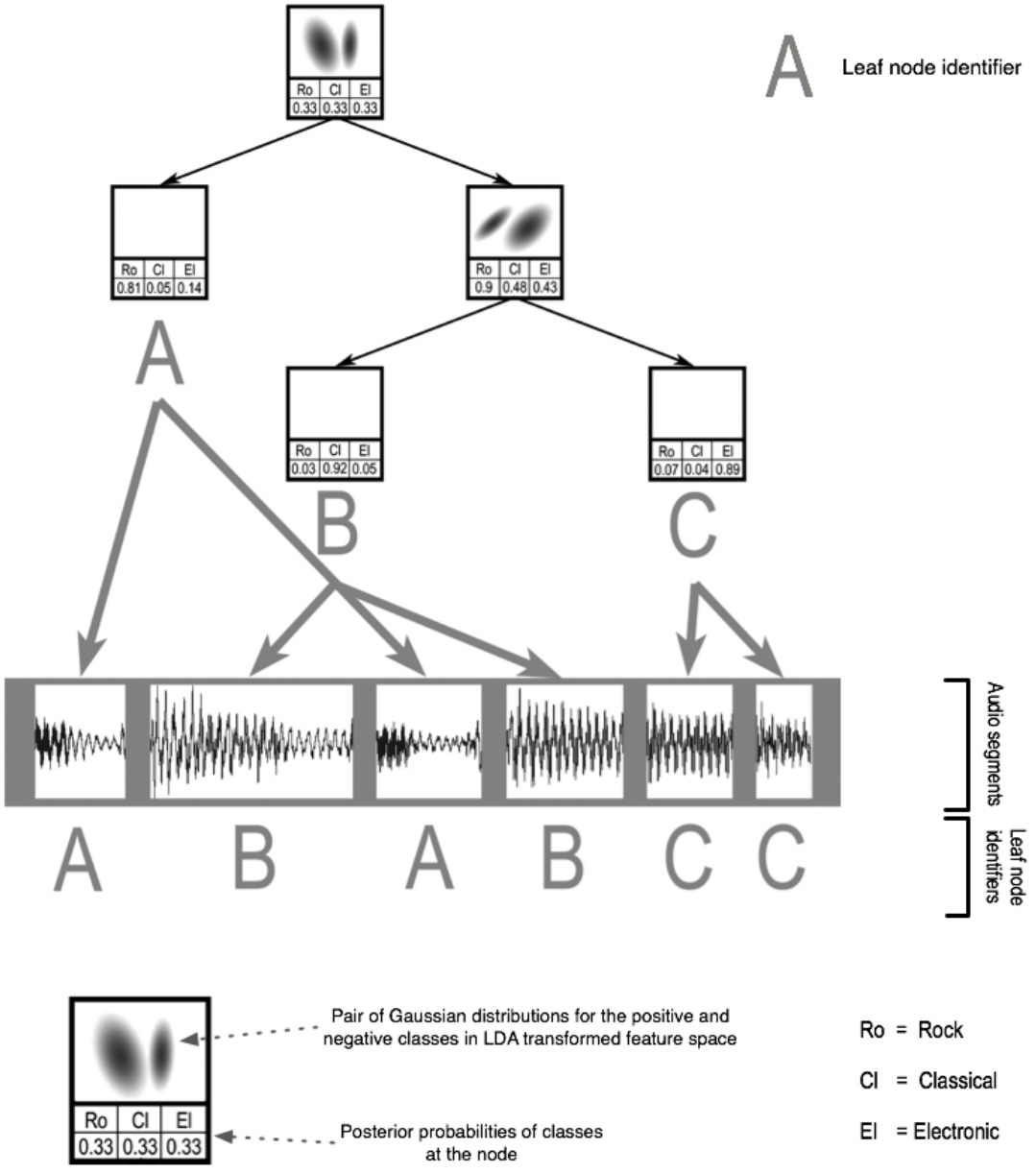


Figure 4.4: Producing a 'text-like' transcription of a track using the MVCART model.

where n_i is the number of occurences of term i. Alternatively, term frequency - inverse document frequency (TF/IDF) (5), can be used:

$$idf = \log \frac{\mid D \mid}{\mid (d_i \supset t_i) \mid} \tag{4.19}$$

$$t_i = tfidf_i = tf_i \cdot idf \tag{4.20}$$

where $\mid D \mid$ is the number of documents in the collection and $\mid (d_i \supset t_i) \mid$ is the number of documents in the database containing term t_i.

The IDF term significantly reduces the weight of commonly occurring terms in the dataset. This is a significant advantage when trying to retrieve text documents as it prevents very common terms such as 'the' or 'and' from having a strong effect on the estimated similarity of a document to a query. However, this property is likely to be of less importance in music, owing to the fact that the genre of a music track (which is analogous to the topic or subject of a text document) will have a strong influence on the most commonly occurring sounds (instrumentation) of the track. Whereas, in text corpora the topic or subject is likely to have little influence on the most commonly occurring terms.

Once the weights vector for each document has been extracted, a classifier may be trained to learn patterns found in the term weight profiles that correspond to each class in the dataset. Both Support Vector machine classifiers, as described in section 4.2.6, and a simple k-nearest neighbour classifier 4.2.7 are trained on these new feature vectors and used to classify the tracks. Training the SVM classifiers on both TF vectors generated by the resubstitution of the data used to build the Decision Tree and an independent sample of the training data have been explored. However, it should be noted that, when using an independent sample of the training data in small data sets, the high proportion of the training data that cannot be used to train the Decision Tree classifier tends to lead to lower performance estimates than if a greater amount of training data were available. Further, the use of artist filtering will often lead to pathological cases where tracks from only a single artist or very few artists form the Decision Tree or text-classifier training sets for a particular class. As music recordings

are copyrighted material and cannot be freely exchanged, this is often the case. Hence, only results for the resubstitution of the MVCART training data are reported here.

The SVM models can be trained on the TF vector profiles very rapidly in comparison to training classifiers on the full audio features. Further, the sparse TF profiles also require significantly less memory to represent than most audio features. Hence, these the TF vectors might be used to train client-side classification models as part of an MIR platform, potentially extending the classifier to classes of audio and training data that were not present for the training of original transcription model. This procedure might also be used to map an existing collection into a new genre metadata taxonomy or to establish links between classes in different taxonomies.

4.2.6 Support Vector Machines

Support Vector Machines (SVMs) are a type of linear classifier, similar to the Linear Discriminant classifier in that both (in binary classification problems) attempt to construct a hyperplane that separates the data into two classes (8). However, SVMs attempt to maximise the geometric margin between the classes (attempting to find the hyperplane along which the data-points of each class are maximally separated from the data-points of another class), whereas Fisher's criterion Linear Discriminant classifiers attempt to maximise the ratio of between class variance to within class variance (attempting to find the plane along which the data points of the classes cluster tightly and are as far from the other classes as possible). Hence, SVMs are known as maximum margin classifiers. Intuitively, maximising the margin between the classes should lead to a robust classifier as small perturbations of data points should not lead to classification errors.

As many data sets are not linearly separable, a hyperplane that cleanly separates the classes of data cannot always be found. Hence, the influence of any single data-point on the solution must be limited and the hyperplane that best separates the data-points of each class, while minimising error must be found. A

suitable formulation for solving this optimisation problem using Lagrange multipliers is given in (19).

The linear SVM classifier may be extended to non-linear classification through the use of the 'kernel trick'. A kernel function is substituted for a dot product in the optimisation problem used to select the maximum-margin hyperplane. This effectively maps the features into a higher dimensional space as part of the maximum-margin hyperplane selection. If the kernel function used is non-linear, then the maximum-margin hyperplane fitted in the transformed feature space will be non-linear in the raw feature space. Hence, through the selection of a suitable kernel function, non-linearly separable classification problems may be solved.

Two kernel functions are utilised in this work:

Polynomial degree d

$$K(u, v) = (u \cdot v + 1)^d \tag{4.21}$$

Radial Basis Function

$$K(u, v) = exp\left(-\frac{||u - v||^2}{2\sigma}\right) \tag{4.22}$$

where u and v represent data vectors, d represents the degree of the polynomial and σ represents the width of the Gaussian.

The implementation of the Support Vector Machine algorithm used in this work is part of the Weka machine learning toolkit (74) and is based on John C. Platt's Sequential Minimal Optimisation (SMO) algorithm (59) and improvements given in (40).

4.2.7 Instance-based K-Nearest Neighbour (IBK)

Instance-based K-Nearest Neighbour (IBK) classifiers belong to a class of algorithms known as Lazy classifiers. Lazy classifiers do not attempt to form a model of data and how it is divided into classes prior to classification (as an SVM, LDA

or Decision Tree does). Instead, virtually the entire classification procedure is executed at runtime. Hence, the training of a Lazy model is often as simple as passing a reference to the training dataset, although some implementations allow for the estimation or optimisation of univariate parameters of a distance function used at runtime.

The IBK classifier computes posterior probability profiles for examples by finding the k-nearest neighbours of a track, t, to be classified within the labelled training dataset. The distance, $\mu_(A, B)$, between two tracks, A and B, is estimated by distance function d calculated between the feature vectors, $\vec{a}$ and $\vec{b}$, used to represent tracks A and B:

$$\mu_(A, B) = d\left(\vec{a}, \vec{b}\right) \tag{4.23}$$

Once the set of the k closest tracks in the training dataset have been found, a posterior probability of each class x can be estimated as the sum of the inverse distances of tracks in the set belonging to class x divided by the sum of the inverse distances of all k tracks in the set as defined in equation 4.24:

$$p\left(x|A\right) = \frac{\sum_{y=1}^{k} \frac{\delta_{y,x}}{\mu(A,y)}}{\sum_{y=1}^{k} \frac{1}{\mu(A,y)}} \tag{4.24}$$

where $\delta_{y,x}$ is a Kronecker delta function which indicates whether example y is a member of class x:

$$\delta_y = \begin{cases} 1, \text{if} & y \in x \\ 0, \text{if} & y \notin x \end{cases} \tag{4.25}$$

In the experiments reported here the distance function, μ, is the Euclidean distance:

$$dist_{euc}(A, B) = \sqrt{\sum_{j=1}^{F} (f_j(A) - f_j(B))^2} \tag{4.26}$$

where F is the total number of features used to represent tracks, $f_j(A)$ represents the j-th feature from the feature vector $\vec{a}$, which represents track A.

The value of parameter k, the size of the neighbourhood used to classify each track, is set by 5-fold cross-validated optimisation on the training dataset, up to a maximum value of 12.

4.3 Evaluation

In this section the performance of the classification algorithms detailed in section 4.2 is evaluated against each segmentation of the feature stream described in section 3.2.3. The full set of components used to create the candidate classification systems is given in section 4.3.2. Classification accuracy scores are estimated using 5-fold cross-validation and the variance in each result is measured and are reported in section 4.3.4.

To avoid inflated estimates of the classification accuracy of each system, each fold in cross-validation has been artist-filtered (55) ensuring that an artist only appears in the test or training set, not split across them.

4.3.1 Test collections

Each system evaluated here was developed on the collection of Creative Commons music available from http://www.magnatune.com. However, as the algorithms were developed and tuned on this dataset, subsequent evaluation might over-estimate the performance of the techniques on other collections. Hence, an independent dataset, supplied by the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) at the University of Illinois (UIUC), has been used to ensure that all performance estimates are unbiased.

This test collection is composed of 8219 tracks divided into 11 genres:

Each track is represented by a 30 second audio clip sampled from the centre of each audio file.

4.3.2 Candidate systems

In the following section, results are reported for systems based on combinations of the feature sets, segmentations and classification algorithms described in this chapter.

Table 4.1: Dataset composition

Genre	Number of tracks
Baroque classical	750 tracks
Blues	750 tracks
Classical classical	750 tracks
Country	750 tracks
Electronica & Dance	720 tracks
Hard Rock & Metal	750 tracks
Jazz	750 tracks
Rap & Hip-Hop	749 tracks
Rock	750 tracks
Rock & Roll	750 tracks
Romantic classical	750 tracks
Total	8219 tracks

4.3.2.1 Feature sets

The feature sets considered in these experiments include:

Rhythm Cepstral Coefficients (RCCs) Computed with an initially linear then logarithmically scaled filterbank, with transition frequency of 0, 2, 3 or 4 Hz. These are referred to as LL0, LL2, LL3 and LL4 respectively.

Timbral feature set Composed of MFCC and the Spectral Distribution Descriptors described in section 3.2.2.

4.3.2.2 Timbral feature stream segmentations

The segmentations of the timbral features considered include:

whole files Single Gaussian (mean and covariance) summaries of the feature frames.

50% overlapped, 10 second segments Single Gaussian (mean and covariance) summaries of the feature frames corresponding to each segment.

event-based segmentation Mean vector summaries of the feature frames corresponding to each segment.

event-based segmentation, whole files Mean vector summaries of the feature frames corresponding to each segment with a single Gaussian (mean and covariance) summary of the event feature frames.

4.3.2.3 Rhythm feature stream segmentations

The segmentations of the RCC features considered include:

50% overlapped, 9 second segments No summary required.

50% overlapped, 9 second segments, whole files Mean vector summaries of the feature frames.

4.3.2.4 Classification algorithms

The classification algorithms applied to these features include:

Simple Gaussian

Support Vector Machine (SMO) - Polynomial kernel degree 1 (SMO Poly 1). Higher-order polynomial kernels have been experimented with but have never produced higher performance.

Support Vector Machine (SMO) - Radial basis function (SMO RBF)

Instance-based K-Nearest Neighbour (IBK) K is set by leave one out cross validation on the training dataset, with the contribution by each neighbour weighted by the inverse of their distance.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA)/Simple Gaussian Simple Gaussian classifier trained in LDA transform space (LDA Gaussian).

Linear Discriminant Analysis (LDA)/IBK Instance-based K-Nearest Neighbour (IBK) classifier trained in LDA transform space (LDA IBK).

Classification and Regression Trees (CART) Stopping criteria based on the percentage of training dataset vectors available at the node - set at 0.05%.

J48 Decision Tree (J48) - Stopping criteria based on a minimum of 50 examples at each node to be split, with reduced error pruning and sub-tree raising.

Multivariate Classification and Regression Trees (MVCART) Linear discriminant analysis and a pair of full-covariance simple Gaussian models used to split each node. Stopping criteria based on the percentage of training dataset vectors available at the node - set at 0.1%, 0.05% and 0.01%.

4.3.3 Statistical significance testing

Although the accuracy scores can be used to rank the different classification systems, they do not tell us if any apparent differences in performance are statistically significant or if the *behaviour* of the systems was different (given the same or similar performance, two models may actually have different sets of correctly and incorrectly classified tracks). Hence, we must evaluate the statistical significance of differences in the performance of the systems and their behaviour. The selection of the correct statistical test is essential for the validity of the conclusions we draw.

4.3.3.1 The Friedman test

Friedman's ANOVA is a test that compares systems or 'treatments' over a number of 'blocks', e.g. datasets, folds in cross validation or queries for retrieval, and is non-parametric (i.e., does not assume any parametric distribution of the underlying data) (9). The result data is viewed as a rectangular matrix, with the different treatments or classification algorithms represented as the columns and the queries or cross-validaiton folds forming the rows. So called 'row effects', or variance introduced into the results by different splits of the dataset into train and test sets, are handled by replacing the actual scores with their ranks amongst the systems compared (the columns). Friedman's ANOVA assumes that the variance introduced by individual queries, human evaluators or splits of a dataset

(the 'blocks') will effect the scaling and distribution of the evaluation data, but not the ordering of the results themselves.

In order to compute Friedman's ANOVA we compute a performance score, x_{ij}, for each system, j over each test (query or cross-validation fold), i. Each score is then replaced with its rank within it s 'block' or test, r_{ij}. The test statistic, Q, may then be calculated as follows:

$$\bar{r}_{\cdot j} = \frac{1}{n}\sum_{i=1}^{n} r_{ij} \tag{4.27}$$

$$\bar{r} = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij} \tag{4.28}$$

$$SS_t = n\sum_{j=1}^{k}(\bar{r}_{\cdot j} - \bar{r})^2 \tag{4.29}$$

$$SS_e = \frac{1}{n(k-1)}\sum_{i=1}^{n}\sum_{j=1}^{k}(r_{ij} - \bar{r})^2 \tag{4.30}$$

$$Q = \frac{SS_t}{SS_e} \tag{4.31}$$

Where n represents the number of 'blocks' or tests and k represents the number of 'treatments' or systems. The probability distribution of Q is either approximated as the Chi-square distribution or obtained from tables.

Conventionally, Friedman's ANOVA determines whether the performance of any one of the systems is statistically significant from the others. However, through the use of an appropriate multi-comparison method, Friedman's ANOVA also allows for the statistically valid pair-wise comparison of each of the system results for statistically significant differences. This process is superior to the commonly (mis)used multiple Student's t-tests (17). The problem with the typical application of multiple t-tests or other similar statistical tests for post-hoc pair-wise comparisons lies in the fact that the probability of incorrectly rejecting the null hypothesis of no difference in error rates (i.e., $H_0 : \mu(x) = \mu(y)$ at some confidence α (e.g., $\alpha = 0.05$) increases in direct proportion with the number of

pair-wise comparison conducted. For a complete (systematic) pair-wise comparison analysis, there will be $x = \frac{c(c-1)}{2}$ comparisons made where c is the number of items in the set of interest. In our case, the items of interest are the final accuracy scores of each classification algorithm on each fold of the dataset *or* on each genre in the test. When doing such multiple comparisons, the experiment-wide α level (α_{ew}) is defined as $\alpha_{ew} = 1(1-\alpha)^x$ where x is the number of comparisons made. Hence, adapting Tague-Sutcliffe and Blustein's example (63) to an experimental situation where there are 12 systems to be compared, there are $\frac{12(11)}{2} = 66$ possible pair-wise comparisons and so, if each of these were tested at the $\alpha = 0.05$ level, the probability of incorrectly rejecting H_0 at least once would be $1(.95)^{66} = 0.97$, i.e., almost a certainty.

Analysts have many options for compensating for the multicomparison problem including the Bonferroni, Scheffe, and Tukey-Kramer techniques For the comparison of results from the human evaluations of the MIREX 2006(27)(28) and 2007 audio and symbolic search and retrieval tasks, IMIRSEL chose the Tukey-Kramer method, as it is not as conservative as the Bonferroni or Scheffe techniques, and it is one with which IMIRSEL has had some prior experience. We have chosen to duplicate that approach in these experiments for the comparison of genre classification results over multiple folds of an experiment. Because of its non-parametric nature, we believe this approach could be extended to compare classification systems over multiple datasets and taxonomies. The α value used to reject the null hypothesis (H_0) in these comparisons is 0.05.

The results of Friedman tests with multiple comparisons are best interpreted as column rank plots. These plots demonstrate the mean column rank achieved by each system on each query and the confidence interval around that mean. Note that the highest rank a system can achieve is equal to the number of systems compared, i.e. the best rank is the highest rather than the lowest value. Two systems are considered to have a statistically significant difference in performance (at the chosen α level) if their confidence intervals do not overlap, as demonstrated in figure 4.5. Each column rank plot shown has the confidence interval of the best performing algorithm highlighted in blue, while any systems that have significantly lower performance are highlighted in red.

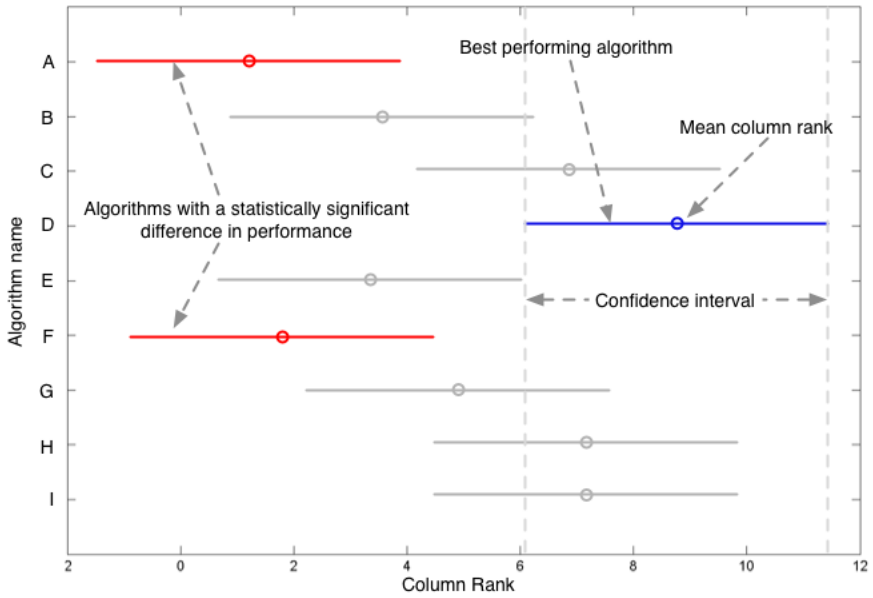


Figure 4.5: An example column rank plot of a number of algorithms. The blue bar represents the highest performing algorithm (as it has the highest mean column rank). Algorithms which have a significantly lower performance compared to the best performing algorithm are shown in red.

4.3.3.2 The McNemar test

McNemar's test (31) is another method used to conduct the pair-wise comparison of a set of systems in order to determine whether any apparent difference in error-rates between any two algorithms, A_1 & A_2, tested on the same dataset is statistically significant. McNemar's test is performed by summarising the classification results of each pair of algorithms tested in the form of a two by two matrix containing the number of examples correctly classified by both algorithms(N_{00}), neither algorithm (N_{11}) and those only classified correctly by one of the algorithms (N_{10} & N_{01}) as shown in figure 4.2.

As there is no information about the relative performance of the two algorithms when they agree, these last two values are the only ones used in McNemar's test. Let H_0 be the hypothesis that the underlying error-rates are the same. Then under H_0 an error is as likely to be made by A_1 as A_2, and the distribu-

Table 4.2: Example summary matrix for the comparison of two algorithms.

Algorithm A / Algorithm B	Correct	Incorrect
Correct	N_{00}	N_{10}
Incorrect	N_{10}	N_{11}

tion of N_{10} & N_{01} is the distribution obtained when tossing a fair coin and, for instance, tails (N_{10}) is obtained. This is a binomial distribution and the P-values are easily obtained from tables.

McNemar's test not only indicates significant differences in the *performance* of the algorithms, but examination of the 2x2 matrix that is used to provide the data for test reveals differences in the *behaviour* of the algorithms. Research into classifier ensembles (systems for combining multiple classification algorithms into a single classifier with higher performance) shows that systems with significantly different error-rates may be combined to form ensembles with higher performance, whereas the combination of highly correlated systems is unlikely to produce a system with higher performance (42). Hence, McNemar's testing can be an important tool in selecting uncorrelated systems where combination might lead to improved performance.

Differences in error-rates are reported as significantly different at an α value of either 0.05 or 0.01. Each comparison with a non-zero P-value is indicated and significant differences are marked with T and non-significant differences with an F.

4.3.4 Classification results

Tables 4.3 and 4.4 provide abridged versions of the classification results and standard deviations achieved by each combination of features, segmentation and classification algorithm over both the Rhythm and Timbre feature sets. The full version of the results table (A.1) is given in appendix A. The best results for each Feature set and segmentation combination are highlighted in **bold** type.

4.3.4.1 Timbral classification results

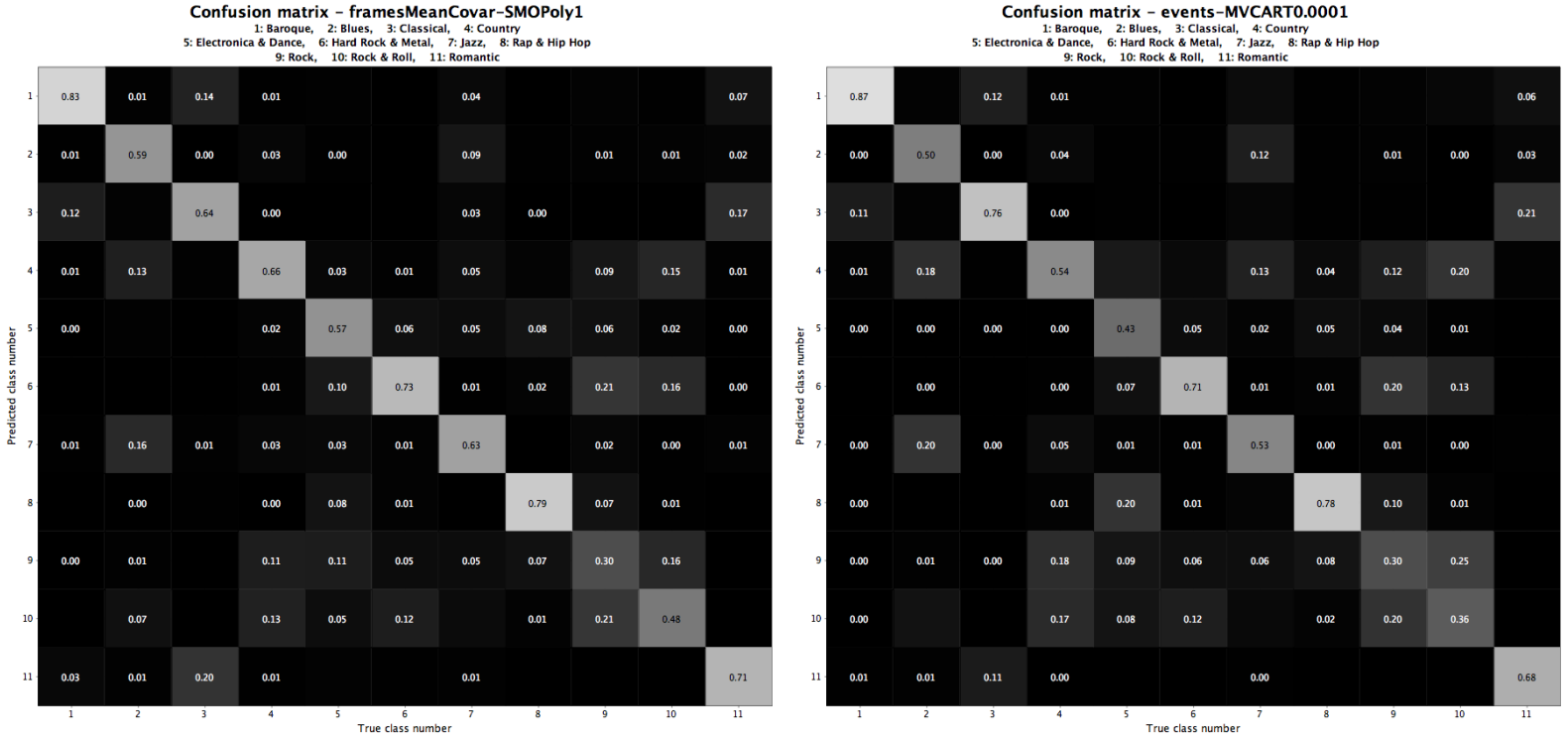
Table 4.3: Classification accuracy - Timbral features

Feature set	Segmentation/summary	Classifier	Acc	St.dev.
Timbre	whole file	CART	44.55	0.61
	mean and covariance	IBK	55.26	0.81
		LDA	60.53	0.76
		LDA IBK	62.56	0.59
		MVCART0.001	54.03	0.45
		SMO Poly1	**63.04**	**0.45**
Timbre	10 second windows	J48	49.58	0.75
		MVCART0.0001	54.38	0.52
		SMO Poly1	**58.62**	**0.66**
Timbre	Event-based segmentation	IBk	54.38	0.60
	whole file	J48	43.85	0.16
	mean and covariance	LDA	58.75	0.87
		SMO Poly1	**63.51**	**0.63**
Timbre	Event-based segmentation	CART	54.07	0.28
	event mean vectors	J48	55.81	0.30
		SMOPoly1	**failed**	–
		LDA	**failed**	–
		MVCART0.0001	**58.94**	**0.69**
Timbre	Event-based segmentation	MVCART0.001 SMOPoly1	58.12	0.57
	event mean vectors	**MVCART0.001 SMOPoly2**	**59.12**	**0.51**
	TF	MVCART0.0005 SMOPoly1	57.04	0.57
		MVCART0.0005 SMOPoly2	58.95	0.55
Timbre	Event-based segmentation	MVCART0.001 SMOPoly1	58.33	0.55
	event mean vectors	MVCART0.001 SMOPoly2	59.19	0.47
	TF/IDF	MVCART0.0005 SMOPoly1	57.29	0.59
		MVCART0.0005 SMOPoly2	**59.25**	**0.47**

In general the confusion matrices for the timbral classifiers show a very consistent pattern of confusion. The majority of confusions that occur with any of the timbral classifiers are localised, e.g. Blues vs. Jazz, Rock vs. Rock & Roll, Country or Hard Rock & Metal vs. Rock or Rock & Roll and amongst the types of classical music (Classical, Romantic and Baroque). Such confusions are easy to understand as the types of music confused are all based on similar principles and instrumentation – yielding similar polyphonic timbres. These patterns are evident across each different segmentation of the timbral features and in combination with any of the classifiers.

Further, the absence of other significant confusions is encouraging, meaning that the mistakes made by all systems are relatively "sensible", i.e. human-like. Extreme confusions (e.g. between Baroque classical and Rap & Hip Hop) would be hard to explain, and might eliminate the usefulness of any otherwise well performing system.

Figure 4.6 provides an example of two of the confusion matrices and demonstrates the similarity in the patterns of confusion produced by the different algorithms. The algorithms whose patterns of confusion are shown, a Support Vector Machine (SMO) with a first-order polynomial kernel trained on the mean and covariance of frame-based features and an MVCART classifier trained on sequences of mean vectors for event-based segments of the audio, are two of the most different timbral classifiers, yet they produce a very similar pattern of confusion.



(a) SMO Poly1 Frames Mean and Covariance

(b) MVCART Event based features

Figure 4.6: Comparative plots of confusion matrices produced by two different genre classifiers demonstrating very similar patterns of confusion

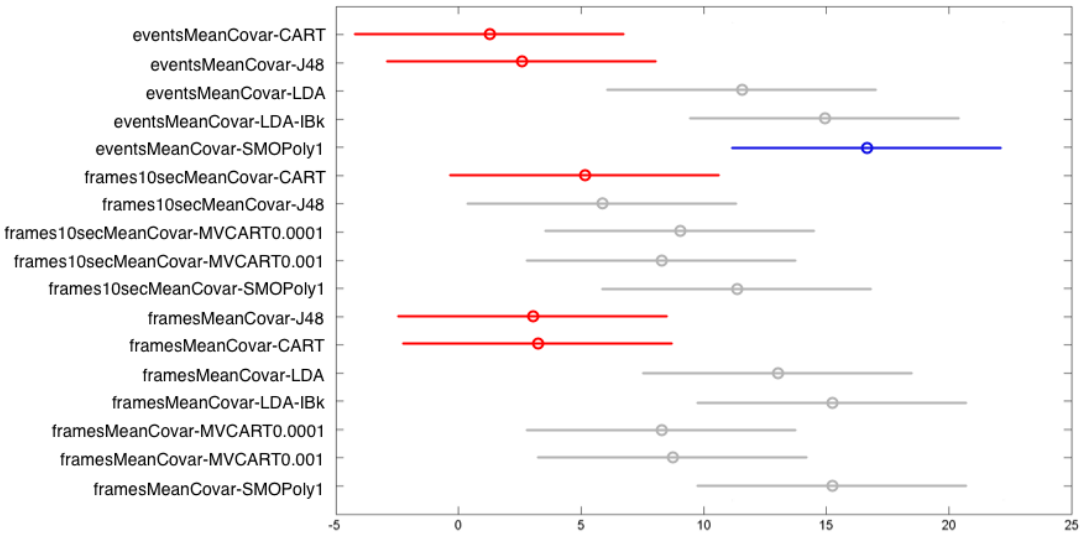


Figure 4.7: Comparison of Decision Tree classifiers and Linear Classifiers based on timbral features using the Friedman test and multiple comparisons.

The accuracy scores and the Friedman test results given in Figure 4.7 show that for the classification of a single mean and covariance timbral feature summary vector computed over the whole file, or multiple 10 second segments of it, the performance of Linear classification models (SMOPoly1, LDA, LDA-IBK) significantly exceeds that of standard Decision Tree classifiers (CART and J48). In particular, the Support Vector Machine Classifiers (SMO) with a 1st order polynomial kernel (SMOPoly1) *significantly* outperformed the CART and J48 decision tree classifiers in nearly every case. The MVCART classifiers perform better than the standard CART or J48 algorithms, but not significantly so. However, they are also not significantly worse than the Linear models.

The McNemar's tests results, displayed in Figure 4.8, support the conclusion that linear models are significantly better at classifying mean and covariance summaries of the timbral feature stream, as there are only statistically significant differences between conventional Decision Tree classifiers, Linear models or MVCART models. However, these maybe built on feature sets with different segmentations. E.g. The J48 classifier trained on the mean and covariance summary of event feature vectors is only not significantly different from the CART and J48

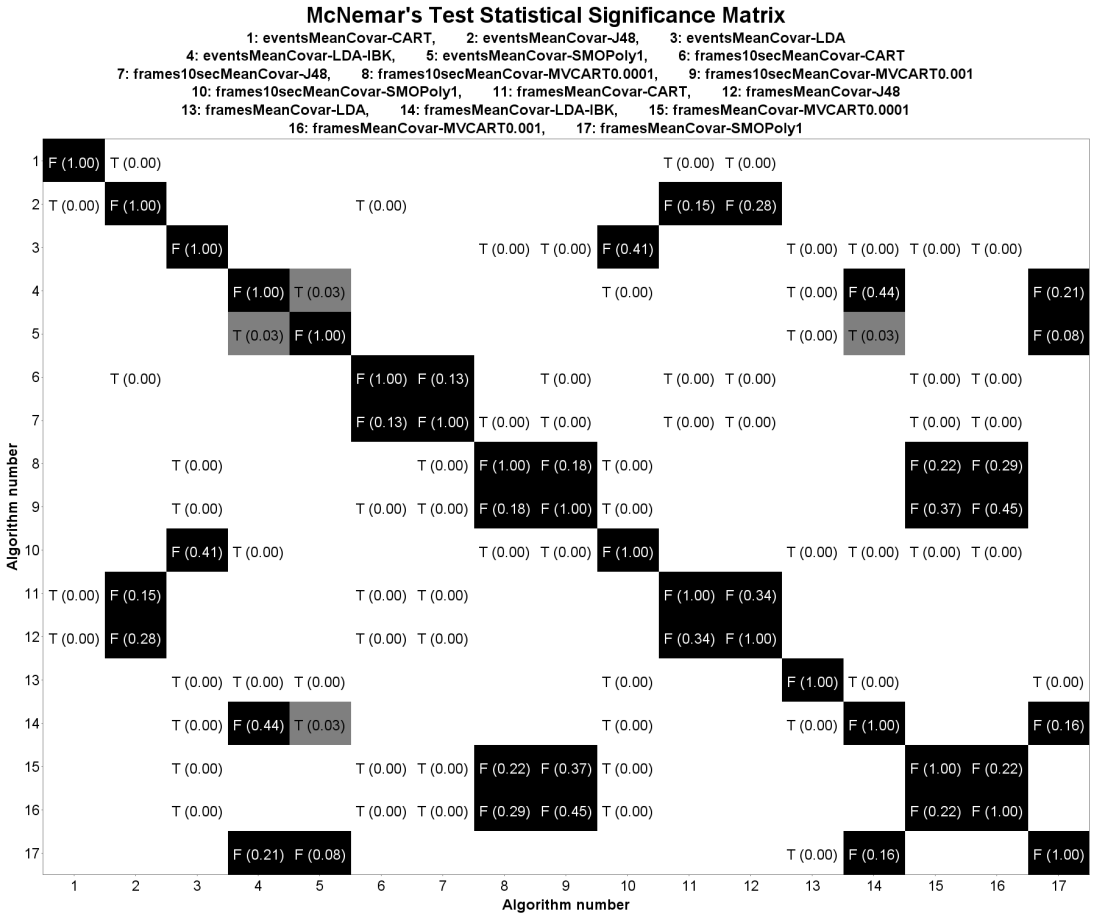


Figure 4.8: Comparison of Decision Tree classifiers and Linear Classifiers based on timbral features using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

classifiers trained on the mean and covariance of individual feature frames while the frames mean and covariance 1st order SMO is only not significantly different from the frames mean and covariance LDA-IBK, event mean and covariance SMO and LDA-IBK.

Hence, amongst this subset of the results the choice of classification *algorithm* was far more important than the choice of *segmentation* (the whole file or multi-

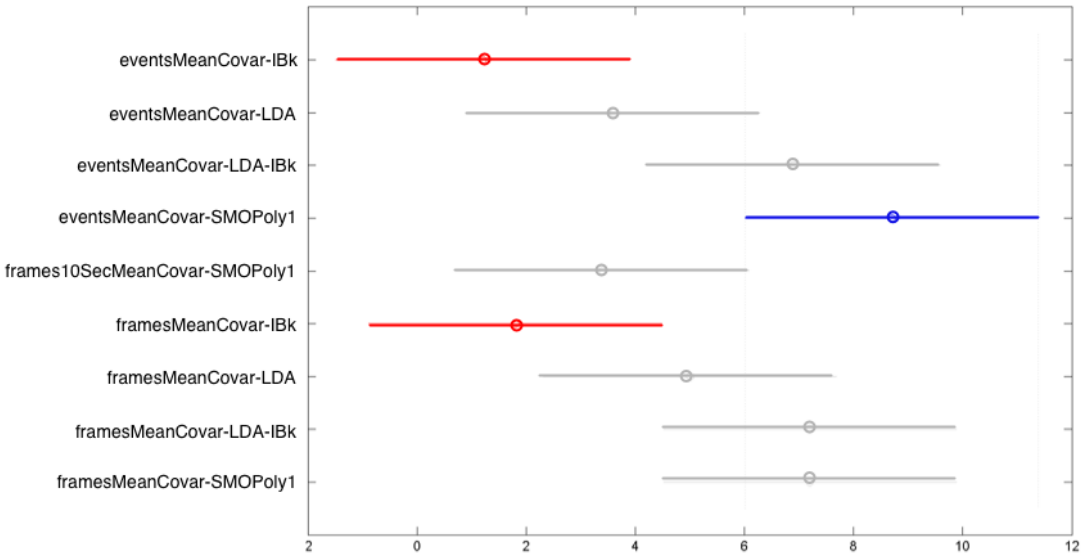


Figure 4.9: Comparison of Linear classifiers and timbral feature spaces using the Friedman test and multiple comparisons.

ple 10 second segments of the track). It should be noted that using the mean and covariance of the mean vectors for each audio event did not reduce performance, in fact it increased performance slightly, although not significantly so.

It was also interesting to note that IBK classifier (k-Nearest Neighbour) in the LDA transform space (LDA-IBK) was often more highly correlated with the SMO classifier than with the basic LDA classifier. This demonstrates the ability of the SMO to select a more robust set of separating hyper planes between the classes of audio than the LDA classifier using criteria based on maximum margins between classes. However, the projection produced by the LDA provides highly similar performance to the SMO when using an instance based classifier, perhaps indicating that the issue is not with the projection but rather the classification strategy (based on measuring the distance between an example and the transformed mean of each class of data).

The Friedman test results given in figure 4.9 compare linear classifiers on the different feature sets and further illustrate these observations. The SMOPoly1 classifiers significantly outperform only the standard instance-based classifiers

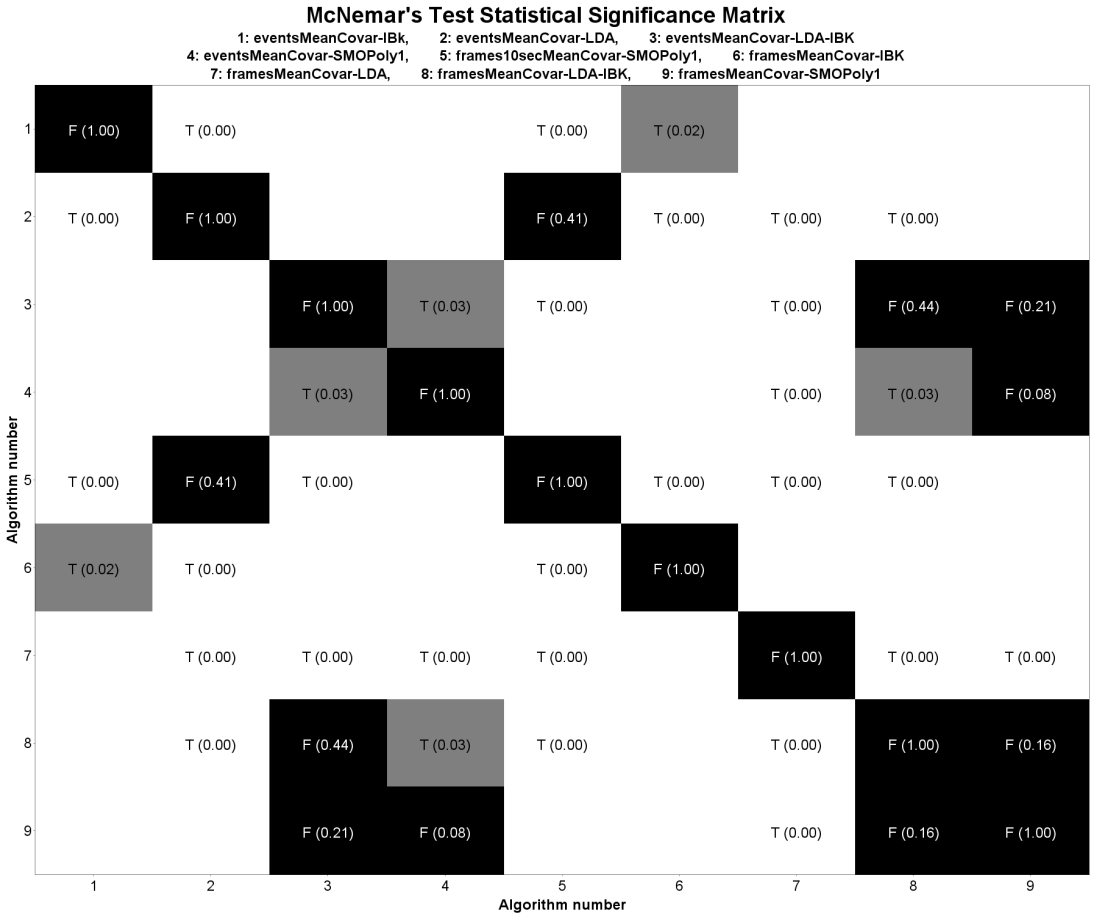


Figure 4.10: Comparison of Linear classifiers and timbral feature spaces using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

(IBK). However, the standard LDA classifiers do not significantly outperform IBK, but an instance based classifier trained in the LDA space (LDA-IBK) does significantly outperform the standard Euclidean IBK. Again the choice of feature segmentation seems less important than the choice of classifier, although it should be noted that the use of 10 second audio frames does hurt performance.

The corresponding McNemar's test matrix in figure 4.10 demonstrates a lack

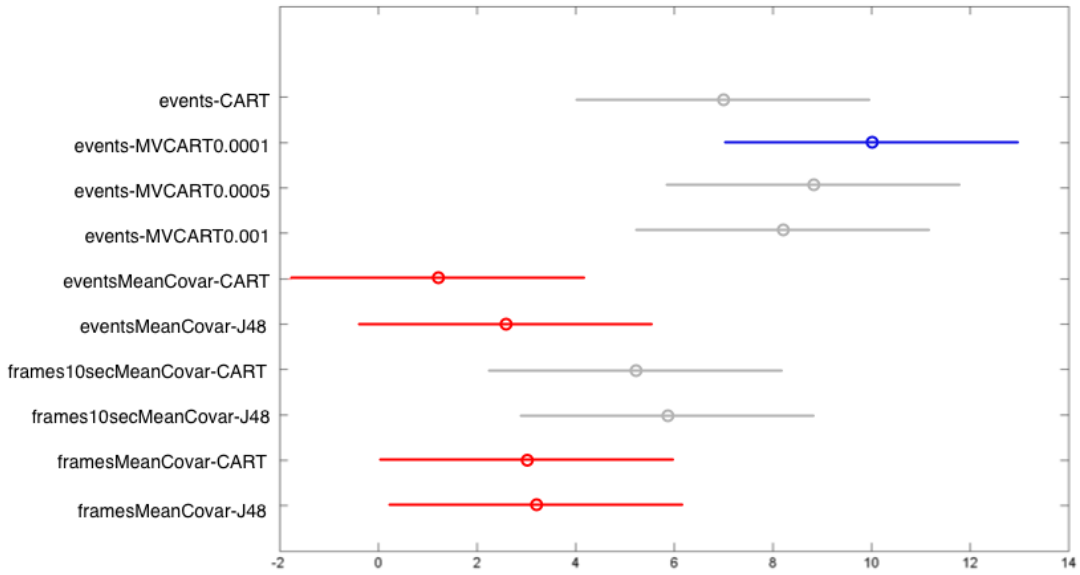


Figure 4.11: Comparison of Decision Tree classifiers and timbral feature spaces using the Friedman test and multiple comparisons.

of significant differences only between classifier based groupings, i.e., the IBK classifiers are only not significantly different from other IBK classifiers, LDA-IBK and SMO classifiers are not significantly different from each other, while the standard LDA is often significantly different to all other systems.

The Friedman test results given in figure 4.11 compare the performances of the Decision tree classifiers on different segmentations of the features stream. It is interesting to note that the performances of the J48 and CART classifiers are enhanced by using shorter segmentations. For example, the results for 10 second mean and covariance summaries are better than those for the whole files although not significantly so, while the results for the classification of event vectors (i.e. many vectors per file, produced by onset detection, rather than one or one every ten seconds) significantly improves the performance of the CART model over the event mean and covariance CART model. This is the opposite trend to that observed with the linear classifiers where the 10 second vectors reduced the performance of the resulting classifier.

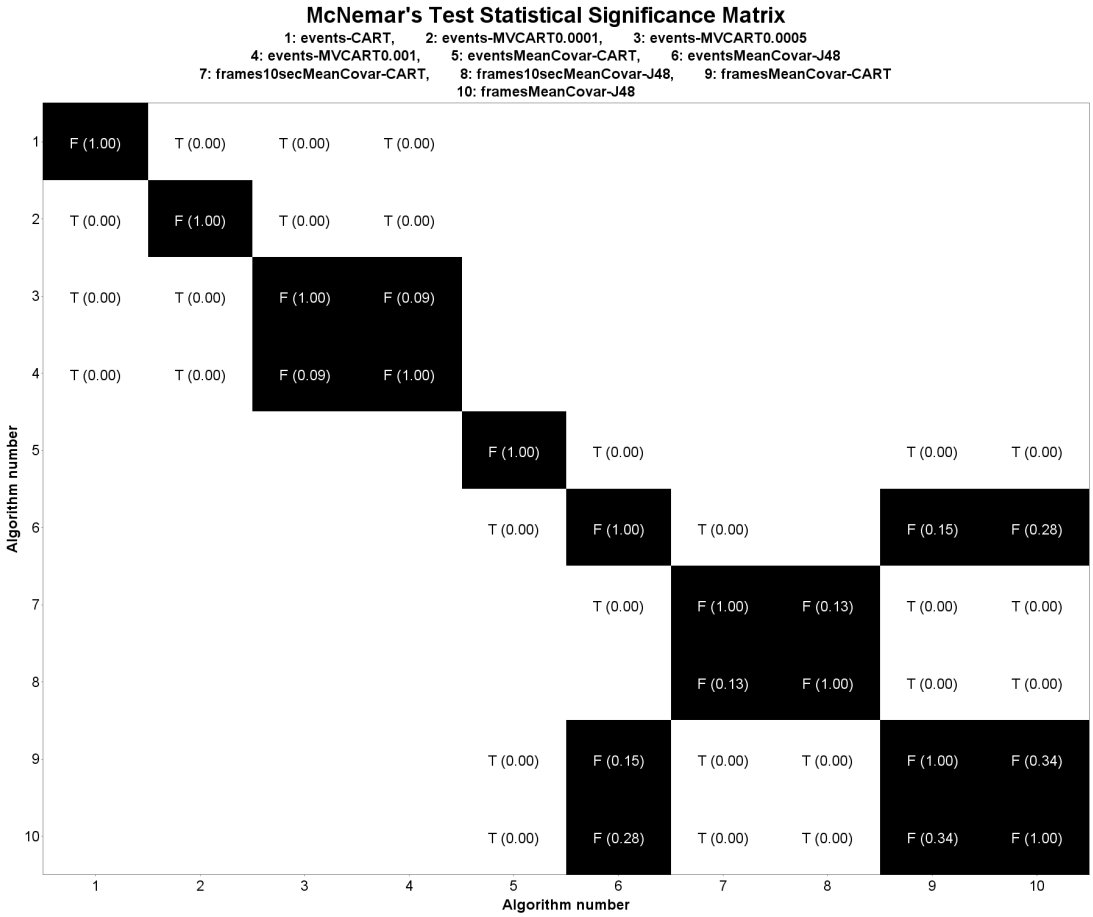


Figure 4.12: Comparison of Decision Tree classifiers and timbral feature spaces using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

The best performing results are for the MVCART based classifiers trained on many short event vectors per file. The largest MVCART tree (requiring a minimum of 0.1% of the training data at each node) is the best performing although there is no significant distance between that and those requiring 0.5% or 1% of the training data. The McNemar's test results given in figure 4.12 further support these conclusions by showing that the MVCART classifiers show only

non-significant differences to each other, while CART and J48 models are not significantly different across the frames and event mean and covariance features. However, no Decision Tree models were correlated across different segmentations (event mean vectors, vs. summaries of whole files).

As shown in table 4.3, the training of both of the linear classifiers (LDA and SMO Poly1) failed on the event-level feature vectors. Both algorithms were allowed long runtimes in order to complete training, but were generally unable to provide a projection of the feature space produced by the event-level feature vectors. Only the first order polynomial SMO managed to complete a single iteration, returning an accuracy score of 51.45% after a week long run, only improving over the single Gaussian classifier. We can only speculate that in this feature space the large number of vectors produced are likely to have a very large range of values for each track (high variance in the different sounds that make up a single piece of polyphonic music) and that there will be a high level of correlation between vectors drawn from different genres (e.g. we might expect to see a vector representing a snare drum *event* in tracks drawn from a variety of genres). Hence, we might expect the feature space to be too complex to *linearly* project into a lower subspace in which to define an effective set of separating hyper-planes. Hence, it is highly likely that the ability of the Decision trees to recursively subdivide the feature space into contiguous regions and thereby identify specific characteristics of each sound event is allowing them to produce better performance on this data type.

Figure 4.13 compares variants of the CART and MVCART classifiers based on feature vectors for each audio event. As demonstrated in figure 4.11, Decision Tree classifiers can produce significantly better performance over the raw event vectors than over mean covariance of the event vectors over the whole file. This is achieved by classifying each individual vector, and combining the classification likelihoods (as described in section 4.11) by making a weak independence assumption and taking their product (summing the log likelihoods). Figure 4.13 demonstrates that the performance of the alternative method of combining the classification likelihoods given in section 4.2.5.2 is higher, but not significantly so. This approach is based on the conversion of the leaf node sequence into TF

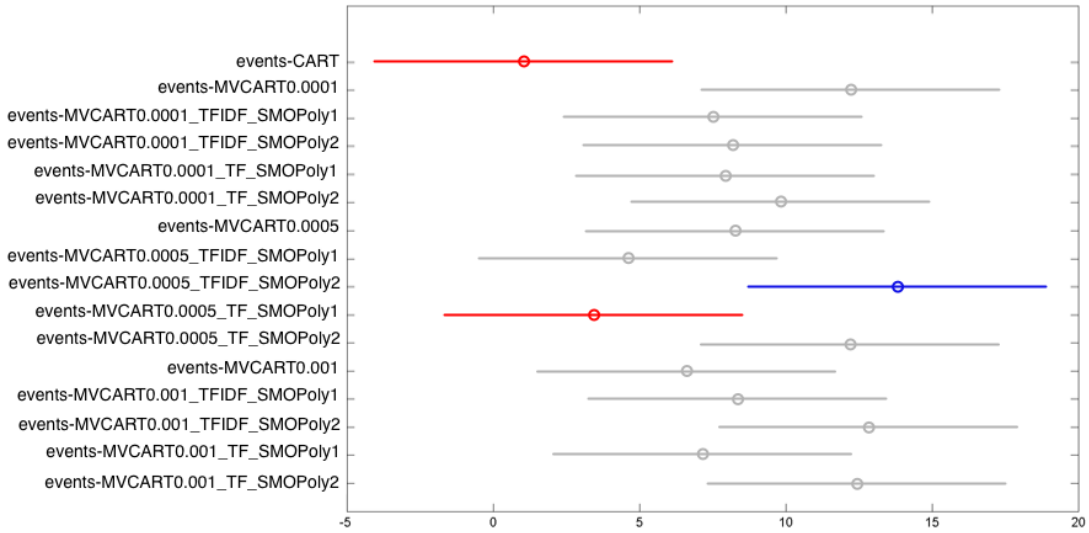


Figure 4.13: Comparison of MVCART-based classifiers using the Friedman test and multiple comparisons.

or TF/IDF vectors that are subsequently classified by a Support Vector Machine classifier (SMO).

The 'deepest' model (the model requiring the lowest number of examples in order to split a node), 0.0001, produced the highest accuracy using the combination of likelihoods, significantly outperforming a standard CART model, where the shallower models (0.0005 and 0.001) outperform the CART model, but not significantly. However, when converting the leaf node sequence into a TF or TF/IDF vector and classifying with a support vector machine (SMO), the smaller models outperform the deepest model. The choice of TF or TF/IDF vectors or the depth of model did not significantly affect performance alone (although TF/IDF vectors do tend to produce better results than TF vectors alone), with greater differences being produced by the choice of kernel in the SMO classifier. Polynomial kernels of degree one to three have been tested, showing that the second order polynomial always produces better results than either lower or higher order kernels, although not significantly so.

The McNemar's test matrix given in figure 4.14 shows that all the systems are uncorrelated with the behaviour of the standard CART model upon which

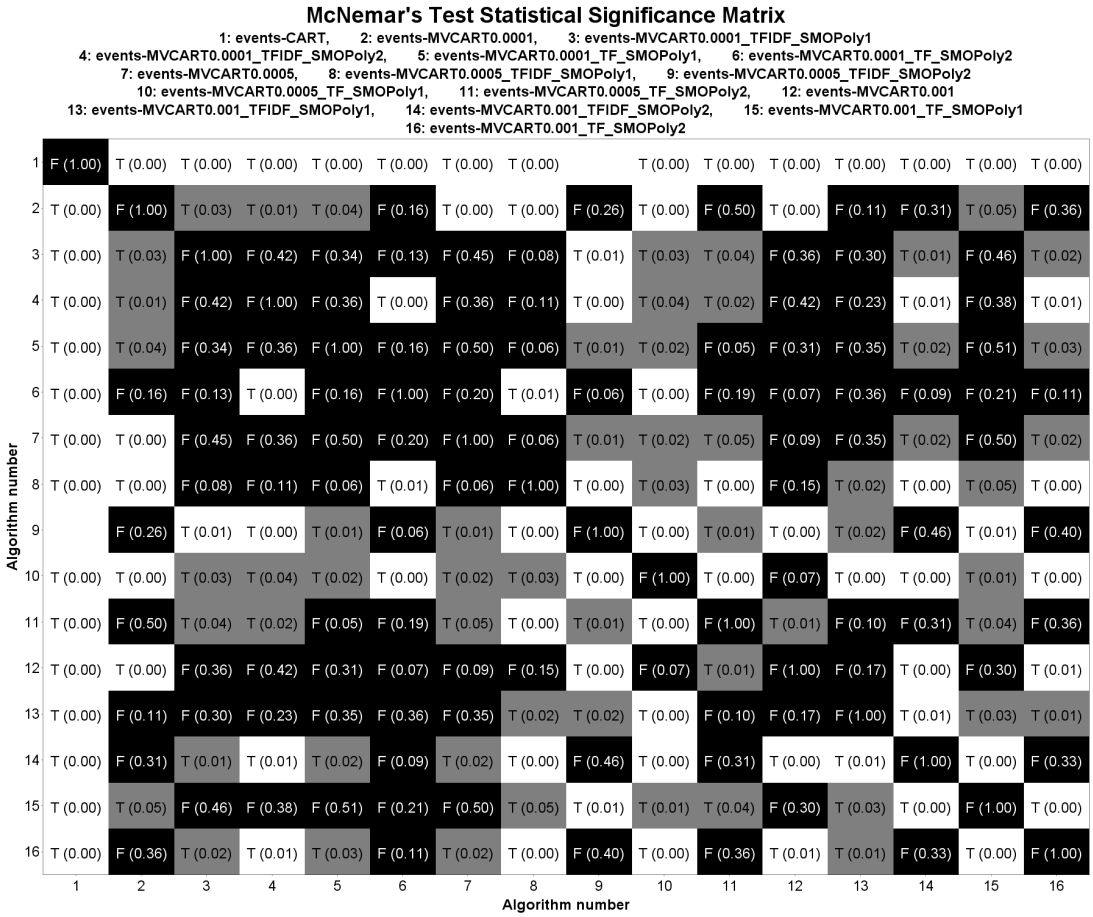


Figure 4.14: Comparison of MVCART-based classifiers using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

the MVCART is based. Other patterns are difficult to discern, although the most shallow (0.001) and deepest (0.0001) models appear to be more correlated than they are with the middle model (0.0005). Also the degree of the Polynomial kernel seems to generally have a very strong effect on whether transcription systems based on different models are correlated (i.e. unless they share the same order of Polynomial kernel they are not likely to be correlated, even when based on the

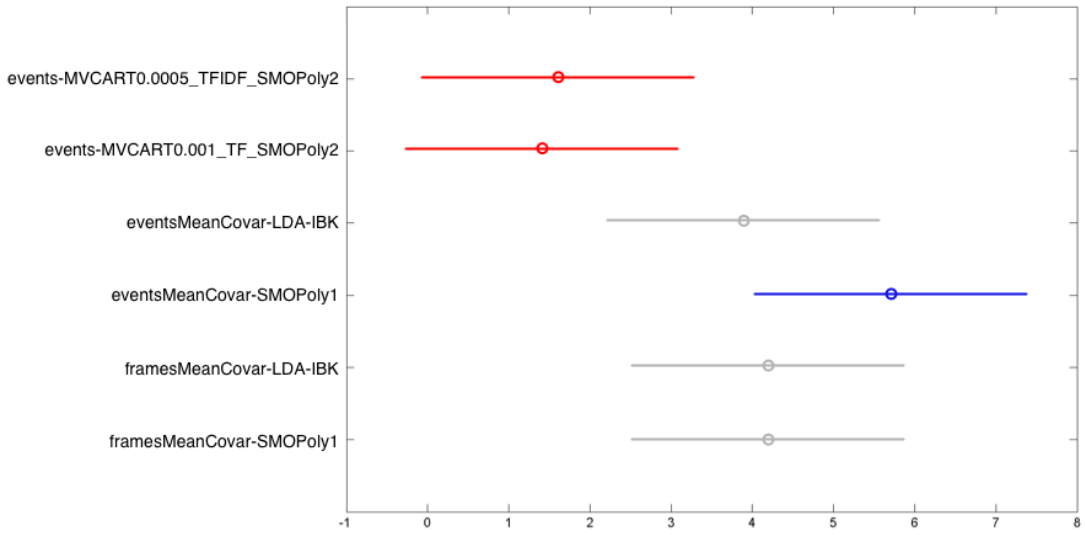


Figure 4.15: Comparison of MVCART transcription classifiers and Linear classifiers using the Friedman test and multiple comparisons.

same model).

It should be noted that the MVCART tree producing the strongest results also produced the weakest – through variation of the combination of Polynomial Kernel degree and Term Frequency weighting (TF, Poly1 vs. TF/IDF, Poly2).

The use of a system that converts the node sequences into term frequency vectors has an advantage over the combination of likelihood vectors, in that it uses a smaller model to produce moderately higher performance. Further, it should also be noted that any of the purely event-based classifiers detailed in this section can be simply extended to real-time classification, which may facilitate their application in scenarios where 'whole file' models may not be appropriate.

The final set of significance test plots for the timbral classifiers, figures 4.15 and 4.16, compare the performance of the MVCART transcription-based classifiers with the best performing Linear models (SMOPoly1 and LDA-IBK). Only the event mean and covariance based SMO significantly outperforms the best performing transcription models (0.0005, TF/IDF, SMO Poly2 and 0.001, TF, SMO Poly2). However, the linear models do generally outperform the MVCART-based

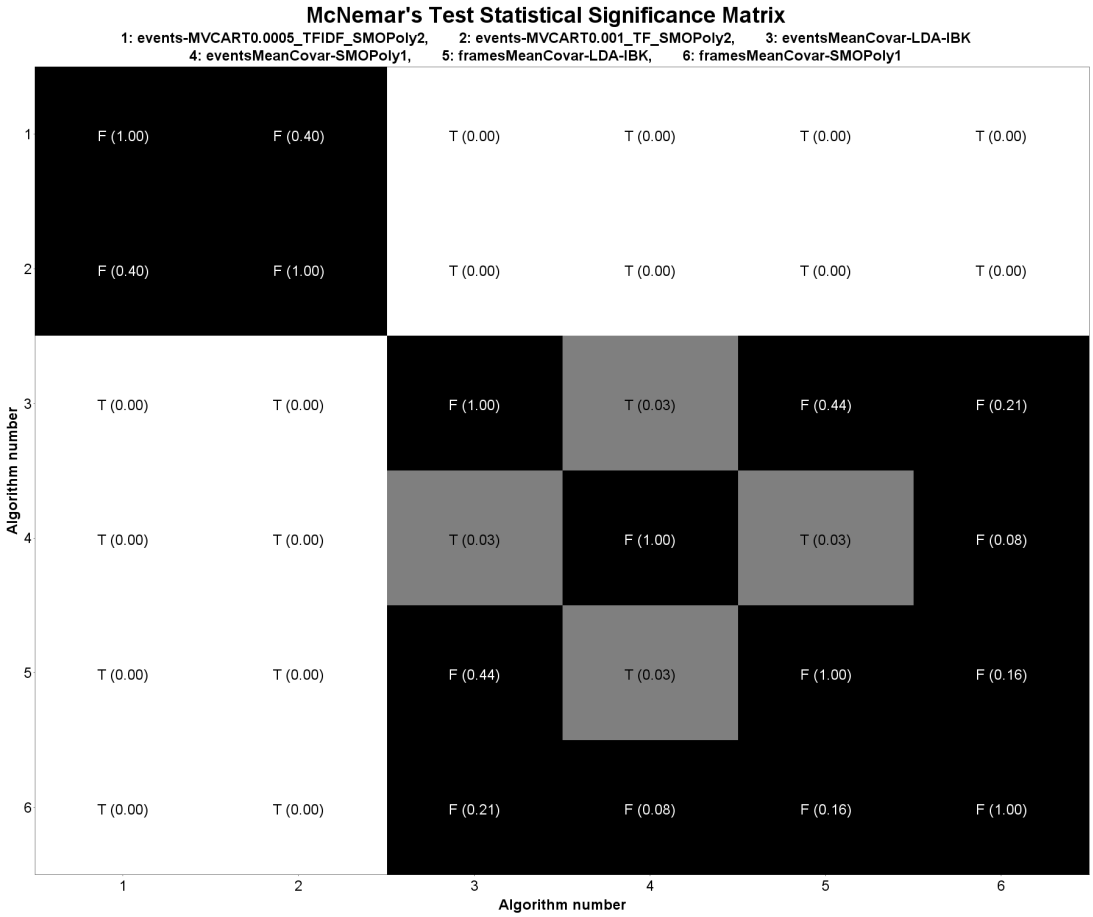


Figure 4.16: Comparison of MVCART transcription classifiers and Linear classifiers using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

classification models. The McNemar's comparisons of these systems show correlation's amongst the MVCART models and amongst the linear classifiers, but no similarities between those two groups. There are marginal differences between the LDA-IBK models and the SMOs, but significant differences between all the linear classifiers and the transcription models.

To summarise, the best performing models are based on linear classifiers trained on the mean and covariance summaries of feature streams. However, Decision Tree classifiers perform much better than linear classifiers on very short (event based) segments of tracks and are improved by any segmentation of the track into shorter units (results are improved by using 10 second segments of tracks). This may be because as the segment length is shortened, the feature space becomes more complicated.

For example, with event level segments we might expect to represent a single piano note (or piano note accompanied by a guitar chord), a guitar chord, a drum hit, all in different segments – which should map to very different parts of the feature space, even within a single track. However, over full tracks, we are only modelling the overall distribution of timbres in the track. Hence, these different approaches are likely to be working in very different manners. Within the feature space produced by the event level feature vectors, we would expect to find a complicated structure composed of a great many clusters of event vectors corresponding to a plethora of individual instrument, electronic or vocal sounds. We would not expect to find all the sound events used in a particular track to exist in a contiguous region of the feature space: rather, they are likely to be mixed together with clusters from other music types with similar instrumentation. In contrast, we expect the whole file summary features (mean and covariance vectors) to produce a relatively simple feature space, where clusters of track profiles of particular music types do exist in contiguous regions of the space. Hence, the Decision Trees yield better performance than the Linear classifiers.

Although the classification of event-based segments does not provide as high a level of performance as linear classifiers based on mean and covariance summaries of tracks, it is a very different approach with significant scope for enhancement. Firstly, there is scope to improve the MVCART classifier further by replacing the current splitting process (the computation of a single LDA transform followed by the division of the classes into two groups and modelling each with a single Gaussian in the LDA space) with the training of binary first order polynomial SMO classifiers for each division of the classes into two groups. Further, sound events maybe further modelled by examining their profile in time, perhaps using a similar approach to that used in audio synthesis, where a sound may be modelled

as a number of states (Attack, Delay, Sustain, Release – ADSR). Other possible extensions include the modelling of sequences of events using Discrete HMMs or N-gram models, or the transcription based approaches maybe applied to the output of a source separation algorithm (i.e., term frequencies can be computed over all the separate source audio signals produced). In particular, the application of a transcription based classifier may be an excellent way of overcoming the issue of how to compare tracks with a variable number of sources (in our system, a single Term frequency vector is produced over all sources).

Further, Aucouturier suggests that the 'Glass Ceiling' that he has observed on the performance of timbral music similarity and classification measures may be caused by the inability of the models to separate audio into individual streams (as humans are able to) and analyse each stream independently. Using a source separation technique, the MVCART models would be able to address each instrument sound in near isolation (although there is likely to be some cross-over between sources with any current source separation technique). In such a situation, we might expect the MVCART models to be better able to detect and model nuances of the playing style, intonation and timbre of each distinct instrument or voice type and thereby better identify relationships to a particular genre or genres. Further, we can hope that the resulting combined term frequency vector (over all the sources) more clearly models the polyphonic timbre.

4.3.4.2 Rhythmic classification results

The confusion matrices for the rhythmic classification systems tend to show a very similar pattern to that produced by the classifiers based on timbral characteristics, by showing strong confusion between classical music types (Baroque, Romantic, Classical), Jazz & Blues and Between Rock, Rock & Roll, Hard Rock & Metal and Country. The better performing classifiers tend to resolve these patterns more clearly. In particular music types that are well defined by the rhythmic characteristics, such as Rap & Hip-Hop or Electronica & Dance are very accurately identified, with little confusion with other classes.

Where music types are not well differentiated by their rhythmic patterns (such as the classical music types, or Rock and Rock & Roll) significant confusion does

Table 4.4: Classification accuracy - RCC-based models

Feature set	Segmentation/summary	Classifier	Acc	St.dev.
Rhythm	9 second windows	CART	39.65	0.50
Log-scale		IBk	37.44	0.70
		LDA	19.30	0.26
		MVCART0.001	**41.29**	**1.02**
		SMO Poly1	27.05	0.60
Rhythm	9 second windows	**CART**	**38.84**	**0.21**
Log-scale after 2 Hz		IBk	35.96	0.51
		LDA	27.04	0.38
		MVCART0.001	38.32	0.44
		SMO Poly1	26.51	0.54
Rhythm	9 second windows	**CART**	**38.84**	**0.60**
Log-scale after 3 Hz		IBk	35.39	0.38
		LDA	27.66	0.14
		MVCART0.001	37.78	0.38
		SMO Poly1	25.86	0.34
Rhythm	9 second windows	CART	37.68	0.38
Log-scale after 4 Hz		IBk	37.68	0.45
		LDA	18.49	1.96
		MVCART0.001	**41.75**	**0.67**
		SMO Poly1	26.38	0.56
Rhythm	9 second windows	CART	28.70	0.53
Log-scale after 4 Hz	Mean	IBk	33.64	0.27
		LDA	20.32	1.52
		MVCART0.001	**34.19**	**1.18**
		SMO Poly1	25.27	0.55
Rhythm	9 second windows	CART	28.71	0.31
Log-scale after 4 Hz	Mean and variance	IBk	33.69	0.14
		LDA	18.36	1.26
		MVCART0.001	**34.33**	**0.48**
		SMO Poly1	25.49	0.57

occur. The similarity between the confusion produced by these systems and those based on timbral features maybe largely due to the composition of the database. Our test database offers few examples of music that is differentiated from other classes largely by rhythmic characteristics, as we would expect in a database of modern dance music. The only two classes that are very well differentiated from the others by the system (Hip-Hop and Electronica & Dance) are those defined in some way by their rhythm. In contrast, the timbral classification systems performed fairly consistently over all classes. Were the Electronica & Dance class

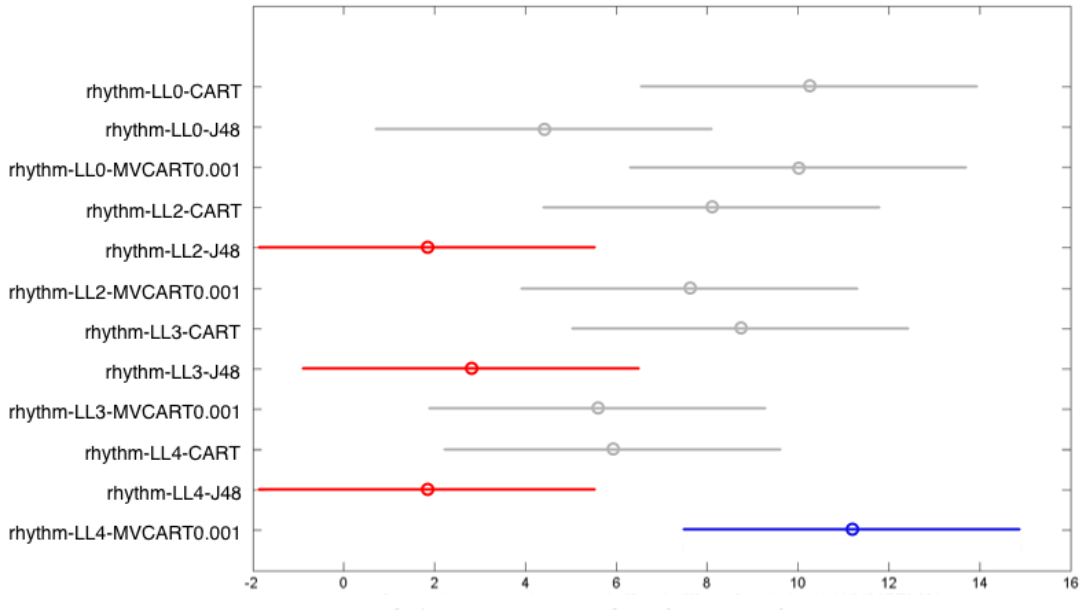


Figure 4.17: Comparison of Decision Tree classifiers against the filter linear/log scaling transition frequency using the Friedman test and multiple comparisons.

to be split into a number of sub-classes, differentiated by rhythmic patterns, we might reasonably expect the RCC based classifiers to do better at differentiating them than the timbral models.

The first Friedman test results plot, figure 4.17, compares the performance of different decision tree classifiers against the filter transition frequency used in the RCC filterbank (i.e. the frequency at which the filter spacing stops being linear and becomes logarithmic). Generally, the filter transition frequency does not appear to affect the performance strongly, producing no significant differences. However, the application of different decision tree classifiers does seem to have a strong effect.

The J48 classifier (the worst performing decision tree model) generated the deepest decision trees with the (default) stopping criteria based on a minimum of 2 example vectors per leaf node. The CART and MVCART classifiers included in table 4.4 were produced with a threshold of 0.001 times the number of training examples. As our test database is composed of 30 second clips and RCC vectors

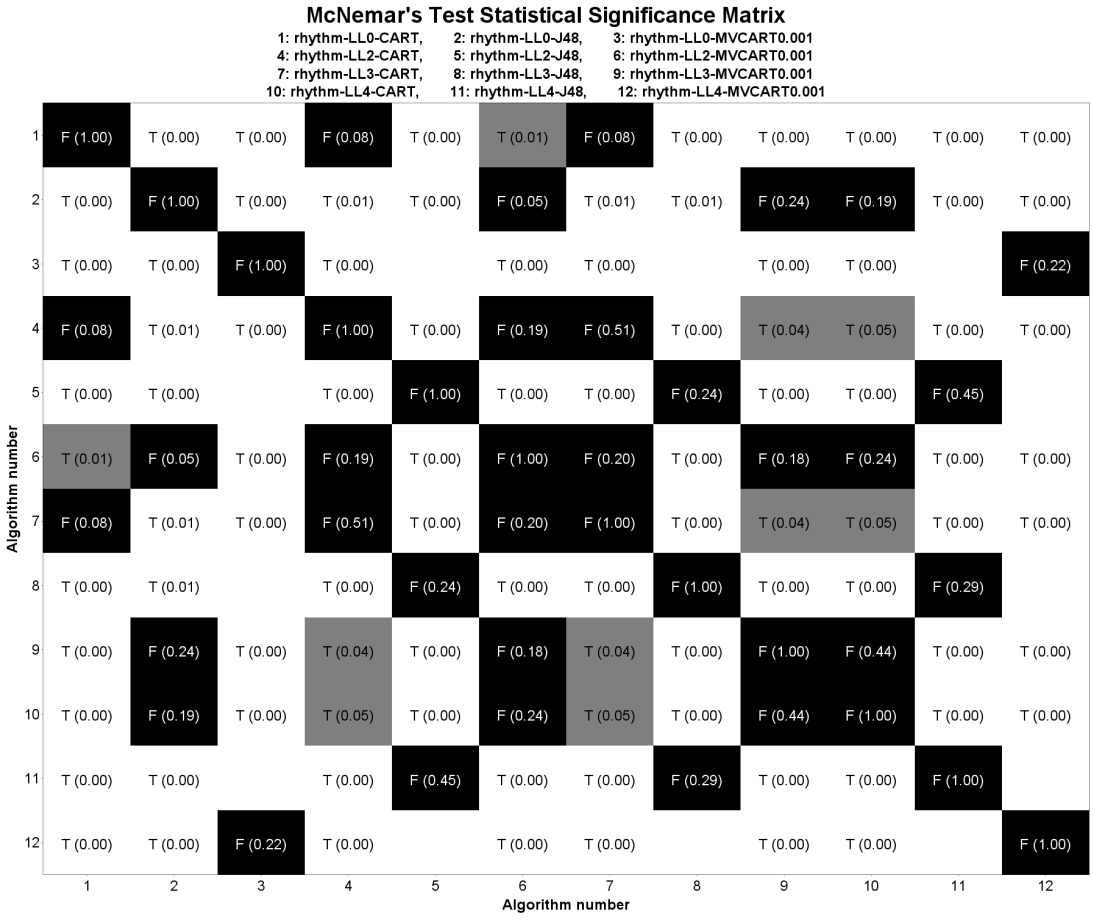


Figure 4.18: Comparison of Decision Tree classifiers against the filter linear/log scaling transition frequency using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

are computed for 9 second frames with an overlap of 50%, we get 6 vectors per track. Given an 80:20 split of the database into training and test sets, this equates to a stopping criterion of a minimum of 39 vectors in order to split a node. The full results given in Appendix A.1 also include MVCART classifiers produced with a stopping criterion of 0.0001, which equates to a minimum of 4 vectors in order to split a node. Generally, the shallower Decision Trees have performed

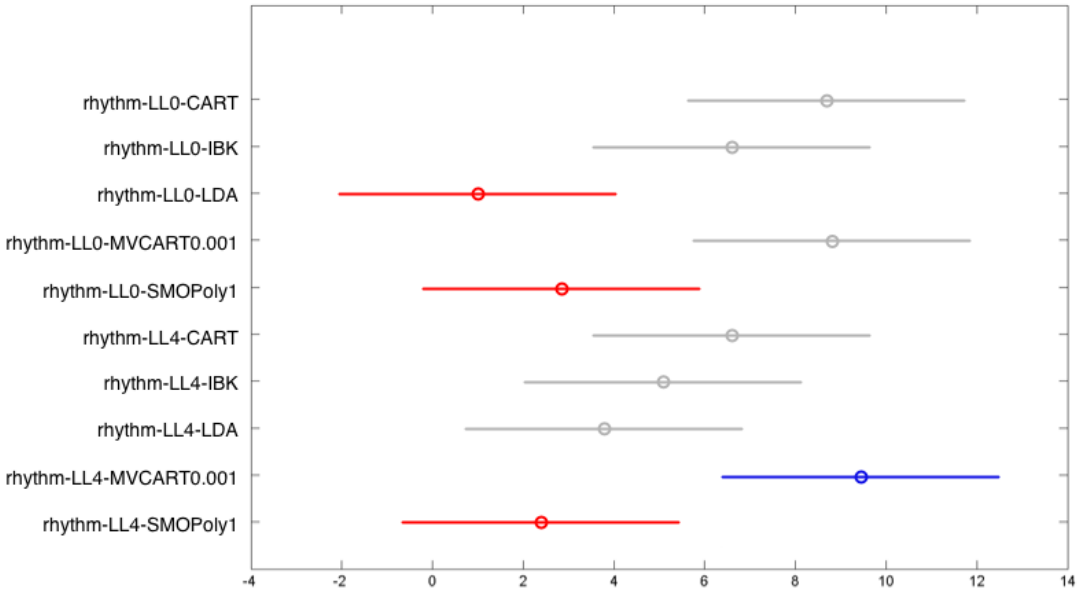


Figure 4.19: Comparison of Decision Tree classifiers against linear classifiers based on RCC features using the Friedman test and multiple comparisons.

better, although not significantly.

The MVCART classifiers often perform significantly better than the J48 classifiers. However, their performance is very similar to that of the standard CART classifier with the same stopping criterion. The McNemar's test results, given in table 4.18, show scattered instances of non-significant differences between the error-rates of many of the decision tree classifiers, although the J48 classifiers are most often not significantly different from each other. The Decision trees based on RCCs produced with cut-off frequencies of 2 Hz and 3 Hz are most often not significantly different. However, it is interesting to note that the MVCART 0.001 models for the LL0 and LL4 RCCs were only not significantly different from each other, despite the fact that they used the most dissimilar filterbanks. This suggests that the periodicities between 2 and 4 Hz (120 - 240 BPM) are quite important and must be represented on a *consistent* scale.

Figures 4.19 and 4.20 compare the performance of linear classifiers to that of Decision Trees on the two best performing RCC sets (LL0 and LL4). Figure

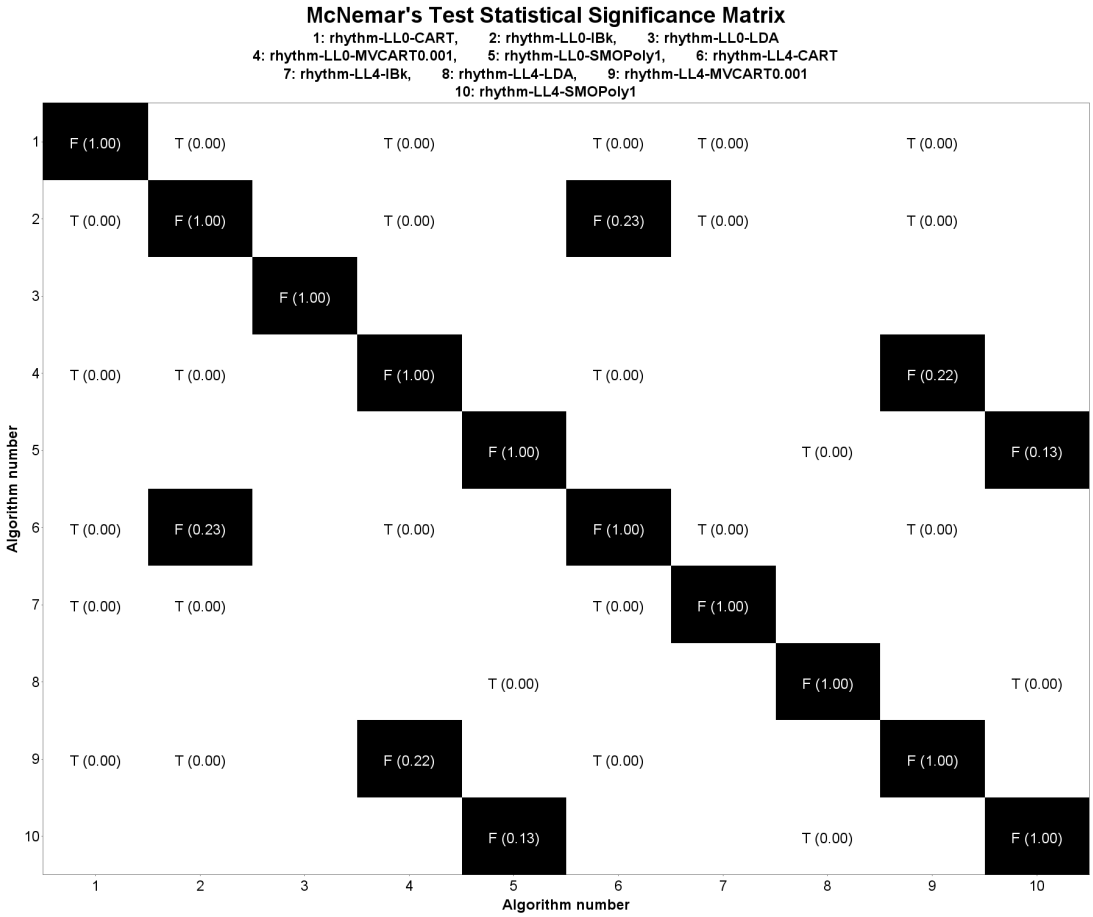


Figure 4.20: Comparison of Decision Tree classifiers against linear classifiers based on RCC features using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

4.19 shows that the MVCART 0.001 models and (for the LL0 RCCs) the CART models significantly outperform the many of the linear classifiers including both of the SMO's and the LL0 LDA. The LL0 RCC results are characterised by good decision tree performance and poor linear classifier performance. The LL4 RCCs produced a slightly better LDA, poor performance from the SMO and the best overall performance for the MVCART 0.001 model – at a very similar level to

that produced on the LL0 RCC features. The IBK models produce better performance than the linear models (although not significantly), but lower performance than the CART and MVCART decision tree classifiers (although again not significantly). Further, Figure 4.20 shows no correlation between the error-rates of the linear models and any of the decision tree classifiers using the McNemar's test.

The reason for the difference in performance between the decision tree classifiers and the linear models on this data may again be due to the ability of the Decision Trees to sub-divide the feature space recursively into contiguous regions. We might expect the RCC feature spaces to be more similar in nature to the event-level timbral feature space, with a number of different clusters of rhythmic patterns corresponding to a particular genre of music. These patterns would be related to the cultural definition of the rhythms used in a particular genre of music, and might not exist in contiguous, homogenous clusters in the feature space. Hence it would be very difficult for a simple linear classifier to draw a separating hyper-plane between two classes.

The final statistical significance plots, figures 4.21 and 4.22, compare the performance of the full 6 RCC vectors per track against models based on their means or means and variances, using the best performing RCC set (LL4). These plots show that the use of the full set of RCC vectors is often superior with both the MVCART and CART models significantly outperforming their counterparts trained on the mean or mean and variance vectors. The McNemar's matrix given in figure 4.22 demonstrates no correlation between any of the results based on the full set of vectors and those based on the mean or mean and variance RCC vectors.

Despite the very slightly better performance of the mean RCC vectors over the mean and variance RCC vectors, the mean and variance vector may be preferable, as they appear to produce better patterns of confusion: the mean vectors tended to misclassify many classes as 'Electronica & Dance' rather than distributing the confusion more sensibly.

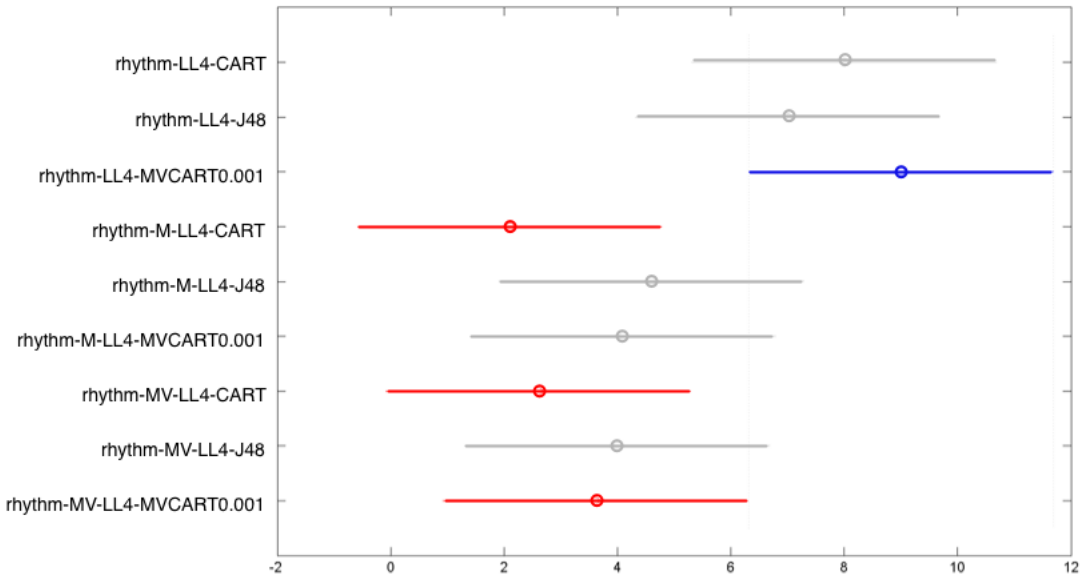


Figure 4.21: Comparison of full, multiple vector RCC-based classifiers against single vector RCC mean and mean & variance classifiers using the Friedman test and multiple comparisons.

4.4 Combining models

Classification algorithms may be combined in an attempt to produce a system with higher overall performance than the best amongst the candidate classifiers. Such systems are often referred to as classifier ensembles. However, for the combined performance of the system to improve over the best individual system, it is important that there is a degree of diversity in the output of each system forming part of the ensemble, i.e. there is little point in combining a system with another that provides very highly correlated results. In contrast, the combined outputs from significantly different systems may provide better results than the best performing candidate (42). This diversity may be introduced by training classifiers on different "slices" of the dataset, or with different modelling parameters (such techniques include bagging and boosting) (42). An alternative approach is train each classifier on a significantly different feature set. For example, in our experiments, we have produced a range of different classifiers trained on timbral or rhythmic features of music.

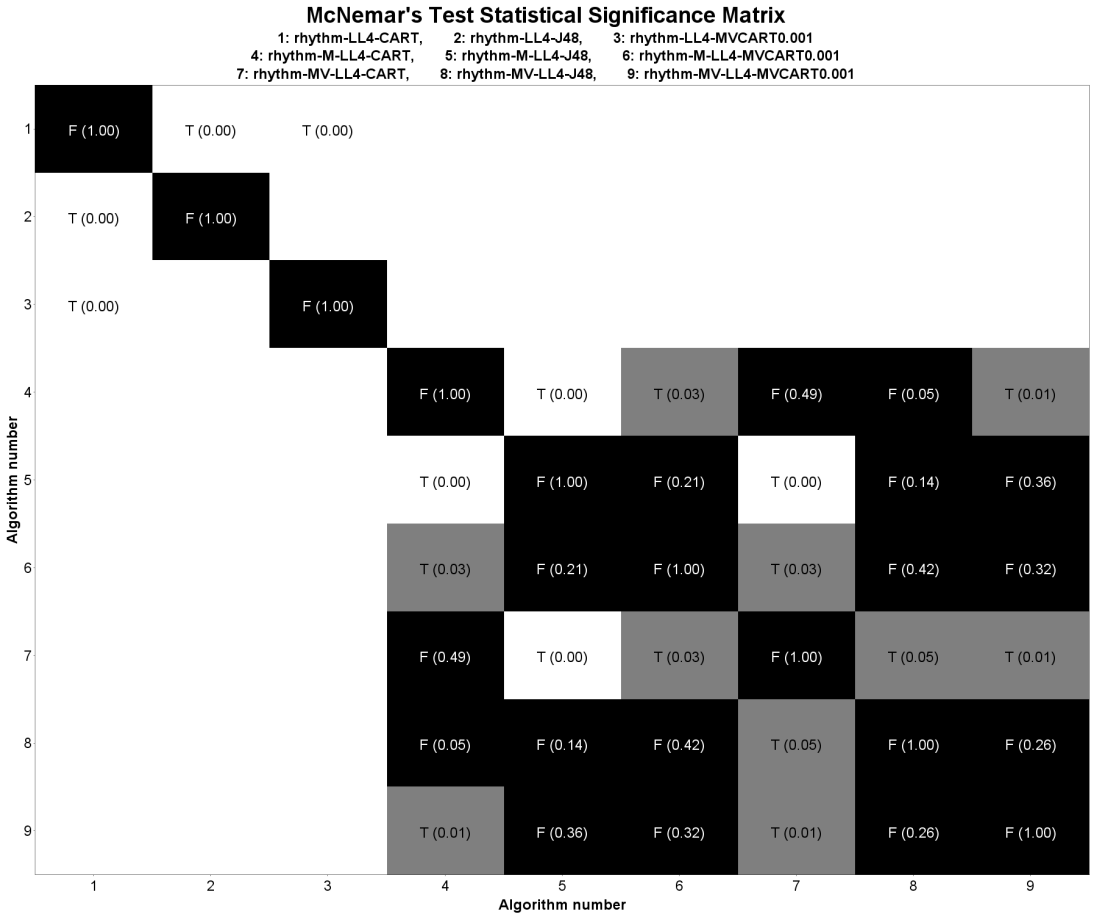


Figure 4.22: Comparison of full, multiple vector RCC-based classifiers against single vector RCC mean and mean & variance classifiers using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

4.4.1 Nomenclature

In the following sections the nomenclature described below is used:

V Number of genre classes.

i,j Subscripts used to iterate over classes.

M Number of classifiers.

m, n Subscripts used to iterate over classifiers.

c_m The m-th classifier.

x A particular track.

T The number of tracks in the training dataset.

T_i The number of tracks in the training dataset belonging to class i.

$P_i(x|c_m)$ Posterior probability of class i for track x given classifier c_m.

$\mathbb{P}(x|c_m)$ The set of posterior probabilities for track x over all V classes, given classifier c_m:
$\mathbb{P}(x|c_m) = \{P_1(x|c_m), P_2(x|c_m), \ldots P_V(x|c_m)\}$
such that:
$\sum_{i=1}^{V} P_i(x|c_m) = 1$

w_m A weight to be applied to outputs ($\mathbb{P}(x|c_m)$) from classifier c_m.

w_{im} A weight to be applied to the probability ($P_i(x|c_m)$) for class i from classifier c_m.

4.4.2 Combined feature vectors

In order to combine classification algorithms based on single vectors of features, one possibility is to build a single classifier that uses the concatenated feature vectors. This enables us to combine our single vector timbral and rhythmic parameterisations of the audio signal into a single classification algorithm. However, for classification algorithms based on a variable number of feature vectors, alternative methods of combination must be used (as there exists no single summary feature vectors to concatenate). To combine such systems, we can use techniques based on the combination of posterior probability profiles produced by the classifier (classification likelihoods).

4.4.3 Combining posterior probability profiles

As stated in section 4.2.4, most classification algorithms can be used to estimate the posterior probabilities of classification ($\mathbb{P}(x|c_m)$) for a track x and a classifier c_m, also known as a classification likelihood profile. We can write $\mathbb{P}(x|c_m) = \{P_1(x|c_m), P_2(x|c_m), \ldots P_V(x|c_m)\}$, where V is the number of classes and $\sum_{i=1}^{V} P_i(x|c_m) = 1$. In most classification systems a single class label with the highest degree of support in a classification likelihood profile ($\max(\mathbb{P}(x|c_m))$) is used as the estimate of the class label. An example illustrating a possible classification decision is shown in figure 4.23A.

In order to combine the outputs of several different classification algorithms $(c_1, c_2, \ldots c_M)$, with the intent of achieving better performance than any of the individual algorithms, each classifier's decision may be considered a vote and collected into a new profile, $\mathbb{V}(x)$, where $\mathbb{V}(x) = \{v_1(x), v_2(x), \ldots v_V(x)\}$. Let $\text{isMax}(P_i(x|c_m))$ be an Kronecker delta function, such that:

$$\text{isMax}(P_i(x|c_m)) = \begin{cases} 1, \text{if } P_i(x|c_m) = \max(\mathbb{P}(x|c_m)) \\ 0, \text{if } P_i(x|c_m) \neq \max(\mathbb{P}(x|c_m)) \end{cases} \tag{4.32}$$

I.e. It takes a value of one if class i has the highest posterior probability given classifier m and zero otherwise. Hence:

$$v_i(x) = \sum_{m=1}^{M} \text{isMax}(P_i(x|c_m)) \tag{4.33}$$

where M is the number of classifiers. The class receiving the most votes ($\max(\mathbb{V}(x))$) is chosen as the output classification.

However, a number of alternative schemes are available. For example, multiple labels can be applied to an example by a single classifier by defining a threshold for each label, as shown in figure 4.23B, where the outline indicates the thresholds that must be exceeded in order to apply a label. Selection of the highest peak in classification likelihood profile removes information, which could have been used in the final classification decision. Hence, the outputs of multiple probabilistic classification algorithms may be better combined through their classification likelihood profiles (42).

Simple methods for combining classification likelihood profiles include the product:

$$P_i(x) = \prod_{m=1}^{M} P_i(x|c_m) \tag{4.34}$$

weighted linear combination:

$$P_i(x) = \sum_{m=1}^{M} w_m P_i(x|c_m) \tag{4.35}$$

where w_m represents a predetermined weight for classifier m and $\sum_{m=1}^{M} w_m = 1$. Setting $w_m = \frac{1}{M}$ yields an equal weight for each classifier.

4.4.4 Decision templates

A further method of using classification likelihood information is to calculate a *Decision Template* (see Kuncheva (42)) for each class of audio (figure 4.23C and 4.23D). A Decision Template, $\mathbb{D}(i)$), is estimated as the average posterior probability profile over all classes ($\mathbb{D}(i) = \{D_1(i), D_2(i), \ldots D_V(i)\}$) for examples of class i, where:

$$D_j(i) = \frac{\sum_{x=1}^{T_i} P_j(x|c_m)}{T_i} \tag{4.36}$$

and T_i is the number of example tracks bearing class label i in the Decision Template training dataset.

A decision is made by calculating the distance of a profile:

$$\mathbb{P}(x|c_m) = \{P_1(x|c_m), P_2(x|c_m), \ldots P_V(x|c_m)\} \tag{4.37}$$

for an example track x from the available Decision Templates and selecting the closest, as illustrated in figures 4.23E and 4.23F. Distance metrics that can be used to make this comparison include the Euclidean distance:

$$Dist_{euc}(x, i) = \sqrt{\sum_{j=1}^{V} (P_j(x|c_m) - D_j(i))^2} \tag{4.38}$$

and Cosine distance:

$$Dist_{cos}(x, i) = \frac{\sum_{j=1}^{V} P_j(x|c_m) D_j(i)}{\sqrt{\sum_{j=1}^{V} P_j(x|c_m)^2} \sqrt{\sum_{j=1}^{V} D_j(i)^2}} \tag{4.39}$$

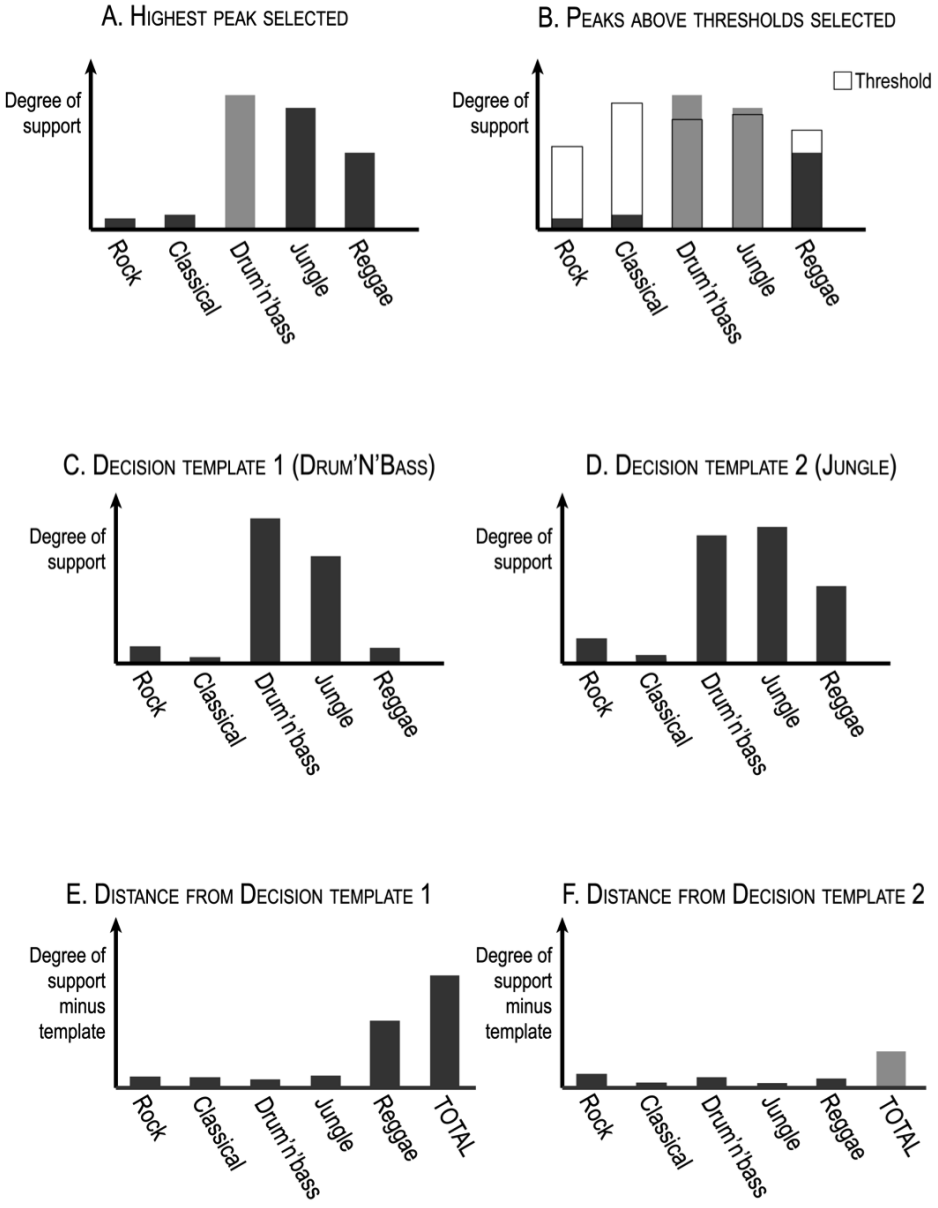


Figure 4.23: Selecting an output label from classification likelihood profiles produced by a probabilistic classification algorithm.

If sufficient training data is available, a Gaussian distribution over the posterior probabilities of classification for the training examples of each class may be estimated instead of a mean profile. These are more correctly compared using the Mahalanobis distance (see equation 4.5).

This method can also be used to combine the output from several classifiers, as the Decision Template is simply extended to contain a posterior probability for each class label from each classifier. Even when based on a single classifier, a decision template can improve the performance of a classification system that outputs continuous degrees of support, as it can help to resolve common confusions where selecting the highest peak is not always correct. For example, Drum and Bass (figure 4.23C) always has a similar degree of support to Jungle music (as they are very similar types of music); however, Jungle (figure 4.23D) can be reliably identified if there is also a high degree of support for Reggae music, which is uncommon for Drum and Bass profiles.

However, classification likelihood profiles estimated from the resubstitution of the classifier's training data do not often well represent the classification profiles of novel examples. Hence, an independent sample of the training data (i.e a portion of the training data not used to train the original classifiers) must be used to estimate each Decision Template. Again it should be noted that, in small tests, the removal of a portion of the training data from each classifier may adversely affect performance, where this would not be so on a larger dataset.

4.4.5 Constrained Regression

Another algorithm for combining posterior probability estimates is *Constrained Regression* (42). This algorithm is based on a weighted linear combination of the posterior probabilities from each system, utilising $V * M$ weights, where V is the number of classes in the dataset and M is the number of classifiers. The degree of support (μ_i) for a class i is calculated as:

$$\mu_i(x) = \sum_{m=1}^{M} w_{im} P_i(x|c_m) \tag{4.40}$$

where w_{im} is the weight for the posterior probability of class i given classifier m and $P_i(x|c_m)$ is the posterior probability of class i returned by classifier m for track x.

The set of weights ($\mathbb{W}_i = \{w_{i1}, w_{i2}, \ldots w_{iM}\}$) are derived by minimising the variance of $\mu_i(x)$. The assumption is made that the estimators are unbiased, and hence the variance of each estimate $P_i(x|c_m)$ is equivalent to its expected error (42). Constrained Regression attempts to find the weights that minimise the variance of μ_i, and is derived by assuming that the classifier's errors in approximating the posterior probabilities are normally distributed with zero mean. The approximation errors are estimated for a training set of classification likelihoods as $1 - P_i(x|c_m)$ for the correct class and $-P_i(x|c_m)$ for the other classes. From the approximation errors we can compute the covariance σ_{mn} of the approximation errors for two classifiers, c_m and c_n. The set of weights used to compute the degree of support for class i ($\mathbb{W}_i$) can then be found by minimising the objective function:

$$J = \sum_{m=1}^{M} \sum_{n=1}^{M} w_{im} w_{in} \sigma_{mn} - \lambda \left(\sum_{m=1}^{M} w_{im} - 1 \right) \tag{4.41}$$

the solution to which is:

$$\mathbb{W}_i = \Sigma_i^{-1} \mathrm{I} \left(\mathrm{I}^T \Sigma_i^{-1} \mathrm{I} \right)^{-1} \tag{4.42}$$

where $\mathbb{W}_i$ is the set of weights for the M classifiers for class i, Σ_i is the estimated covariance of the approximation errors of the classifiers for class i, and I is an M element vector of ones.

4.4.6 Candidate systems

A number of different classification ensemble techniques have been explored for the combination of the classification systems developed in chapters 3 and 4. The sets of classifiers and features used in each ensemble have been chosen for diversity, as the combination of highly correlated systems is unlikely to produce a more accurate ensemble than the best performing candidate system (42). Hence, we attempt to combine classifiers based on the rhythmic and timbral feature sets and, as there was a relatively low correlation between them, we also attempt to

combine timbral classifiers based on mean and covariance summaries of feature vectors and those based on the direct classification of event-level timbral feature vectors.

In the following sections the different variations of classification ensemble attempted are detailed and a designation given to each combination of component classifiers and ensemble type, used to identify each system in the results.

4.4.6.1 Combined feature vectors

Three systems based on concatenated feature vectors, as described in section 4.4.2, are tested. These are:

com-framesMC-rhythmLL4MVCART0.001TFIDF-SMOPoly1 Timbral feature frames mean and covariance vectors concatenated with TFIDF transcriptions of the RCC LL4 mean feature frames.

com-framesMC-rhythmLL4MV-SMOPoly1 Timbral feature frames mean and covariance vectors concatenated with the mean and variance vector of the RCC LL4 mean feature frames.

com-framesMC-rhythmLL4Mean-SMOPoly1 Timbral feature frames mean and covariance vectors concatenated with the mean vector of the RCC LL4 mean feature frames.

4.4.6.2 Simple posterior probability combination algorithms

Three systems based on the *unsupervised* combination of posterior probabilities produced by each classifier, as described in section 4.4.3, are tested. These are:

mean-eventsMC-SMOPoly1-RhythmLL4-MVCART-eventsTFIDF The mean combination of likelihoods from three classifiers: An SMOPoly1 trained on event mean and covariance vectors, an MVCART 0.001 model trained on the full RCC LL4 vectors and an SMOPoly2 trained on the TF IDF vectors produced by an MVCART 0.0005 model trained on event mean vectors.

product-eventsMC-SMOPoly1-RhythmLL4-MVCART-eventsTFIDF The product combination of likelihoods from three classifiers: An SMOPoly1 trained on event mean and covariance vectors, an MVCART 0.001 model trained on the full RCC LL4 vectors and an SMOPoly2 trained on the TF IDF vectors produced by an MVCART 0.0005 model trained on event mean vectors.

voted-eventsMC-SMOPoly1-RhythmLL4-MVCART-eventsTFIDF The voted combination of the final decisions of three classifiers: An SMOPoly1 trained on event mean and covariance vectors, an MVCART 0.001 model trained on the full RCC LL4 vectors and an SMOPoly2 trained on the TF IDF vectors produced by an MVCART 0.0005 model trained on event mean vectors.

4.4.6.3 Trained posterior probability combination algorithms

Three systems based on the *supervised* combination of posterior probabilities produced by each classifier, as described in sections 4.23 and 4.4.5, are tested. These are:

ConstrainedRegress-eventsMC-SMOPoly1-RhythmLL4-MVCART Constrained regression combination of an SMOPoly1 trained on event mean and covariance vectors and an MVCART 0.001 model trained on the full RCC LL4 vectors.

DTemplate-cos-eventsMC-SMOPoly1-RhythmLL4-MVCART Decision template based combination of an SMOPoly1 trained on event mean and covariance vectors and an MVCART 0.001 model trained on the full RCC LL4 vectors, using Cosine comparisons between the template and input likelihood profiles.

DTemplate-euc-eventsMC-SMOPoly1-RhythmLL4-MVCART Decision template based combination of an SMOPoly1 trained on event mean and covariance vectors and an MVCART 0.001 model trained on the full RCC LL4 vectors, using Euclidean distance comparisons between the template and input likelihood profiles.

4.4.7 Ensemble classification results

In this section various combinations of the classification algorithms and feature sets, produced using the techniques outlined above, are evaluated. For the sake of comparison, the component classification systems are also included in the ensemble results, given in table 4.5, and the statistical significance comparisons. The ensembles that managed to improve performance over that of their component systems are highlighted in **bold**.

Table 4.5: Classification accuracy - classifier ensemble models

Algorithm or Ensemble name	Acc	St.dev.
com-framesMC-rhythmLL4Mean SMOPoly1	**63.79**	**0.49**
com-framesMC-rhythmLL4MV SMOPoly1	**64.05**	**0.68**
com-framesMC-rhythmLL4MVCART0.001TFIDF SMOPoly1	59.49	0.42
ConstrainedRegress-eventsMC-SMOPoly1-RhythmLL4-MVCART	61.57	0.51
DTemplate-cos-eventsMC-SMOPoly1-RhythmLL4-MVCART	56.06	0.84
DTemplate-euc-eventsMC-SMOPoly1-RhythmLL4-MVCART	56.06	0.84
mean-eventsMC-SMOPoly1-RhythmLL4-MVCART-eventsTFIDF	46.83	0.33
product-eventsMC-SMOPoly1-RhythmLL4-MVCART-eventsTFIDF	46.37	0.38
voted-eventsMC-SMOPoly1-RhythmLL4-MVCART-eventsTFIDF	61.55	0.54
eventsMeanCovar-SMOPoly1	63.51	0.63
framesMeanCovar-SMOPoly1	63.04	0.45
rhythm-LL4-MVCART0.001	41.75	0.67
rhythm-M-LL4-MVCART0.001	29.84	0.54
rhythm-M-LL4-SMOPoly1	25.86	0.26
rhythm-MV-LL4-MVCART0.001	30.11	0.54
rhythm-MV-LL4-SMOPoly1	25.64	0.25

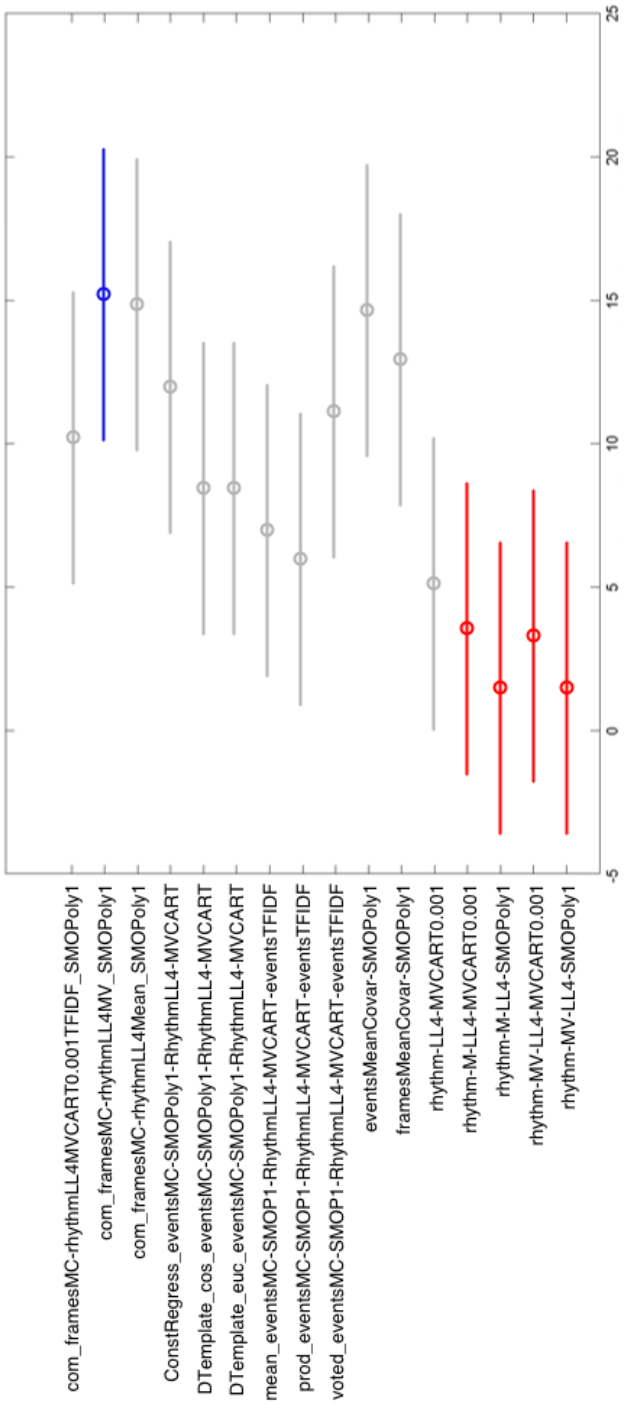


Figure 4.24: Comparison of classifier and feature ensembles using the Friedman test and multiple comparisons.

As shown in the table 4.5 and figure 4.24, the true classification ensembles (i.e. those that combine results from distinct classification algorithms) do not actually provide any gain in performance. In all cases, the combined systems produced a lower performance overall than the best of the component classifiers. However, none of these differences was significant and in the case of the voted ensemble (voted-eventsMC-SMOPoly1-RhythmLL4-MVCART-eventsTFIDF) and constrained regression ensemble (ConstrainedRegress-eventsMC-SMOPoly1-RhythmLL4-MVCART) there is a fairly high chance that their performances are not significantly different from their best performing component classifiers. However, the more sensitive McNemar's matrix indicates that the error-rates of these two ensembles are not significantly different from each other but are significantly different from all other ensembles and individual classifiers.

The Friedman test did not identify any statistically significant differences in the performance of any the different ensemble building approaches. Further, there were no statistically significant differences between the ensembles, including the single vector rhythm classifiers.

During the development of the ensemble systems, early non artist-filtered experiments on a different dataset did show moderate gains in performance for some of the systems. However, these gains were not sustained in any of the artist-filtered experiments reported here. Hence, it must be assumed that such gains were related to the over-fitting of characteristics associated with a particular artist. However, as observed in section 4.3.4.2, the test database is not well separated by rhythmic characteristics, with only two classes that are easily differentiable by rhythm (Electronica & Dance and Hip-Hop). This yields similar patterns of confusion for the rhythmic classifiers to that of timbral classifiers, reducing the 'diversity' of the ensemble.

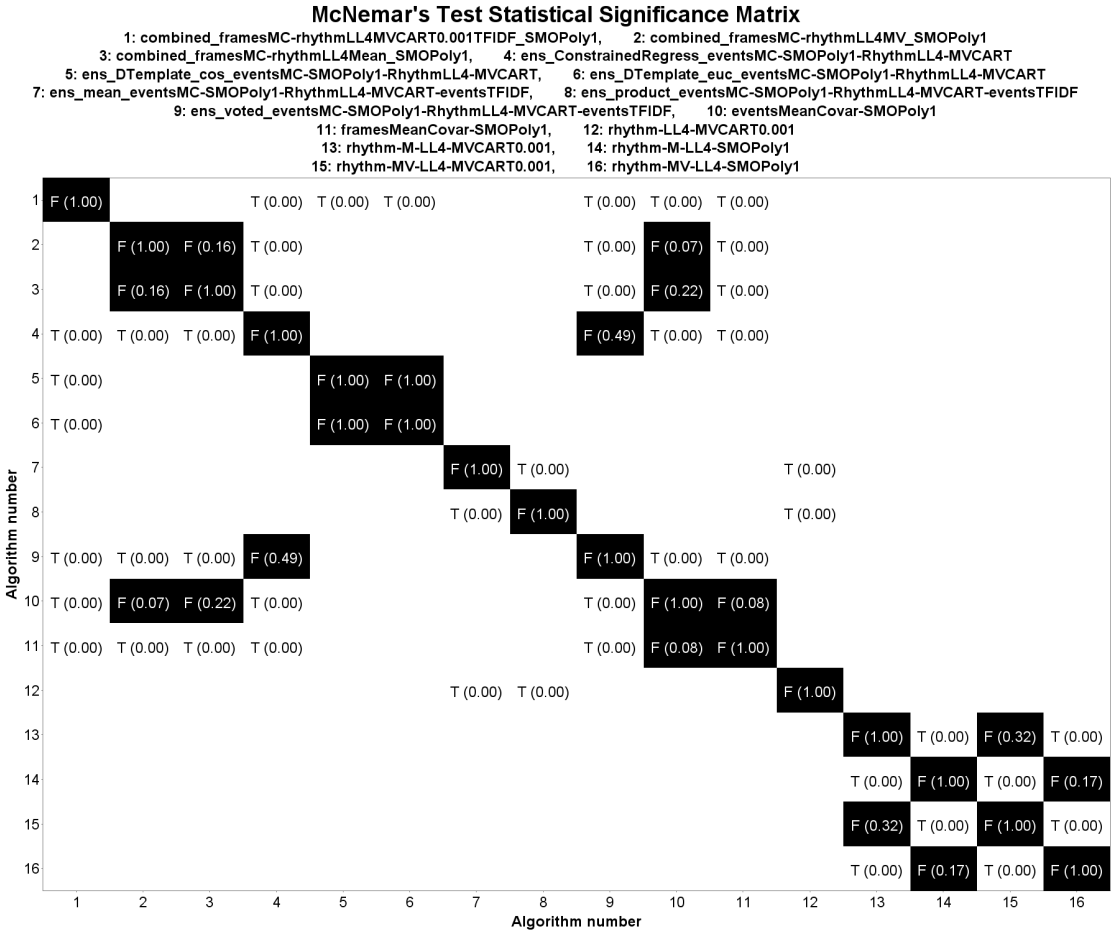


Figure 4.25: Comparison of classifier and feature ensembles using the McNemar's test. Results in white indicate a significant difference in performance at $\alpha = 0.01$. If there is a non-zero P-value these are also marked with a T and the P-value. Grey indicates differences at $\alpha = 0.05$ and are again marked with a T. Black indicates systems that are not significantly different P-value > 0.05 and are marked with an F.

The combined feature vector ensembles cannot be considered 'true' classification ensembles because they are in fact based on the training of a single classifier on a combined vector of features from both the timbral and rhythmic pre-processing front-ends. As stated in section 4.4.2, we cannot easily combine the multiple-vector parameterisations of the audio signals, either with each other or the single vector parameterisations due to varying time resolutions and numbers of vectors. Unfortunately, this prevents us from combining the features used by the best performing rhythm classifiers (based on multiple vectors) with the features used by the best performing timbral classifiers.

Hence our combined feature vector ensembles must be based on the mean or mean and variance vectors, which, as shown in figure 4.21 are significantly outperformed by the multiple vector parameterisations. Further, the timbral features depended on linear models to provide the best performance, significantly outperforming tree based classifiers, as shown in figure 4.7, whereas the best performing RCC-based models were Decision Tree-based classifiers, which significantly outperformed the linear models, as shown in figure 4.19. This leaves us in a difficult situation where we must either attempt to combine our best performing timbral model with the RCC-based features using the worst segmentation and classifier combination, or sacrifice the performance of the timbral model by using a decision tree-based classifier.

One potential alternative is to apply the MVCART transcription (TF or TF IDF) classification approaches to the RCC features and combine the resulting vectors with the timbral mean and covariance vectors. However, these systems performed very poorly in testing on the RCC features and are not reported in section 4.3.4.2. However, the ensemble result is reported here to show the reduction in performance this approach yields over the use of the simple mean or mean and variance vectors.

The combined vector classifiers (based on the combination of mean and covariance of the timbral feature frames and the single vector RCC features) produce the highest performance of any model in these experiments as shown in table 4.5. However, as shown in figure 4.24, this increase in performance is only statistically

significant over the single vector RCC features. Interestingly, the McNemar's matrix shows no statistically significant differences between the error rates of these systems and the events mean and covariance SMO classifier, but not the timbral frames mean and covariance SMO, despite a difference in performance of only 0.75 - 1.01%.

The results in table 4.5 demonstrate little improvement in performance (certainly not a statistically significant improvement), despite an increase in computational complexity. However, as already stated, additional tests should be conducted to determine whether the combined system does in fact yield significantly better performance over datasets containing classes that are largely differentiated by rhythmic characteristics rather than timbral ones (such as many contemporary ballroom and popular dance music genres).

Chapter 5

Audio Music Similarity Estimation

As stated in section 1.1.3, in order to implement *search-by-example* queries for music databases ("find me music similar to X"), recommendation queries ("find me music similar to music I already own") and playlist generation queries ("find me music similar to these N tracks"), we need to be able to estimate the similarity of one track to another. Audio music similarity estimators have significant advantages over other estimators based on user behaviour or human edited metadata catalogues, as they estimate similarity directly on the audio data and can therefore operate on any track for which the audio is available without requiring expensive human intervention.

Existing work in audio music similarity estimation focuses on the estimation of similarity purely on low-level feature values computed from the audio. However, when describing the similarity of two tracks, humans often make reference to (one or more) cultural labels (such as genres or moods), artists or other high-level characteristics of the track. Hence, the dimensions of the similarity spaces in which humans place tracks are culturally defined, and may not well match the spaces that can be produced using distribution-based or simple geometric distance measurements over low-level features computed from the audio signal. In audio similarity estimation, complex patterns in the feature spaces used by existing algorithms, which encode cultural information such as the genre or mood perceived in a track by a listener, are not used to inform the similarity computation.

Rather, it is hoped that comparison of the raw audio features to find tracks with similar *audio* characteristics will yield good estimates of the *musical* similarity and will therefore naturally cluster tracks with similar cultural characteristics.

However, given that important cultural characteristics of audio may be defined by complex patterns in the feature space, results for tracks with certain cultural links to the seed tracks but moderately dissimilar low-level feature profiles may not be returned by an automated estimator, although, they would have been by a human listener. Hence, in section 5.2 we propose that using pattern classification algorithms (such as those developed in section 4), which can learn complex patterns in feature space corresponding to culturally defined characteristics, we can attempt to incorporate knowledge of high-level cultural information into automated similarity estimates.

In the following sections, existing work in audio music similarity estimation is summarised, three methods of using classification algorithms to enhance the performance and utility of music similarity estimators are proposed and compared to existing techniques.

5.1 Existing work

The most powerful music similarity estimators found in the literature are based on the comparison of distributions of the raw timbral feature frames. For example, a set of feature vectors of a particular track may be compared to a probabilistic model of another track to determine the log likelihood of a relationship. In order to create a symmetric distance measure the comparison is repeated in reverse and the results averaged.

Unfortunately, it is impractical to keep all the feature frames for a track in memory because they are sampled at 50 - 100 Hz and require a large amount of memory per track. Hence, methods of directly comparing the *models*, P and Q, of two tracks have been developed. These methods need only retain the model parameters, not the full set of feature vectors. For a Gaussian mixture model, this is the mean vector, covariance matrix and mixture weight of each component of the mixture.

The Kullback-Leibler divergence (KL), or relative entropy, is a distance measurement useful for estimating the distance between probability distributions. In Information Theory, the KL-Divergence is interpreted as the expected extra message-length per datum that must be communicated if a code that is optimal for a given (wrong) distribution Q is used, compared to using a code based on the true distribution P (2). It takes the form:

$$KL(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, dx \tag{5.1}$$

where $p(x)$ and $q(x)$ are probability distributions. For a pair of Gaussian distributions (G_P and G_Q), this has the closed analytic form:

$$D_{KL}(G_P \parallel G_Q) = \frac{1}{4} tr\left(\Sigma_P \Sigma_Q^{-1} - \Sigma_Q \Sigma_P^{-1}\right) + (\mu_A - \mu_B)^T \left(\Sigma_B^{-1} - \Sigma_A^{-1}\right)(\mu_A - \mu_B) \tag{5.2}$$

where Σ represents the covariance matrix and μ the mean vector of each Gaussian distribution (G) and tr represents the *trace* operation. As the KL divergence is also non-symmetric, i.e. $D_{KL}(G_P \parallel G_Q) \neq D_{KL}(G_Q \parallel G_P)$, (hence the term divergence, rather than distance) it must be computed in both directions and averaged to produce a useful similarity metric.

Unfortunately, there is no closed form of the KL-Divergence for mixtures of Gaussian distributions. Hence, two alternative approaches to comparison have been proposed: Monte Carlo sampling and the Earth Mover's distance. Both approaches allow the comparison of mixture models with an arbitrary (variable) number of components, but may be considered highly computationally intensive.

Monte Carlo sampling compares the distributions by generating a large number of points from one of the distributions and estimating their log likelihood given the second distribution (again this process must be reversed and the results average to produce a symmetric distance measure) (2).

The Earth Mover's distance (EMD) visualises each mixture of Gaussian distributions as a mass of earth distributed on the ground according to the mixture distribution and attempts to measure the least amount of 'work' (probability

'mass' times distance) that would have to be performed to convert one distribution into the other (47). To achieve this, the distance computation can be characterised as a linear programming problem, based on a distance matrix computed between each component of one mixture with each component of the other. The solution to this linear programming problem provides the optimal *flow* of earth from each mixture component to its counterparts in the other model and the total *flow* required to convert one distribution into the other. The use of the EMD for the comparison of MFCC/GMM based signatures of music tracks was introduced by Logan and Saloman (47).

Several authors have evaluated the performance of GMM based signatures of timbral audio features compared using Monte Carlo sampling and/or the EMD (2) (43) (47) (55). It should be noted that both techniques are highly computationally intensive, particularly the use of Monte Carlo sampling. Further, in (50), Mandel and Ellis propose a system based on the comparison of simple Gaussian distributions with the KL-Divergence. Both (2) and (55) provide experimental evidence that the performance of this technique is at least equal to that of GMM based approaches, making their extreme additional computational cost impossible to justify. This lack of additional discriminative power for the use of mixture distributions is at odds with research on many other indexing problems (2).

Aucouturier (2) explores the reasons for the failure of mixture distributions and features describing the dynamics of the feature space to improve performance. Aucouturier concludes that the representation of *polyphonic* timbre is too complex to achieve without also emulating the human ability to separate the audio stream into a number of distinct sources and direct our attention to each source individually. The inability of the standard MFCC-based pre-processing/feature extraction system to achieve this may lead to a 'glass-ceiling' on the performance of timbral music similarity estimators based upon it.

Further, Aucouturier also demonstrates that many attributes of music tracks perceived by users are not well correlated with timbral similarity measurements, in the context of an automatic tagging system that suggests attributes for a track based on the common attributes amongst its nearest neighbours (according to the timbral similarity metric). However, through the use of historical/cultural knowledge Aucouturier significantly improves the performance of the system, for

example by leveraging relationships between attributes such as 'Rock music' and 'Guitar' or 'Rag time' and '1930s'. This cultural/historical information is encoded in the form of a Decision Tree trained using the co-occurence of attributes in textual metadata descriptions of tracks and applied to the attributes suggested by the timbral similarity metric, yielding a significant improvement in performance.

There are few other examples of the use of a historical or cultural context in music similarity estimation. However, this is at the heart of an audio genre classifier, which must place a music track within the context of its genre. The 'sound' of a genre (that which discriminates it from other music genres) will have been defined by a culture and must therefore be learnt by experience (or in the case of a genre classifier, from a reference dataset). Owing to the lack of such a set of cultural references, we propose that it is difficult for conventional approaches to audio music similarity estimation, detailed above, to be able to fully emulate human music similarity judgements. This is due to the fact that cultural aspects of the music may override the audio. Conventional techniques can return tracks that have similar cultural attributes, but achieve this by seeking similar audio characteristics. They are therefore likely to miss many other perceptually related items through an inability to use attributes measured at a higher perceptual level (e.g. the genre, mood, style) rather than the low-level audio characteristics.

In (43), Levy and Sandler propose two 'lightweight' timbral similarity measures which attempt to provide similar performance to the KL Divergence between single Gaussian distributions or Monte Carlo Sampling of Gaussian Mixture Models but at a lower computational cost. The first approach is based on the vector quantisation of the feature vectors using a 16 x 16 Self-Organising Map (SOM). The sequence of MFCC vectors can then be converted into a sequence of map indices and the histogram of their occurrences computed. By treating this histogram as a probability distribution, the KL-divergence can again be used to compare the resulting features. Levy and Sandler report a similar but lower level of performance for this technique in comparison to the conventional KL-Divergence technique and the comparison of GMMs with Monte Carlo sampling. Further, they report that the distance computation requires approximately three times the runtime of the conventional KL Divergence calculation.

The second approach is based on the replacement of the KL-Divergence between full covariance Gaussian distributions with the Mahalanobis distance between diagonal covariance Gaussian distributions. The Mahalanobis distance is modified to take into account the the variance of each feature dimension in the dataset, and is given by:

$$D_M(G_P \parallel G_Q) = (\mu_P - \mu_Q)^T \Sigma_\mu^{-1} (\mu_P - \mu_Q) + (\Sigma_P - \Sigma_Q)^T \Sigma_\Sigma^{-1} (\Sigma_P - \Sigma_Q) \quad (5.3)$$

Where, in this case, G represents a diagonal covariance Gaussian distribution, Σ and μ represent the distribution's diagonal covariance matrix and mean vector and Σ_μ and Σ_Σ represent the variances of the feature means and variances respectively (43). Levy and Sandler again report a lower level of performance for the Mahalanobis distance compared to the full covariance KL-Divergence and GMMs with Monte Carlo sampling. However, they also report a very high computational saving for the use of the Mahalanobis distance, resulting in a 50-fold reduction in the search times compared to the KL-Divergence (1 ms vs. 50 ms per comparison) and a 2000-fold reduction compared to the comparison of GMMs with Monte Carlo sampling (1 ms vs. 2000 ms).

The use of the Mahalanobis distance to implement the comparison of the feature distributions demonstrates two key advantages over the better performing KL-Divergence and GMM based systems, which offset its lower performance. Firstly, the significantly improved runtime will help the system scale to larger collections as (based on Levy and Sandler's runtime measurements): 1,000 comparisons may be performed per second compared to 20 per second for the KL-Divergence. However, this still does not allow the technique to scale to very large collections: a search of a 1,000,000 song catalogue would take approximately 1,000 seconds. The second advantage of the Mahalanobis distance is that it provides a metric similarity space that may be indexed using techniques such as VP-Trees or M-Trees to provide significantly faster searches (16).

In the following sections, we introduce a number of techniques that use the cultural information learnt by a genre classification model to perform timbral music similarity estimates. These approaches use the classification models in a

number of different ways in order to achieve dimensionality reduction, faster comparisons with lower order distance measures, and different or improved similarity estimates. Each of the techniques proposed is compared to the KL-Divergence between Gaussian distributions and all achieve significant computational savings over this technique. Hence, systems based on these techniques will be able to scale to much larger collections than the KL-Divergence. This is particularly true when it is considered that all of the techniques produce metric or very close to metric distance spacesand therefore may be optimised with indexing structures, such as VP-trees, MVP-trees or M-trees (16), that are not applicable to the KL-divergence based search indices.

5.2 Using genre classifiers to enhance content-based music search

As stated in the preceding sections, existing systems of music similarity estimation assume a correlation between probabilistic or geometric distance measures (such as the Euclidean distance or KL-Divergence) calculated over features computed from an audio track, and the human perception of music similarity. Complex non-linear machine learning based estimators, which could learn cultural associations between timbres and perceptual attributes of the music such as genre or mood, would be quite likely to outperform these techniques, but would have to be trained on a reference database containing many paired examples with labelled similarities. Unfortunately, as stated in section 2.5.2, this data is prohibitively expense to collect from humans and it is therefore unlikely that robust, general algorithms for audio music similarity estimation will be built in this manner.

In this section we make the alternative assumption that the human perception of music similarity is correlated with the division of music into culturally defined categories, such as genres. This assumed relationship is used to conduct automated pseudo-objective evaluations of the performance of audio music similarity estimators (27) (47) (55) (71). Pampalk (55) and the results of the MIREX

2006 Audio Similarity task (27) provide experimental evidence that human evaluation of the performance of automated music similarity estimators are correlated with the neighbourhood clustering statistics (the average proportion of the top N matches for all possible queries which belong to the same artist, album or genre as the query, where N is a small constant ranging from 5 - 50).

Hence, it is proposed that the use of genre classification models trained on audio feature profiles will be able to provide useful information for music similarity estimation. This would be achieved by filtering the low-level audio characteristics through the complex patterns learnt by the classification model to represent the cultural division of the tracks into genres. If the model estimation is successful, we could reasonably expect to see fewer results caused by the matching of superficially similar audio (i.e. the pairing could be perceived to have a high 'Aha' factor (3) as described in section 2.3). We can also expect to find less emphasis on matches with highly similar audio characteristics due to the partial abstraction of information in the audio features. The similarity estimates produced by this approach will be somewhat correlated with the *audio* similarity of two tracks. However, it also proposed that we will be able to construct better similarity estimation techniques using this approach, or at least enhance the performance of existing approaches.

Several of the techniques described in the following sections were first introduced in (70).

5.2.1 Culturally informed projections

The results of the classification experiments show that the best performing *genre* classifiers compare features in a linear subspace (i.e. the first order polynomial SMO and linear discriminant analysis (LDA)). Hence, we can define a simple system that filters the audio features through complex culturally defined patterns optimised to separate them according to genre. The resulting feature space should represent both an abstracted representation of the *audio* content and a strong indication of the cultural (genre) profile of a track. A simple example of such a system might be the linear projection of a single vector of features ($\vec{f}(x) = \{f_1(x), f_2(x), \ldots f_Y\}$) for a track x, using an LDA transform, into a space

($\vec{f'}(x) = \{f'_1(x), \ldots f'_{V-1}\}$) that provides the maximal separation between genre classes in the audio.

The LDA transformation matrix $\vec{w}$ is trained as described in section 4.2.1. The features are projected into the new space ($\vec{f'}(x)$) as follows:

$$\vec{f'}(x) = \vec{f}(x) \cdot \vec{w} \tag{5.4}$$

These transformed features may then be compared with a distance metric such as the Euclidean distance or (if using pre-processing techniques based on multiple vectors of features) distribution based distance measures, such as the KL-divergence, to estimate the similarity between tracks. The dimensionality of the feature space will be reduced to $N-1$, where N is the number of classes that the LDA transform was trained to separate.

As the LDA transforms of multiple vector feature representations (such as those produced by event-level segmentation) are too time-consuming to compute, we would compute the LDA of the mean and covariance summary of the features. However, this prevents us from applying the KL Divergence measure which requires a distribution to work with. On the other hand the transform does produce a transform space with uncorrelated (independent) dimensions, in which we can apply the Euclidean distance. This will produce a metric similarity space (the triangular inequality will hold), and therefore there are a number of techniques we could use to index the similarity space in order to scale the search index. The projection also represents a huge reduction in the size of the data, as an 11-class genre classifier will produce just 10 dimensions from the full mean and (flattened) covariance vector which originally had $\left(\frac{((38*38)-38)}{2} + (2*38) = 779\right)$ dimensions. Hence, this transformation can be expected to significantly speed up the time it takes to perform simple distance measure calculations and enable the sub-linear scaling of search times with index size (as the distnace space produced is metric). These are significant advantages over the computationally intensive KL divergence based estimator.

5.2.2 Classification likelihoods based similarity estimation

As already stated, humans often make reference to cultural labels, artists or other high-level characteristics of a track when describing it or discussing its

similarity to another track. Hence, it is proposed that the classification likelihood profile (posterior probabilities of classification) of a track may represent useful information for use in the estimation of its similarity to another track. If we were to compare text-based descriptions of two tracks and find positive references to the same genres of music, we would reasonably expect those tracks to be more similar to each other than tracks chosen at random. We speculate that the same comparison can be made between two tracks posterior likelihood profiles from a genre classifier to estimate their broad *musical* similarity.

For simplicity, we describe a system based on a single classifier; however, it is simple to extend this technique to multiple classifiers, multiple label sets (genre, artist or mood) and feature sets/dimensions of similarity by concatenation of the likelihood matrices, or by early integration of the likelihoods (for homogenous label sets), using a constrained regression, geometric mean or weighted linear combination model combiner. The system could potentially also be integrated into a conventional similarity estimation systems, based on geometric distance measurements between low-level feature values, by concatenation of the of the feature vectors.

Let $\mathbb{P}(x|c) = \ P_1(x|c), P_2(x|c), \ldots, P_V(x|c)\}$ be the profile for track x, where $P_i(x|c)$ is the posterior probability, given classifier c, that example x belongs to class i, and $\sum_{i=1}^{V} P_i(x|c) = 1$, which ensures that likelihoods returned are in the range [0:1]. The distance, $Dist_{A,B}$, between two examples, A and B can be estimated using the Euclidean distance:

$$Dist_{euc}(A,B) = \sqrt{\sum_{i=1}^{V}(P_i(A|c) - P_i(B|c))^2} \tag{5.5}$$

between their profiles, $P(A|c)$ and $P(B|c)$, or as the Cosine distance:

$$Dist_{cos}(A,B) = \frac{\sum_{i=1}^{V} P_i(A|c)P_i(B|c)}{\sqrt{\sum_{i=1}^{V} P_i(A|c)^2}\sqrt{\sum_{i=1}^{V} P_i(B|c)^2}} \tag{5.6}$$

In our tests the Cosine distance has always performed better than the Euclidean distance.

This approach is somewhat similar to the 'anchor space' described by Berenzweig, Ellis and Lawrence (10), where 'clouds' of classification likelihood profiles, for individual feature vectors in a song, are compared with the KL divergence, Earth Mover's Distance (EMD) or Euclidean distance between sets of centroids. However, the comparison of the centroids of the mixture of distributions with the KL divergence or EMD is likely to require significantly more processing and memory to implement than the comparison of simple track classification likelihood profiles with geometric distance measures, such as the Cosine or Euclidean distances.

5.2.2.1 Non-trivial classification likelihood estimates

Reducing the likelihood profiles to a simple one bit indicator function will reduce the similarity space produced to an V cell table with each track in a cell being maximally similar to all other tracks in that cell, where V is the number of classes in the dataset used to train the classifier. Hence, classification algorithms that return single bit likelihood profiles (i.e, a binary indicator function that is 1 for the chosen classification and zero for all other classes, known as a Kronecker Delta function) or very poorly distributed likelihood profiles will yield extremely coarse similarity estimates.

Further, due to the additional information in full classification likelihood profiles, we can expect the performance of a similarity estimator based on them to be superior to simply recommending tracks from the same genre, particularly by mitigating against likelihood profiles that do not correctly assign the classification label, but do achieve similar profiles to other tracks of the correct genre.

5.2.2.2 Runtime performance

In (71) we introduced a system based on this technique, using 10 class genre classification likelihood profiles. The system performed reasonably well, returning a high proportion of tracks in the top N search result lists from the same genre, artist and album as the query.

It is proposed, in (71), that in order to implement an industrial scale music search engine, the indexing techniques used must be able to return results on collections of around two million tracks in under a second. The search runtimes for this system are known to scale linearly. Hence, they were measured on the test collection and used to predict search runtimes on a collection size of 2 million tracks. Based on the 10-class genre likelihoods profiles it was estimated that searches of a two million track collection would be performed in approximately 410 ms. The hardware used to make these estimates was a 2GHz AMD Turion 64 CPU, running a Java 1.5 64-bit server Virtual Machine (VM).

These search times compare very favourably with distribution-based distance metrics such as the Monte Carlo sampling approximation used to compare Gaussian Mixture Models of features, described in (56). The CPU performance optimised Monte Carlo system used in (56) calculates 15,554 distances in 20.98 seconds. This can be extrapolated to two million distance calculations, yielding a runtime of 2697.61 seconds or 6580 times slower than the classification likelihoods-based model.

5.2.3 Leveraging clustering patterns learnt by a classifier

As described in section 4.2.5.2, an improved classification tool can be based on the MVCART classification model trained by converting the sequence of leaf node identifiers returned for event-level feature vectors into Term Frequency (TF) or Term Frequency Inverse Document Frequency (TF IDF) profiles and training a another classification model on them. It is proposed that these TF or TF IDF vectors may also be used to implement efficient audio music similarity estimation.

In simple vector-model based text search engines, vectors of TF and TF IDF profiles are compared using the Cosine distance to implement query-by-example searches. Alternatively, keyword searches may be implemented by the construction of a pseudo-profile for the keywords and comparison to the document profiles indexed by the system. Efficient sparse vector techniques can be used to provide excellent search times over the collection. The Cosine distance is given by:

$$\theta\left(Q, D_i\right) = \frac{W_Q \cdot W_{D_i}}{|W_Q||W_{D_i}|} = \frac{\sum_{j=0}^{L} W_{Q,j} W_{i,j}}{\sqrt{\sum_{j=0}^{L} W_{Q,j}^2}\sqrt{\sum_{j=0}^{L} W_{i,j}^2}} \tag{5.7}$$

where L represents the total number of unique terms, Q represents the query document, D_i represents a potential result document and $W_{Q,j}$ represents the TF or TF IDF weight for the j-th term in the query document.

A vector-model search can be implemented for the MVCART leaf node transcriptions of event-level feature vectors using the same techniques. It is thought that the clustering of event-level feature vectors produced as part of the MVCART training procedure will also be useful for the comparison of tracks for similarity. The decomposition of the event-level vectors for genre classification is expected to divide the vectors into groups (leaf nodes) with average characteristics that help to identify a single genre or small group of genres. The nature of the model ensures that these groups contain vectors with similar audio characteristics. However, they also retain a significantly lower granularity of audio representation than the classification profiles that they produce. Hence, they should able to provide music similarity searches with both a reasonable degree of audio similarity and higher-level cultural (genre) similarity.

Event-level vectors are preferred to frame-level features for the training of the MVCART model and production of TF vectors, as they contain significantly more information for audio representation than frames corresponding to only 23 ms of audio, and produce models and dictionaries with a convenient number of distinct terms. By contrast, frame-level models will produce significantly larger models/dictionaries. The reduced length of the transcriptions (e.g. an average of 400 vectors per 120 seconds of audio) can also be far more efficiently processed than the much longer frame level transcriptions (e.g. over 10,000 frames per 120 seconds of audio).

Another useful characteristic of the leaf-node transcriptions is that they may be used to query the collection based on any reasonable length segment of an audio track or continuous audio stream without the re-extraction of features over that segment. The relevant segment of the transcription of the audio track may be rapidly summarised as a term frequency vector and used to query the collection. The TF weighting of terms in the segment implicitly normalises the

weights for the length of the piece. Hence the length of query clip and each potential search result in the collection should not influence the results significantly.

There are several alternative strategies for the indexing of the leaf-node transcriptions that could be used to implement query-by-example searches over the collection, including the production of term weight vectors for N-grams extracted from the symbol sequences (allowing us to leverage sequential information encoded in the transcriptions), Latent Semantic Indexing (LSI) of the term weight vectors (which may produce a more compact search model with reduced noise) or discrete Hidden Markov or Markov chain modelling of each track transcription.

5.2.3.1 Optimising the index for fast searches

The exhaustive comparison of all term weight profiles to a seed profile using the Cosine distance may be relatively quickly performed using sparse vector techniques. However, significantly faster comparisons may be performed through more intelligent indexing of the term weight profiles.

By maintaining a hash of track identifiers to the list of leaf node identifiers corresponding to a track, and a second hash of leaf node identifiers to relevant track identifiers, we are able to conduct a first pass search in order to identify a subset of the collection on which to perform the full Cosine distance comparison. For each seed track, the list of relevant leaf nodes is retrieved. For each leaf node in this list the list of relevant tracks is retrieved, and for each track a count maintained of the number of leaf nodes (relevant to the query) within which it appears. The tracks with non-zero counts are a subset of the collection guaranteed to produce non-zero cosine distance scores.

This approach may be further optimised by setting a threshold (α) of the term weight below which a track is not included in a leaf node's membership list, or the leaf node included in the track's term list. This eliminates the least significant term weights from consideration in the first-pass search and can significantly reduce the size of the collection subset returned by it. By also setting a minimum number of terms per track, empty subsets may be avoided.

Table 5.1: Percentage of identical result lists (average over 2,500 queries)

Search depth	% result lists identical to full search
1 result:	100.0%
3 results:	99.6%
5 results:	99.2%
10 results:	98.9%
20 results:	98.4%

Table 5.2: Percentage overlap between result lists (average over 2,500 queries)

Search depth	% overlap between optimised and full result lists
1 result:	100.0%
3 results:	99.8%
5 results:	99.9%
10 results:	99.7%
20 results:	99.7%

Further, the subset returned may be sorted by the counts for each track and a smaller set of size β (consisting of the β tracks with the highest counts) returned for full comparison. Finally, the subset returned by the first pass search is compared to the seed using the full Cosine distance in order to rank the results.

The results yielded by the optimised index search have been compared to the full Cosine search on a collection of 25,084 tracks, using 2,500 randomly chosen queries. The results of this comparison are given in tables 5.1 and 5.2.

As Tables 5.1 and 5.2 demonstrate, the results yielded by the optimised search deviate very little from the full search results. Hence, such optimisation is very unlikely to have any impact on the perceived performance of the search model.

Figure 5.1 plots the average search time (in milliseconds) against index size (number of tracks) for the standard Cosine search over TF/IDF weightings, the optimised Cosine search over TF/IDF weightings and a KL-Divergence based

search. The times reported were averaged over 1000 random queries selected from the index. The KL Divergence-based search has been optimised as much as possible by pre-computing a number of terms in the KL-Divergence equation for each track. However, both the standard Cosine search and optimised Cosine search return results in significantly lower runtimes. E.g the KL-Divergence requires 2040 ms to satisfy a search over 20,000 tracks, while the standard Cosine search takes 179 ms and the optimised Cosine search requires just 39.7 ms.

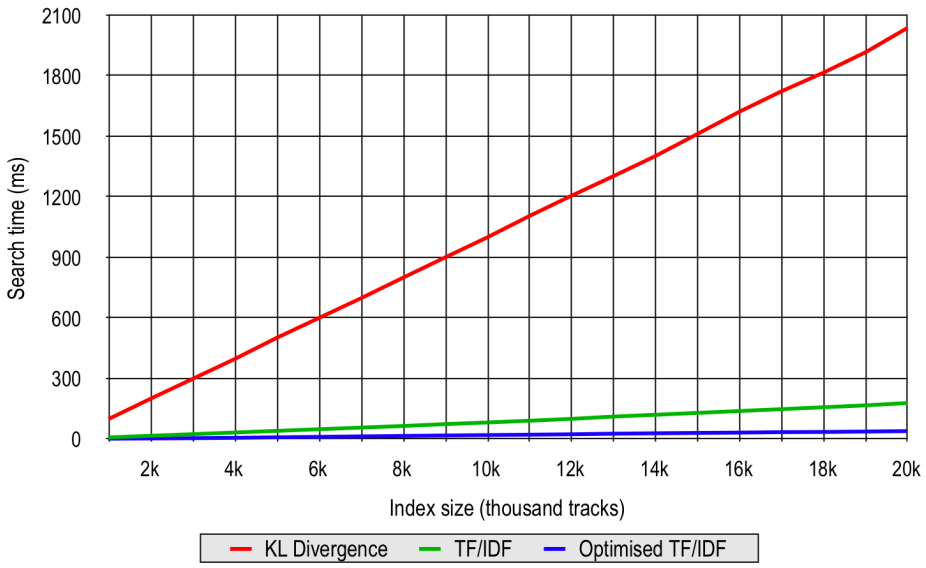


Figure 5.1: Comparison of average search times vs. index size for KL-Divergence and Cosine searches over TF/IDF weightings

The search times of the optimised index are dominated by the full comparison (Cosine distance) of the 750 (β) tracks returned by the first pass search. Hence, the β parameter allows search times to be adjusted simply and at runtime, while the combination of the α and β parameters controls the effect of the search time optimisation on the results.

As the complexity of the search can be throttled using the β parameter, this indexing strategy has significant advantages over distance measure based search strategies. Further, the target search times can actually be specified in place of

the beta parameter so that the system can continue to evaluate the retrieved set at increasing depth until the target search time is reached or exceeded.

A final search time optimisation, which has not yet been explored, is the partitioning of the index into a series of sub-indicies. This might be accomplished through the clustering of term weight profiles into the desired number of index segments, ensuring that each track is in the same index segment as its likely results. Such a division might be possible because of the spare nature of the index, although it is likely that some degree of overlap between index segments will be required to ensure that queries on the border of clusters are not negatively affected by the partitioning. In order to query the index the query would be mapped to the relevant index segment(s) using a hash-map and only those segments searched, significantly reducing the number of tracks that must be considered in the first pass search. Such a partitioning scheme might allow the search time to scale sub-linearly with the index size and would facilitate distribution of large or frequently accessed indices over a number of separate servers.

5.3 Combining information from multiple search systems

As was demonstrated for audio music classification, the simplest way to combine systems based on different feature sets is to concatenate the feature vectors. This allows us to combine any feature set that produces a single summary feature vector per track, including the classification likelihoods based systems, MVCART transcription based systems, LDA/Euclidean distance based systems and standard Euclidean distance based search systems. The KL divergence based systems can only use combined feature sets if each set to be combined has the same segmentation so that the individual feature vectors can be combined prior to estimating the distributions used in the comparison.

There are number of alternative approaches that may be used to combine the outputs of different search systems, including weighted averaging, the product

of the similarity scores and the rank normalised product of the similarity scores. Both the weighted sum and product combinations require that the similarity estimates produced by the systems to be combined have the same range and similar distributions. However, the rank normalised product can be applied to significantly different systems. Any of these approaches may be applied to either similarity scores or distance scores, and hence this may be considered a minor implementation detail.

The weighted average of L search systems is defined as:

$$S_{comb}(x) = \sum_{i=1}^{L} w_i S_i(x) \tag{5.8}$$

where $S_i(x)$ represents the similarity estimate from the i-th search system for example x and w_i represents its weight. The weights for each search system must be established by experimentation.

The product combination is defined as:

$$S_{comb}(x) = \sqrt[L]{\prod_{i=1}^{L} S_i(x)} \tag{5.9}$$

However, as we are interested only in the results final rank amongst all the tracks in the collection, we can drop the root and define the product combination as:

$$S_{comb}(x) = \prod_{i=1}^{L} S_i(x), \tag{5.10}$$

where L is the number of search systems to be combined.

The rank normalised product may be defined as:

$$S_{comb}(x) = 1 - \frac{\sqrt[L]{\prod_{i=1}^{L} \text{rank}\left(S_i(x)\right)}}{N} \tag{5.11}$$

where $\text{rank}\left(S_i\right)$ is a function which replaces the similarity score S_i with its integer rank amongst the similarity scores for all tracks in the collection from that particular search system and N represents the collection size. Again, as we are only

interested in the final rank of the combined similarity score ($S_{comb}(x)$) amongst the similarity scores for all tracks in the collection, we can define this combination as:

$$S_{comb}(x) = N^L - \prod_{i=1}^{L} \text{rank}\left(S_i(x)\right) \tag{5.12}$$

In practice we can just use the $\prod_{i=1}^{L} \text{rank}\left(S_i(x)\right)$ term and reverse the sort order (as it is a distance score).

One final optimisation relevant to the product combination is the use of each search system as a filter. If the similarity matrix can be considered sparse (i.e. if we are not guaranteed to find a non-zero similarity between all examples), we only need to conduct similarity searches with subsequent search indexes over the subset of tracks with non-zero similarities from the the other systems. This is due to the fact that if any system returns a zero similarity, the overall similarity will be zero. When using the rank normalised product combination, tracks with $S_i(x) = 0$ in the first index searched may also be ignored when searching other indexes.

In particular, the MVCART transcription based indexes are sparse in that zero similarities will be returned for tracks that do not share any leaf node symbols. However, other search indices can be made sparse by setting a threshold of the similarity score or a retrieved set size (in a similar fashion to the β parameter used to control the first pass search in the optimised MVCART transcription search).

Because of these optimisations, the product and rank normalised product combinations are preferable to the weighted linear combination, as they can yield significantly lower search times for the combined search system, without compromising the quality of the results returned. Further, the product and rank normalised product combinations do not require the setting of weights for the combination, which must be hand-tuned for the weighted linear combination.

5.4 Evaluation

5.4.1 Candidate systems

In the following section results are reported for systems based on combinations of the feature sets, segmentations, search strategies and methods of combining multiple search systems described in this chapter. These include:

5.4.1.1 Feature sets

The feature sets considered in these experiments include:

Rhythm Cepstral Coefficients (RCCs) Computed with either an initially linear then logarithmically scaled filterbank with transition frequency of 4 Hz or a fully logarithmically scaled filterbank. These are referred to as LL4 and LL0 respectively.

Timbral feature set Composed of MFCC and the statistical spectrum descriptors described in section 3.2.2.

5.4.1.2 Timbral feature stream segmentations

The segmentations of the timbral features considered include:

whole files Single Gaussian (mean and covariance) summaries of the feature frames.

event-based segmentation Mean vector summaries of the feature frames corresponding to each segment.

event-based segmentation, whole files Mean vector summaries of the feature frames corresponding to each segment with a single Gaussian (mean and covariance) summary of the event feature frames.

5.4.1.3 Rhythm feature stream segmentations

The segmentations of the RCC features considered include:

50% overlapped, 9 second segments No summary required.

50% overlapped, 9 second segments, whole files Mean vector summaries of the feature frames.

5.4.1.4 Search strategies

The search strategies considered include:

KL Divergence The Kullback-Leibler divergence computed between Single Gaussian distributions of feature vectors generated by individual timbral feature frames or timbral event vectors. The RCC features generate too few vectors to allow comparison using the KL divergence.

Euclidean distance The Euclidean distance computed between single feature vectors.

LDA-based projection and Euclidean distance Fishers criterion Linear Discriminant Analysis-based transformation of single feature vectors and comparison in the transform space using the Euclidean distance.

Classification likelihoods and Euclidean distance Comparison of genre classification likelihood vectors (posterior probability profiles) with the Euclidean distance.

Classification likelihoods and Cosine distance Comparison of genre classification likelihood vectors with the Cosine distance.

MVCART transcription and TF weighting Conversion of timbral event feature vectors or RCC vectors into MVCART leaf node sequences and Term Frequency (TF) summaries. Comparisons are performed with the Cosine distance.

MVCART transcription and TF IDF weighting Conversion of timbral event feature vectors or RCC vectors into MVCART leaf node sequences and Term Frequency Inverse Document Frequency (TF IDF) summaries. Comparisons are performed with the Cosine distance.

5.4.1.5 Combinations of search systems

The methods of combining search systems considered include:

Combined feature vectors The simple concatenation of the feature vectors.

Weighted linear combination The weighted sum of the similarity scores. Weights have been established through limited experimentation.

Product The product or harmonic mean of the similarity scores.

Rank normalised product The product or harmonic mean of the ranks of the similarity scores from each search system over the set of all tracks in the collection.

5.4.2 Automated statistical evaluation

Because of the considerable expense and complexity of large-scale human evaluations, several authors have presented evaluations of their audio search systems based on statistics of the number of examples bearing the same label (genre, artist or album) amongst the N most similar examples to each song (*neighbourhood clustering* (47)), or on the distance between examples bearing the same labels, normalised by the distance between all examples (*label distances* (71)). In this work, the neighbourhood clustering statistics are reported as it is hoped they provide the strongest indications of the performance of each system at *recommendation*, based on the assumption that results from the same album, artist or genre and likely to be relevant recommendations. In contrast, label distances require all examples of a particular class to be tightly clustered in order to yield good results. Within varied genres of music this may not be as appropriate a statistic as neighbourhood clustering for estimating recommendation performance.

These automated statistical evaluations are sometimes referred to as pseudo-objective statistics. The term pseudo-objective is used as the statistic most appropriate to the evaluation of music recommendation performance, genre neighbourhood clustering, is based on objective measurements of the agreement between the tracks retrieved by a system and a set of subjectively applied genre labellings.

The results from MIREX 2006, an annual evaluation Music Information Retrieval techniques hosted by the International Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL) at the University of Illinois, indicated that the Neighbourhood clustering metrics were correlated with the large-scale subjective human evaluation, producing the same ranking of five similarity estimation techniques from different sources (27) at 5, 10, 20 and 50 results. Hence, we report the genre, artist and album clustering at 20 results.

The distance space may also be evaluated both on how often the triangular inequality is satisifed (indicating that it is either a metric or non-metric space). In a metric space $\mathbb{S}$, with distance metric $dist$, the triangular inequality is defined as:

$$dist(x,z) \leq dist(x,y) + dist(y,z) \quad \text{for all } x,y,z \text{ in } M \tag{5.13}$$

I.e. the distance between any two examples (x and z) must be less than or equal to the sum of the distances between each example and a third point(y).

In the experiments reported here, 100,000 randomly chosen 'triangles' are tested using the same set for each distance matrix. As indicated in section 5.1, it is very important to determine whether a particular similarity estimator produces a metric search space as metric spaces may be indexed using a number of different procedures, such as VP-Trees or M-Trees (16), which allow search times to scale sub-linearly with database size.

Finally, the degree to which the system is affected by '*hubs*' (tracks that are similar to many other tracks) and '*orphans*' (tracks that are never similar to other tracks) (55) should be measured. The presence of hubs in the search space can lead to an extremely poor user experience within a music recommendation system, as the same, potentially irrelevant track may be returned as a recommendation for a great many tracks (2). On the other hand, *orphaned* tracks are

essentially 'lost' in the recommendation system and will never be recommended for any query. It is possible that there are some unique tracks in the collection, which would not be relevant search results for any other track in the collection at a particular search depth (number of results). However, this is quite unlikely, so a significant number of orphans may indicate a poorly performing recommendation system/music similarity estimator. In order to highlight the effect of hubs on a distance space produced by one of our search systems we, report the size of the worst hub in the distance matrix (the track appearing in the results for the most other tracks at a particular search depth).

5.4.3 Experimental set-up

The same dataset described in table 4.1 is re-used for the search experiments. As several of our approaches are based on a trained classifier, this dataset is split 50:50 into training and test sets. Hence, the genre model is trained on a different subset of the database and will not gain advantage by being exposed to the genre metadata for any track it is evaluated on.

The difficulty of automated statistical evaluation of music similarity measures on small databases is well-known in the Music Information Retrieval community and several authors have identified the potential for a significant bias to be introduced into evaluations of this type, either by over-fitting the production, acoustic or vocal characteristics of an artist (56) or album (41). To avoid such bias, the partitioning of the database into a training and test set has been *artist-filtered*, as described in section 2.5.1, ensuring that all tracks belonging to an artist are placed in either the test or training set and not split across them. It should be noted that this often results in the least favourable split of any evaluation database, producing the least diversity in the training set. This leads to a high potential for over-fitting during model training and a low potential for that over-fitting to inflate the evaluation statistics. Hence, the results produced may be significantly lower than those reported in studies based on less rigorous evaluation procedures.

5.4.4 Automated statistical evaluation results

The full results of the automated pseudo-objective evaluation are given in table B.1 in appendix B. Excerpts are provided in this section for ease of reference.

To put the results into context, the well known KL-Divergence based search can be used as baseline against which to measure the performance of the other systems. Systems based on the KL-Divergence have been amongst the best performing systems at both MIREX 2006 (72) and 2007. As shown in table 5.3, the KL divergence based index is significantly better at finding tracks from the same artist, album or genre than the simple Euclidean distance, achieving scores 11 – 14% better for artist matching, 6 – 9% better for artist-filtered genre matching and 3 – 9.5% better for matches from the same album. However, the KL-Divergence based retriever does produce a much larger hub (the track similar to the most other tracks) and a greater number of orphans (tracks that are never similar to any other track). In fact over 16% of all tracks are never in the top 5 results of any other track and 1.7% never in the top 50 for any other track. The Euclidean distance is slightly better than this with 10% never similar at 5 results and 0.8% never similar at 50. Finally, the Euclidean distance space is metric (as expected) whereas the KL-Divergence based space is not (the triangular inequality only holds for about 40% or triangles).

The use of a trained Linear Discriminant Transform, to project the mean and covariance features into a subspace with better separation between tracks from different genres, appears to have a very positive effect on the performance of the Euclidean distance based search. This approach improves the artist filtered genre matching performance by over 11% (24.6% at 5 results – 20.2% at 50 results compared to 35.8 – 27.7%) and even exceeds the performance of the KL-Divergence on that statistic (which achieved 33.3 – 26.7%). Such an increase is to be expected, given that the transform was trained to separate the classes by genre. Further, the artist match rate also increases from 25.7% to 30.6%, despite the fact that the transform was not trained to optimise that statistic (although the majority for tracks by a single artist would have appeared within a single genre in the training set).

Table 5.3: Automated evaluation of timbral frame based search indexes

Feature set	Retrieval system	Eval metric	Result level 5	10	20	50	-
Random baseline	Random baseline	album matches	0.15%	0.15%	0.15%	0.15%	-
		artist matches	7.74%	7.74%	7.74%	7.74%	-
		genre matches	9.07%	9.07%	9.07%	9.07%	-
		Triangular ineq.	-	-	-	-	-
Timbre frames	KL divergence	album matches	15.06%	10.72%	7.22%	4.13%	-
		artist matches	39.52%	36.05%	32.92%	29.55%	-
		artist-filt genre	33.30%	31.36%	29.33%	26.70%	-
		Worst 'hub'	61	90	152	316	-
		% orphans	16.15%	8.61%	4.28%	1.70%	-
		Triangular ineq.	-	-	-	-	40.0%
Timbre frames mean & covar	Euclidean dist	album matches	5.49%	4.07%	3.00%	1.93%	-
		artist matches	25.65%	23.25%	21.16%	18.56%	-
		artist-filt genre	24.64%	23.32%	21.90%	20.15%	-
		Worst 'hub'	34	61	93	195	-
		% orphans	10.66%	4.97%	2.29%	0.81%	-
		Triangular ineq.	-	-	-	-	100.0%
Timbre frames mean & covar	LDA Euclidean dist	album matches	5.38%	4.57%	3.65%	2.58%	-
		artist matches	30.59%	29.74%	28.81%	27.69%	-
		artist-filt genre	35.84%	34.98%	33.83%	31.74%	-
		Worst 'hub'	23	45	82	196	-
		% orphans	6.79%	2.51%	1.06%	0.34%	-
		Triangular ineq.	-	-	-	-	100.0%

The LDA and Euclidean distance based search strategy produces a lower number of hubs (23 – 196) than either the standard Euclidean distance (34 – 195) or the KL Divergence (61 - 316) and a lower proportion of orphans (6.8 – 0.3% compared to 10.7 – 0.8% and 16.2 – 1.7%). If, as Aucouturier (2) and Pampalk (55) suggest, the presence of such search space anomalies are significantly detrimental to the perceived performance of the system, then this approach may have significant advantages over the KL-Divergence. Further, the transformation reduced the 299 feature columns to just 10 coefficients (the LDA produces $V - 1$ dimensions where V is the number of classes it is trained to separate), yielding a huge saving in memory footprint and search time. Finally, as the search space is metric, it can be indexed with standard techniques for metric spaces to scale to even larger databases.

The versions of the KL-Divergence, Euclidean distance and LDA-Euclidean distance searches based on the timbre event vectors produce very similar performance although fractionally lower. In the classification experiments the event vectors produced fractionally higher performance hence any differences may simply be noise in the performance estimates.

Table 5.4: Automated evaluation of timbral frame and event based search indexes - classification likelihoods based systems

Feature set	Retrieval system	Eval metric	Result level 5	10	20	50	-
Timbre frames mean and covar	SMO likelihoods Euclidean dist	album matches	2.96%	2.45%	2.06%	1.62%	-
		artist matches	27.86%	27.24%	26.90%	26.28%	-
		artist-filt genre	29.37%	28.42%	26.70%	24.87%	-
		Worst 'hub'	59	69	93	171	-
		% orphans	13.61%	6.92%	2.51%	0.02%	-
		Triangular ineq.	-	-	-	-	100.0%
Timbre frames mean and covar	SMO likelihoods Cosine dist	album matches	2.94%	2.46%	2.07%	1.64%	-
		artist matches	27.82%	27.26%	26.94%	26.35%	-
		artist-filt genre	29.20%	28.14%	26.61%	24.91%	-
		Worst 'hub'	59	66	93	170	-
		% orphans	13.51%	6.89%	2.56%	0.05%	-
		Triangular ineq.	-	-	-	-	49.1%
Timbre frames mean and covar	LDA likelihoods Euclidean dist	album matches	0.76%	0.72%	0.76%	0.93%	-
		artist matches	0.87%	0.70%	4.83%	15.05%	-
		artist-filt genre	18.60%	22.26%	25.22%	27.62%	-
		Worst 'hub'	506	506	506	506	-
		% orphans	98.38%	97.02%	94.31%	86.19%	-
		Triangular ineq.	-	-	-	-	100.0%
Timbre frames mean and covar	LDA likelihoods Cosine dist	album matches	0.76%	0.72%	0.76%	0.93%	-
		artist matches	0.87%	0.70%	4.83%	15.05%	-
		artist-filt genre	18.60%	22.26%	25.22%	27.62%	-
		Worst 'hub'	506	506	506	506	-
		% orphans	98.38%	97.02%	94.31%	86.19%	-
		Triangular ineq.	-	-	-	-	100.0%
Timbre events	MVCART likelihoods Euclidean dist	album matches	5.73%	4.77%	3.86%	2.73%	-
		artist matches	30.90%	30.23%	29.50%	28.48%	-
		artist-filt genre	32.66%	30.98%	29.73%	28.76%	-
		Worst 'hub'	14	29	49	113	-
		% orphans	3.00%	0.96%	0.34%	0.15%	-
		Triangular ineq.	-	-	-	-	100.0%
Timbre events	MVCART likelihoods Cosine dist	album matches	5.99%	4.91%	3.94%	2.77%	-
		artist matches	31.07%	30.24%	29.48%	28.41%	-
		artist-filt genre	32.74%	30.73%	29.48%	28.51%	-
		Worst 'hub'	15	32	57	120	-
		% orphans	3.22%	1.01%	0.32%	0.12%	-
		Triangular ineq.	-	-	-	-	56.4%

The timbral classification likelihoods based search systems results shown in table 5.4 demonstrate surprisingly good results given the high level of abstraction from the raw timbral feature vectors that the posterior classification probability vectors (likelihood vectors) represent. For example, the SMO and MVCART likelihoods provide 29.4 – 24.9% & 32.7 – 28.5% artist filtered genre match rates and 27.9 – 26.4% & 31.1 – 28.4% artist match rates respectively. These results are an improvement over the simple Euclidean distance based search, but do not quite reach the level of the KL-Divergence based search (which achieves 33.3 – 26.7% at genre matching and 39.5 – 29.6% at artist matching).

The choice of the Cosine or Euclidean distance measures does not appear to have a significant effect on performance. However, the Cosine distance does appear to produce a non-metric similarity space, preventing the use of standard techniques for scaling the search index.

The relatively high artist match rates may be caused by the low number of artists in the test collection, a trend exacerbated by the artist-filtered split of the dataset. It is with the classification likelihoods based systems that we would expect to see the highest decline in artist match rates as the number of artists is increased.

The performance of the LDA likelihoods based system is poor achieving 18.6 - 27.6% genre matches and 0.9% – 15% artist matches (note this is the only system in the results that reverses the trend of metadata match rates declining with increases in search depth). Further, the results show at least one very large hub (similar to 506 other tracks at all search depths). Hence, hubs might compose the majority of results lists and be contributing to the very high number of orphans (tracks never recommended), which comprise the majority of the collection (98.4 – 86.2%). This behaviour is likely caused by poorly distributed classification likelihood estimates. Examination of the MVCART likelihoods vectors and classification algorithm shows that they are not true posterior probabilities but normalised distances from the transformed means of each class (although they are normalised to range 0:1 and sum to 1.0) and also show a very narrow range.

It is interesting to note that the performance of the MVCART based model exceeds that of the SMO with a first-order polynomial kernel. As we know from the experiments in chapter 4 the performance of the SMO exceeds that of the MVCART model at classification. Hence, we must conclude that although the SMO might be a more accurate classifier, the MVCART algorithm provides posterior probability profiles that are better suited to comparison.

The results for MVCART transcription based systems are given in table 5.5. These show good performance for the transcription search indexes on both the artist and artist-filtered genre matching metrics achieving similar performance

Table 5.5: Automated evaluation of timbral frame based search indexes - MVCART transcription based systems

Feature set	Retrieval system	Eval metric	Result level 5	10	20	50	-
Timbre events	MVCART 738 terms TF/IDF	album matches	9.68%	7.67%	5.83%	3.73%	-
		artist matches	33.97%	32.37%	30.82%	28.84%	-
		artist-filt genre	31.37%	30.18%	29.40%	28.11%	-
		Worst 'hub'	27	39	73	162	-
		% orphans	8.74%	2.34%	0.42%	0.00%	-
		Triangular ineq.	-	-	-	-	100.0%
Timbre events	MVCART 2048 terms TF/IDF	album matches	10.78%	8.32%	6.23%	3.99%	-
		artist matches	35.10%	33.19%	31.40%	29.22%	-
		artist-filt genre	31.28%	30.42%	29.36%	28.19%	-
		Worst 'hub'	31	49	89	175	-
		% orphans	9.03%	2.81%	0.34%	0.02%	-
		Triangular ineq.	-	-	-	-	100.0%
Timbre events	MVCART 4091 terms TF/IDF	album matches	10.28%	8.05%	6.04%	3.84%	-
		artist matches	34.33%	32.70%	31.12%	29.01%	-
		artist-filt genre	31.34%	30.15%	29.21%	27.98%	-
		Worst 'hub'	26	46	85	151	-
		% orphans	8.69%	2.26%	0.22%	0.00%	-
		Triangular ineq.	-	-	-	-	100.0%
Timbre events	MVCART 6210 terms TF/IDF	album matches	10.44%	8.19%	6.15%	3.98%	-
		artist matches	34.95%	32.98%	31.20%	29.09%	-
		artist-filt genre	31.10%	30.18%	29.56%	28.28%	-
		Worst 'hub'	42	68	119	192	-
		% orphans	10.88%	3.05%	0.37%	0.00%	-
		Triangular ineq.	-	-	-	-	100.0%

to the likelihoods-based indexes based on an MVCART model and the KL-Divergence. The artist matching scores exceed those of the Likelihoods based systems and approach the performance of the KL-Divergence on the longer results lists (20 and 50 results). The numbers of hubs and orphans produced by the MVCART transcription-based systems are lower than those produced by the KL Divergence and similar to the numbers produced by the other search techniques introduced. The distance space produced appears to be mostly metric (at worst failing only 5 of the 100,000 triangles tested).

The size of the MVCART model (number of leaves) used does not seem to have a strong effect on the performance of the search index although 768 leaf model does seem to perform worse than those based on 2048, 4091 and 6210 leaf models. It is likely that the model size will have more of an effect as the collection size is increased, by providing slightly more detailed descriptions of the track's content. Larger models may also lead to more sparse indexes (as tracks are less likely to contain the same subsets of leaf nodes in their event transcription). Hence, the

optimisation of the search index performance this parameter should be examined when scaling the search index by a significant amount.

The choice of TF or TF IDF weighting also appears to have little effect on the performance of the search index. The simpler TF weighting may therefore be preferable from an implementation standpoint. However, if attempting to make the index more sparse (as we do in the optimised version of the index search by manipulating the α parameter) the TF IDF weighting may be advantageous as it will eliminate tracks from the first pass search index which overlap with the query only on the most widely visited leaf nodes. Hence, the choice of term weighting should also be considered when scaling an index to a much larger dataset.

Overall the results for the RCC based search indices, shown in table 5.6, are lower than those for the timbral features. As the RCC feature extraction pipeline produces too few vectors to estimate the KL-Divergence, other techniques must be used to produce a search index from them. Computing a mean vector and comparing with the Euclidean distance is the simplest approach to this, producing 20.9 – 18.9% artist match rates and 20.7 – 17.6% genre matches fully the fully log-scale RCC features. The use of an LDA transform on the features tended to hurt performance, supporting the conclusions in chapter 4 that linear classifiers do not perform well on the RCC features.

The RCC features computed with a fully log-scale filterbank appear to produce marginally superior performance to those produced with linear and log-scale (above 4 Hz) filterbank in all cases, despite producing fewer feature dimensions.

The MVCART transcription based models consistently provide a higher level of performance on the artist-filtered genre matching statistic than the Euclidean distance based models and a similar level of performance at artist matching. In particular, the MVCART models appear to extend their genre matching performance deeper into the results lists, e.g. the 602 leaf MVCART based on RCCs produced with a log-scale filterbank provides 23.6 – 22.2% genre matches at 5 and 50 results respectively, while the Euclidean distance provides only 20.7 – 17.7%.

Table 5.6: Automated evaluation of RCC based search indexes

			Result level				
Feature set	Retrieval system	Eval metric	5	10	20	50	-
Rhythm	MVCART 602 terms	album matches	1.47%	1.31%	1.21%	1.08%	-
log scale	TF/IDF	artist matches	19.71%	19.27%	18.52%	17.78%	-
		artist-filt genre	23.61%	23.33%	22.83%	22.17%	-
		Worst 'hub'	292	348	407	575	-
		% orphans	11.52%	5.29%	3.03%	1.03%	-
		Triangular ineq.	-	-	-	-	100.0%
Rhythm	MVCART 1555 terms	album matches	1.24%	1.14%	1.10%	1.00%	-
log scale	TF/IDF	artist matches	18.99%	18.47%	17.97%	17.64%	-
		artist-filt genre	23.58%	22.99%	22.63%	21.44%	-
		Worst 'hub'	304	353	460	683	-
		% orphans	11.69%	5.09%	3.08%	1.35%	-
		Triangular ineq.	-	-	-	-	100.0%
Rhythm	mean	album matches	2.26%	1.97%	1.65%	1.23%	-
log scale	Euclidean dist	artist matches	20.87%	20.31%	19.64%	18.91%	-
		artist-filt genre	20.65%	19.91%	19.06%	17.64%	-
		Worst 'hub'	21	31	57	141	-
		% orphans	3.37%	0.79%	0.17%	0.05%	-
		Triangular ineq.	-	-	-	-	100.0%
Rhythm	mean	album matches	1.38%	1.13%	0.92%	0.68%	-
log scale	LDA	artist matches	19.08%	18.48%	17.92%	16.97%	-
	Euclidean dist	artist-filt genre	15.11%	13.83%	12.91%	11.60%	-
		Worst 'hub'	17	30	58	133	-
		% orphans	2.98%	0.89%	0.25%	0.10%	-
		Triangular ineq.	-	-	-	-	100.0%
Rhythm	MVCART 352 terms	album matches	1.26%	1.12%	1.04%	0.98%	-
log scale after 4Hz	TF/IDF	artist matches	18.98%	18.82%	18.89%	18.60%	-
		artist-filt genre	21.03%	20.79%	20.51%	20.00%	-
		Worst 'hub'	649	664	728	1062	-
		% orphans	26.11%	18.68%	16.34%	12.72%	-
		Triangular ineq.	-	-	-	-	93.1%
Rhythm	MVCART 1079 terms	album matches	1.14%	1.10%	1.07%	1.02%	-
log scale after 4Hz	TF/IDF	artist matches	19.00%	19.20%	19.26%	19.08%	-
		artist-filt genre	21.87%	21.02%	20.76%	19.82%	-
		Worst 'hub'	567	640	833	1,265	-
		% orphans	21.24%	14.92%	12.45%	9.01%	-
		Triangular ineq.	-	-	-	-	99.9%
Rhythm	mean	album matches	2.13%	1.75%	1.51%	1.21%	-
log scale after 4Hz	Euclidean dist	artist matches	19.68%	18.80%	18.13%	17.18%	-
		artist-filt genre	18.25%	18.15%	17.70%	16.82%	-
		Worst 'hub'	15	26	42	92	-
		% orphans	2.46%	0.62%	0.12%	0.02%	-
		Triangular ineq.	-	-	-	-	100.0%
Rhythm	mean	album matches	1.80%	1.43%	1.12%	0.81%	-
log scale after 4Hz	LDA	artist matches	19.41%	18.65%	17.89%	16.88%	-
	Euclidean dist	artist-filt genre	16.90%	15.97%	14.64%	13.02%	-
		Worst 'hub'	15	30	55	128	-
		% orphans	3.10%	0.89%	0.17%	0.05%	-
		Triangular ineq.	-	-	-	-	100.0%

The downside to the MVCART indices appears to be that they produce a relatively high number of hubs , e.g. 292 – 575 for the index produced from the 602 leaf model based on log-filterbank RCCs and TF IDF term weighting, but few orphans in longer results lists (10.9 – 0.0%). TF term weightings seem to exacerbate the effect of hubs and, in the case of the 602 leaf, fully log-scale

RCC based models increase the size of the worst hub to 467 – 960 matches and the orphan rates to 17.7 – 2.1%. This effect may indicate that the RCC-based features are better used as part of a search ensemble with other more detailed similarity measures.

Again the search spaces produced are mostly metric, with the log-scale models failing (at worst) 76 out of 100,000 triangles and the linear then log scale RCC-based models failing (at worst) 6,880 out of 100,000 (although generally they fail 1,000 or less).

5.4.4.1 Combined search systems

The results of the pseudo-objective evaluation for the combined search systems are given in table C.1 in appendix C.

The results for the combination of the the KL-Divergence based search system and the best performing RCC-based model (MVCART transcription of log-scale RCCs with TF IDF term weighting) show significantly greater performance for the rank normalised product combination of the search indices than either the weighted average or simple product combination. The performance of the rank normalised product approaches the performance of the KL-divergence on the artist-filtered genre matching metric but does not achieve the same level of performance on the artist-matching metric. However, it seems likely that the combination is not achieving any advance in performance overall. The drastically reduced performance of the standard product and weighted average combinations perhaps indicates that the indexes combined have very different distributions and that these can be effectively normalised through rank normalisation.

Generally the similarity spaces produced by the combinations are significantly more metric (99.97 – 91.13% of triangles tested are valid) than the original KL Divergence based similarity space (where only 40% of triangles are valid).

The matching combinations (product, weighted average and rank normalised product) of the MVCART transcription based index of the timbral event vectors with the same RCC-based MVCART transcription index show different trends

from the combinations of the KL-Divergence between timbral frames and the RCC-MVCART index. In particular every combination type improves on the performance of the component search indices on the artist-filtered genre matching metric. These gains are made at the expense of a slight drop in artist matching performance. Further, there are few apparent differences between the product combination and rank normalised product combination, while the weighted linear combination exceeds the performance of both. This may be due to the fact that the similarity scores from the two different MVCART transcription based search indexes have more similar distributions and may be more easily combined without normalisation.

As the search indexes combined are metric (or largely so) it is not surprising that the resulting combinations also produce similarity spaces that are metric.

The final set of combined search indexes are based on combined feature vectors with a common search strategy. The feature vectors combined are the mean and covariance of timbral feature frames and mean RCC vectors. Comparison of these combined feature profiles with the Euclidean distance yields slightly improved performance over the the comparison of just the summaries of the timbral features, demonstrating up to a 2% increase at artist-filtered genre matching and up to a 2.5% increases at artist matching. However, the use of an LDA to transform the combined features before comparison again yields a strong increase in performance, with gains of 10% on the artist-filtered genre matching metric and $5 - 6\%$ on the artist matching metric. This system also produces the highest level of performance on the artist-filtered genre matching metric of any system in the test, although this is only a moderate improvement over the the index based on an LDA of the just the timbral feature summaries.

These two systems (Euclidean distance based and Euclidean distance in an LDA transform space) also produce some of the smallest 'hubs' in the tests with the worst hub being similar to only $21 - 187$ or $25 - 180$ other tracks. This compares very favourably to the KL-divergence based systems, which tend to produce ubs $2 - 2.5$ times larger.

Although the best of the various combination strategies tested in these experiments produce, at best, marginal increases in performance of the best of their component search indexes, we can confirm that several of the combinations do not hurt performance and therefore on more diverse datasets (particularly those containing a greater proportion of dance music or other rhythmically defined musics) we may see better increases in performance. We have also seen that for the combination of dissimilar search strategies it is advantageous to normalise the similarity spaces before combination.

5.4.5 Human evaluation

To complement the automated pseudo-objective statistical evaluation of the search systems, a human evaluation of their performance is also conducted. This is labour intensive, and so it is difficult to get enough evaluations to achieve good coverage over the collection. However, in order to ensure that each genre of music is at least represented in the evaluation, queries have been chosen in a stratified random fashion (i.e., an equal number of queries are randomly chosen from each genre in the test database). A total of 60 queries are evaluated by the human 'graders'. Each query is evaluated by a different 'grader', ensuring that there is no within-query variance that must be normalised, as such normalisation is non-trivial.

5.4.5.1 Candidate systems

To reduce the complexity of the human evaluation a subset of the systems were selected to be evaluated. These were:

framesMeanCovarKL The KL-Divergence search index based on the timbral feature frames.

framesMeanCovarLDAEuc The Euclidean distance based search index based on the timbral feature frames transformed using a Linear Discriminant Analysis.

eventsTranscription0.003 TFIDF Timbral event vectors transcribed using an MVCART model and TF IDF term weighting, compared using the Cosine distance.

rhythmLL0Transcription0.001 TFIDF RCC LL0 vectors transcribed using an MVCART model and TF term weighting, compared using the Cosine distance.

eventsMVCARTLikeyCos Timbral event vectors converted into posterior probability profiles (likelihoods), compared using the Cosine distance.

events0.0003TFIDF RhythmLL0 0.001TFIDF weightedMean The weighted mean of the eventsTranscription0.003 TFIDF and rhythmLL0Transcription0.001 TFIDF indexes.

events0.0003TFIDF RhythmLL0 0.001TFIDF productRankNorm The rank normalised product combination of the eventsTranscription0.003 TFIDF and rhythmLL0Transcription0.001 TFIDF indexes.

framesMeanCovar RhythmLL0 Mean CombinedVector LDAEuc Concatenated Timbre frames mean and covariance vectors and RCC LL0 mean vectors, transformed using a Linear Discriminant analysis and compared using the Euclidean distance.

5.4.5.2 Evaluation infrastructure

The Evalutron 6000 system produced by the IMIRSEL group at the University of Illinois was used to conduct the human evaluation of the search systems. Figure 5.2 shows a screen-shot of the Evalutron 6000 in use. The results from each system for each query are pooled to produce a 'retrieved set' for each query. The queries are distributed equally amongst the human 'graders' and all candidates for a particular query are presented to the user that the query was assigned to.

Each query and candidate result is represented by a 30 second audio clip. To avoid sampling problems (where an inappropriate clip is selected to represent the track), these are exactly the same clips analysed by each search system.

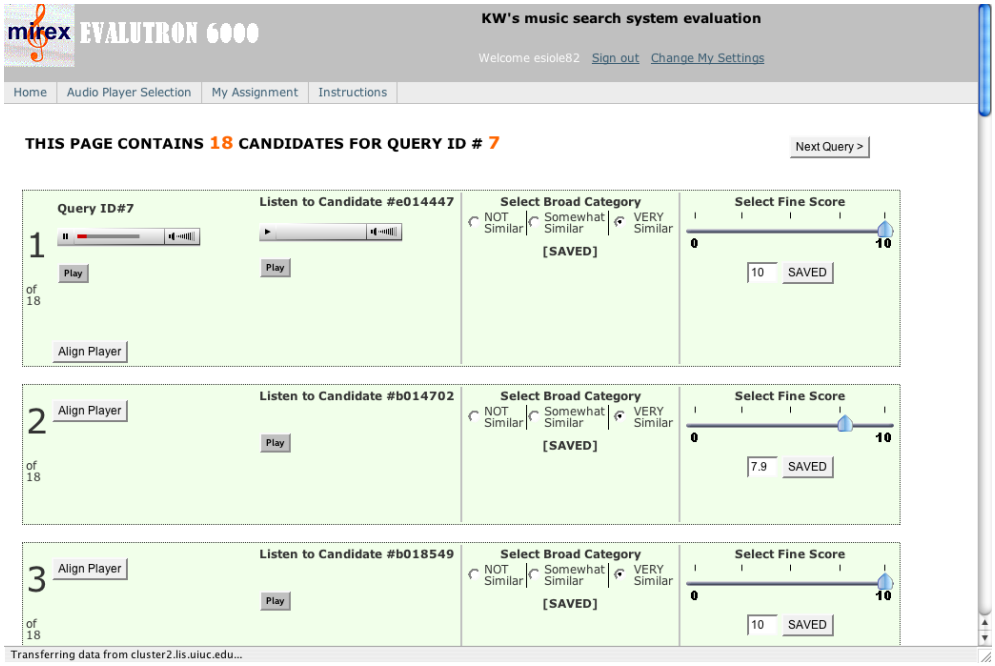


Figure 5.2: Screen-shot of the Evalutron 6000 system used to conduct the human performance evaluation.

5.4.5.3 Scoring systems

The Evalutron system requires human 'graders' to enter two scores for each query/candidate pair. The first score is categorical, asking a user to determine whether a candidate is 'not similar', 'somewhat similar' or 'very similar' to the query. These are translated to scores of 0, 1 or 2 respectively, for the purposes of numerical evaluation. The second score that must be applied is a fine score in the range 0:10.

It is assumed that the categorical scores applied by 'graders' will vary less than the fine numerical scores. However, they provide significantly less scope for the grader to express their preferences regarding the search results. At present, it is not clear which approach to scoring is appropriate, hence both scoring systems are used in the evaluation and their results compared.

As part of the post-processing of the human applied evaluation scores, the categorical and fine scores corresponding to the candidates from each system for each query are averaged to produce a single evaluation score for each system for

each query. It is these per query averages that are used to rank and compare the systems using the Friedman test and multiple comparisons.

5.4.5.4 Statistical Significance testing

It has been demonstrated that retrieval result sets and human evaluation scores have non-normal distributions in many different Information Retrieval disciplines (63). This is likely to also be true of human evaluation scores of music search results, as different subjects are likely to grade on different scales, have different music tastes, different levels of expertise with particular music types, and different expectations of what a 'search-by-example' system for music should find. This poses a number of problems when evaluating music search results, as there is no simple way of normalising and combining the human evaluation scores in order to produce a single evaluation metric. This is due to the fact that each human's opinion on performance is subjective and set on their own internal 'similarity scale' – which may be dependent on mood, time of day or other uncontrollable or un-measurable factors. For a number of years, the TREC community has used the Friedman test to overcome such issues because it is non-parametric (i.e., it does not assume normal distribution of the underlying data) (9).

As stated in section 4.3.3.1, Friedman's ANOVA compares systems over a number of blocks, in this case the queries whose results were evaluated by human 'graders'. So called 'row effects', or variance introduced into the results by different combinations of human 'grader' and query are handled by replacing the actual scores with their ranks amongst the systems compared (the columns). Statistically valid pair-wise comparison of all the system results for significant differences are again made using the Tukey-Kramer technique. The α value used in the comparisons made in order to reject H_0 (the hypothesis that each pair of systems have the same error rate) is 0.05.

5.4.6 Subjective (Human) evaluation results

5.4.6.1 Average result scores

Although simple averages of the scores applied by the human evaluators over all queries are not a principled method of comparing the performance of the

Table 5.7: Average categorical scores from the human evaluation by system

System name	Average categorical score
events0.003TFIDF RhythmLL0 0.001TFIDF weightedMean	1.16
framesMeanCovarKL	1.12
events0.003 TFIDF	1.11
framesMeanCovar RhythmLL0 CombinedVectorLDAEuc	1.06
eventsMVCARTLikeyCos	1.04
events0.003TFIDF RhythmLL0 0.001TFIDF productRankNorm	1.01
framesMeanCovarLDAEuc	0.98
rhythmLL0Transcription0.001 TF	0.77

systems, they are still interesting for use in gauging the users' overall opinion of the performance of the search indices. The average categorical scores shown in table 5.7 demonstrate that the majority of systems achieved results that, on average, the human 'graders' classified as somewhat similar or better.

The results also show good performance for the standard timbral frames KL-Divergence index and similar performance for MVCART index of the timbral event vectors. Interestingly, the Euclidean distance in an LDA transform space-based index does not perform well (despite the excellent performance it achieved on the artist-filtered genre matching metric). The best performing system is the weighted mean combination of the MVCART index of the timbral event vectors with MVCART index of the RCC features. The rank normalised product version of this combination does not perform as well, failing to exceed the performance of the MVCART index of the timbral event vectors. The LDA-Euclidean index of the combined feature vectors performs better than the LDA-Euclidean index based on only the timbral feature summaries, but does not approach the performance of some of the other indices.

The performance of the RCC only based index is significantly below that of the other indices.

The average fine scores applied by the human graders, shown in table 5.8, show very similar trends and virtually the same ranking of systems, transposing one pair of results and significantly promoting one other system. Specifically, the MVCART index of the timbral features and the KL-divergece based index swap positions, although the interval between these two systems in both result tables is marginal at best. The LDA-Euclidean index of the timbral feature frames is promoted up the ranking in the average fine score table. However, it still provides

Table 5.8: Average fine scores from the human evaluation by system

System name	Average fine score
events0.003TFIDF RhythmLL0 0.001TFIDF weightedMean	5.55
events0.003 TFIDF	5.37
framesMeanCovarKL	5.36
framesMeanCovar RhythmLL0 CombinedVectorLDAEuc	5.24
framesMeanCovarLDAEuc	5.05
events0.003TFIDF RhythmLL0 0.001TFIDF productRankNorm	4.95
eventsMVCARTLikeyCos	4.90
rhythmLL0Transcription0.001 TF	3.78

significantly lower performance than the LDA-Euclidean index augmented with the RCC mean vectors.

5.4.6.2 Significance test results

The Friedman test with multiple comparisons tables must be used to make statistically valid pair-wise comparisons between the performances of the search indices. The Friedman test results for the categorical scores are given in table 5.3, while the results for the fine scores are given in table 5.4. These tables are highly correlated and the trends evident can be dealt with together.

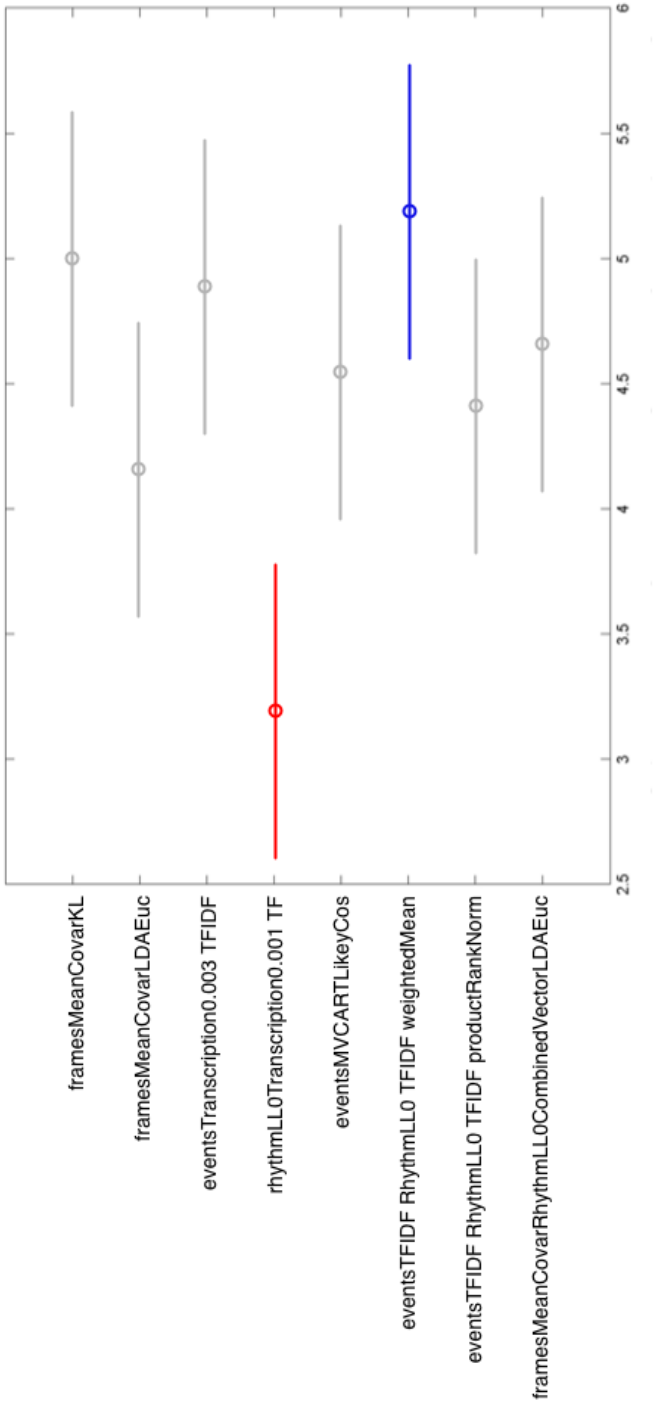


Figure 5.3: Comparison of search index performance using human applied category scores and the Friedman test and multiple comparisons.

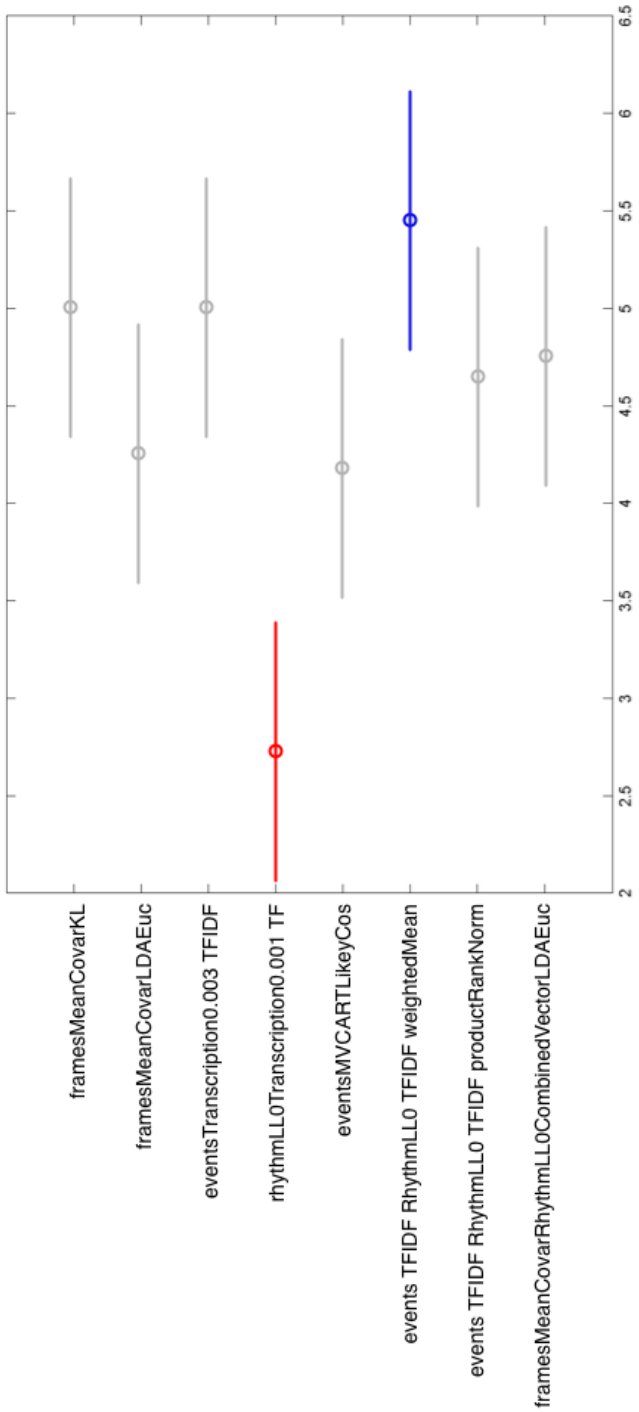


Figure 5.4: Comparison of search index performance using human applied fine scores and the Friedman test and multiple comparisons.

The only statistically significant differences evident in the results are between every other system and the RCC-based MVCART transcription index, although the difference is not significant between this index and the LDA-Euclidean index of the timbral frame summaries in the categorical scores. The best performing system in both results sets is the weighted average of the MVCART transcription indexes of the timbral event vectors and the RCC vectors, although the increase in performance is not statistically significant.

One of the most valuable observations from these results is that there are a number of approaches to the creation of audio music similarity based recommendation/search indexes that produce an equivalent level of performance to the highly computational intensive KL-Divergence. These approaches all require a training set on which to base their analyses (by training a genre classifier that may be exploited in a number of different ways). Although this might be a considered a disadvantage (as a representative training set must be selected and labelled from any collection to be indexed) it does yield three strong advantages.

Firstly, the similarity spaces produced by all these approaches (LDA transforms with Euclidean distance measurements, Classification likelihoods with Euclidean distance measurements and MVCART-based transcription, TF IDF term weighting and comparison with the Cosine distance) are metric (the triangular inequality holds), allowing the application of standard techniques for scaling indices based on metric spaces. In the case of the MVCART transcription indices, alternative methods of scaling the index based on retrieval techniques used in the field of text search may also be applied to scale indices, without the long indexing runs normally associated with techniques for scaling metric similarity estimators.

Further, the second advantage of the LDA and likelihoods based approaches is that they achieve significant reductions in the dimensionality of the feature space and therefore will achieve significant reductions in the computational complexity of the index building process, or on-the-fly queries. The MVCART transcription based systems also achieve significant computational savings through efficient sparse vector techniques and intelligent indexing procedures (including the first pass search described in section 5.2.3.1).

The final advantage of these systems is that the use of a training dataset offers opportunities for the tuning of a search index, which is not possible with a KL-divergence-based index, other than by modifying the audio features upon which it is based. Hence, exemplar selection might be one method of improving performance of such systems, and might offer a method of adjusting the performance of a system that under-performs on a particular class or genre of music.

Chapter 6

Summary and Discussion

This thesis has presented a number of novel or modified techniques for the common audio MIR tasks of genre classification and similarity–search (or search–by–example). The specific contributions made and conclusions drawn are detailed in the following sections.

6.1 Music audio feature extraction

6.1.1 Efficient timbral feature processing based on segmented feature streams

The use of onset-detection based segmentation to produce feature vectors corresponding to individual audio events in the audio stream was proposed and shown to yield minor gains in performance at genre classification. Further to this, the event–level segmentation appears to reduce the complexity of some types of feature comparison and increase utility of the feature vectors over simple frame–level vectors. For example, in early tests (69), the transcription of event level feature vectors with an MVCART model provided significantly better performance than the transcription of frame–level features for a very large computational saving. Such gains might also be achieved for other tasks such as preview clip selection.

Further, the significant reduction in the data retained may have other benefits including an order of magnitude reduction in the storage space required to keep

the full feature stream (rather than the single vector summaries). This would enable applications that operate on the audio stream rather than over whole tracks (such as preview clip selection, automated mixing, intelligent effects, lighting and visualisation) by significantly reducing transmission bandwidths and computational complexity.

6.1.2 A new audio representation for rhythmic classification

A novel and compact parameterisation of music audio, known as Rhythm Cepstral Coefficients (RCCs), intended to inform a model as to the rhythmic content of a track were introduced. The descriptors are derived from an onset detection function and therefore only represent timing information in the signal. The performance of the RCCs at classification and retrieval tasks is often significantly lower than the timbral parameterisations. However, they do capture useful information that would not be available in the timbral parameterisations (as they are based on the mean and covariance of the frame or event level feature vectors and not temporal patterns over the stream of vectors).

Unfortunately, the tests reported here do not include many dance music genres, which are likely to be differentiated more by their timing or rhythmic patterns than their spectral or timbral characteristics. However, in classification tests, the RCCs did perform very well at identifying the two genres with strong beats or rhythms (Dance & Electronica and Hip Hop/Rap). Additional tests should be performed to examine the performance of these features over collections with a greater proportion of dance music genres, such as those used in (32) and (24).

Experiments which varied the scaling of the frequency axis on which the RCCs are computed showed the best performance for scales which incremented consistently (linearly or logarithmically) between 1 and 4 Hz, while those that change from linear to logarithmic scaling produced lower performances. Given the low difference in performance between the fully logarithmic RCCs and those with

a transition frequency of 4 Hz the logarithmically scaled RCCs are likely to be preferable due to the reduced computational complexity yielded by shorter vectors.

6.2 Genre classification

Experimentation with the classification of feature vectors generated by varying segmentations of the feature stream has shown that the best performance is produced by linear classifiers trained on mean and covariance summaries of the timbral feature vectors over a whole track. This performance is statistically significantly higher than that of Decision Tree based classifiers and higher (although not significantly) than the performance of any models trained on 10 second windows of a track.

The training of linear classifiers over event-level feature vectors was problematic, perhaps due to the very large number of vectors produced per track. However, the performance of Decision-tree classifiers was often significantly better when trained on event-level feature vectors rather than the summaries over whole tracks. The additional level of detail provided by the event level feature vectors may have allowed the decision tree models to provide a meaningful separation of detected audio events into clusters (leaf nodes) which can be used to classify (or index) a track.

The RCC features also performed significantly better when classified using a Decision tree-based technique, rather than a linear classifier, with gains in performance of up to 15%. On the test database the RCCs produced better performance with shallower decision trees, perhaps indicating that the model size should be optimised when indexing collections of varying size and taxonomy. The RCC-based decision tree classifiers also performed better over multiple vectors of RCCs per track than mean or mean variance vectors.

These differing trends for different segmentations of feature streams and feature types show that the classification strategy employed for novel feature types should be examined through experimentation for each feature type developed to

represent music audio.

The confusion demonstrated by many of the classifiers explored was distributed over very similar classes (e.g. Hard Rock & Metal is confused with Rock, while Classical music genres are confused with each other). Such mistakes might be considered 'human-like'.

6.2.1 A Multivariate Decision Tree Classifier

A multi-variate version of the CART Decision tree classifier (MVCART) was proposed, which implements the multivariate node splitting process by transforming the feature vectors using a linear discriminant analysis and iterating over all the combinations of the classes into two groups (a binary split) and training single Gaussian classifiers to reproduce the split. The split which maximises the reduction in entropy of the distribution of the vectors at the node over the classes is chosen as the final split of the node. The reported experiments show that the MVCART models provide gains in performance over conventional Decision Tree based techniques, particularly when dealing with larger numbers of vectors per example. Further, the increase in performance significantly closes the gap between the best event-level feature vector based classifier and the best classifier based on the whole track.

6.2.2 Combining classification models

A number of methods were explored for combining genre classifiers. The best performing approach examined was the concatenation of the single feature vectors representing the whole track and the training of a single linear SVM. This provided a limited gain in performance over the use of only timbral features, however further experiments should be conducted to examine whether the combined classification system performs better over datasets with more dance music genres.

6.3 Music similarity estimation / search

Many of the best performing music similarity estimation techniques detailed in prior work suffer from very high computational complexity as they are based on techniques such as the KL-Divergence, Monte-Carlo sampling or the Earth Mover's Distance (EMD). These techniques are also difficult to scale with standard indexing techniques as they produce non-metric similarity spaces. Levy and Sandler (43) propose a modified approach based on the Mahalanobis distance, which does produce a metric similarity space and at a lower computational cost, but also provides a lower level of performance. A number of alternative approaches have been demonstrated in this thesis that leverage the important cultural information in music genres to significantly reduce the query-time complexity of audio music searches, produce metric similarity spaces and even to yield a minor gain in performance.

In particular, the use of either genre classification likelihood profiles, Linear discriminant transforms (trained to separate genres) or MVCART transcription of event-level feature vectors enables us to build search models which both drastically reduce the computational complexity of searches and produce metric similarity spaces, enabling the use of standard techniques for scaling search models. It should be noted that the performance of the KL-Divergence based search was not significantly better (in human tests) than any of the likelihoods, transcription or linear transform-based approaches (where trained on the same underlying features).

The MVCART transcription-based model provides the best performance amongst the proposed approaches, producing very similar performance to that of the KL-Divergence based retriever. Further, the ability to throttle the complexity of the optimised MVCART-transcription based search and the possibility of achieving sub-linear scalability through index partitioning may make the MVCART-transcription based search the most practically and industrially useful technique explored.

The automated statistical evaluation of the music search techniques broadly showed the same grouping and ranking of search systems as the human evaluation (as it did in the results of MIREX 2006 (27), (28)), despite a few permutations of similarly performing techniques. These results further support the use of such statistical evaluations in the development and evaluation of music search tools.

6.3.1 Combining search models

A number of techniques were explored for the combination of search models. The weighted mean of the search models based on the TF/IDF vectors computed from MVCART transcriptions of both the event-level timbral features and the RCCs provided the highest level of performance although this was not significantly better than the other approaches or several of the purely timbral search models. However, a number of advantages have been outlined for each technique, including the ability of the ranked normalised product combination to combine search models with significantly different distributions of result scores and efficient use combination of sparse indices using the product combination (tracks with no similarity in one index need not be compared using other indices).

6.4 Future work

The experiments and novel techniques presented here provide a number of different avenues that warrant further exploration, including:

- The performance of RCC-based classification and search models needs to be measured over sets of dance music genres, such as those used in (32) and (24), needs to be explored to determine if their combination with timbral classification procedures yields statistically significant gains in performance.
- Evaluation of the performance Pitch Class Profile or Chroma vectors used in musical Key estimation (58) and cover song detection (35) for classification and search and both to discover what model type is most appropriate

and whether the resulting classifier can be used in combination with other techniques to yield better performance overall.

- Measure the performance of an alternative form of MVCART classifier which replaces the node splitting process, currently implemented using an LDA (over all classes) and Gaussian classifiers trained to reproduce splits of the classes into two groups), with linear support vector machines trained to reproduce the splits of the classes into two groups.

- Evaluation of the performance of the MVCART classification algorithm (and its use in search) when applied multi-label tag classification (where the tree training procedure selects splits based on the reduction in entropy over the tags applied to tracks with vectors appearing in a node).

Finally, Aucouturier (2) suggests that the 'glass ceiling' that he has observed on the performance of audio-based musical genre classification and search techniques may be due to the polyphonic nature of music and the inability of current techniques to separate the audio into separate voices and analyse each individually, a faculty displayed by human auditory perception. Unfortunately, source separation techniques will return a variable number of sources for recording depending on the content, recording conditions and specifics of the technique applied. This presents a problem for most techniques which require single vector or distribution to analyse (and therefore cannot handle a variable number of vectors per track), while techniques based on mixture distributions are computationally expensive, as described in section 5.1.

However, MVCART-transcription based classification and search procedures may be applied by training the MVCART model on the event vectors from all sources in the dataset and the production of a single combined TF or TF/IDF vector for each track. This would allow the modelling and transcription of each source independently of the other sources, although some degree of cross-over must be expected with current source separation techniques. Because a single model is used to transcribe each track a single TF or TF/IDF vector may be estimated for the whole track over all sources. Finally, the effect of residual channels on the term weight vectors for each track may be minimised by the

use onset detection based segmentation as less onsets will be generated on such channels and therefore a low number of events will be transcribed.

Appendix A

Classification Results

A.1 Classification Accuracy

Table A.1 shows the classification accuracy and standard deviation for each classification system, computed over a 5-fold cross-validaiton of the evaluation collection. All splits of the collection were artist filtered so that an artist's tracks only appeared in the test or training set, not split across them.

Table A.1: Classification accuracy

Feature set	Segmentation/summary	Classifier	Acc	St.dev.
Timbre	whole file mean and covariance	CART	44.55	0.61
		Simple Gaussian	41.97	0.42
		IBK	55.26	0.81
		J48	44.25	0.34
		LDA	60.53	0.76
		LDA Simple Gaussian	59.56	0.58
		LDA IBK	62.56	0.59
		MVCART0.0001	53.89	0.53
		MVCART0.001	54.03	0.45
		SMO Poly1	**63.04**	**0.45**
		SMO RBF	50.58	0.69
Timbre	10 second windows mean and covariance	CART	48.90	0.47
		J48	49.58	0.75

Continued on next page

Table A.1 – continued from previous page

Feature set	Segmentation/summary	Classifier	Acc	St.dev.
		MVCART0.0001	54.38	0.52
		MVCART0.001	54.12	0.60
		SMO Poly1	**58.62**	**0.66**
Timbre	Event-based segments	CART	41.30	0.65
	whole file	Simple Gaussian	44.89	0.51
	mean and covariance	IBk	54.38	0.60
		J48	43.85	0.16
		LDA	58.75	0.87
		LDA Simple Gaussian	59.54	1.04
		SMO Poly1	**63.51**	**0.63**
		SMO RBF0.01	48.79	0.66
Timbre	Event-based segments	CART	54.07	0.28
	event mean vectors	Simple Gaussian	39.34	0.11
		J48	55.81	0.30
		SMOPoly1	**failed**	–
		LDA	**failed**	–
		MVCART0.001	57.86	0.59
		MVCART0.0005	58.13	0.57
		MVCART0.0001	**58.94**	**0.69**
Timbre	Event-based segments	MVCART0.001 TF SMOPoly1	58.12	0.57
	event mean vectors	**MVCART0.001 TF SMOPoly2**	**59.12**	**0.51**
	TF	MVCART0.001 SMOPoly3	58.79	0.66
		MVCART0.0005 SMOPoly1	57.04	0.57
		MVCART0.0005 SMOPoly2	58.95	0.55
		MVCART0.0005 SMOPoly3	57.33	0.83
		MVCART0.0001 SMOPoly1	58.12	0.41
		MVCART0.0001 SMOPoly2	58.52	0.96
		MVCART0.0001 SMOPoly3	27.43	0.46
Timbre	Event-based segments	MVCART0.001 TF/IDF SMOPoly1	58.33	0.55
	event mean vectors	MVCART0.001 TF/IDF SMOPoly2	59.19	0.47
	TF/IDF	MVCART0.001 SMOPoly3	58.47	0.63
		MVCART0.0005 SMOPoly1	57.29	0.59
		MVCART0.0005 SMOPoly2	**59.25**	**0.47**
		MVCART0.0005 SMOPoly3	56.53	0.92
		MVCART0.0001 SMOPoly1	58.06	0.41
		MVCART0.0001 SMOPoly2	57.96	1.02
		MVCART0.0001 SMOPoly3	26.23	0.49
Rhythm	9 second windows	CART	39.65	0.50
Log-scale		Simple Gaussian	23.17	0.49
		IBk	37.44	0.70
		J48	37.34	0.32
		LDA	19.30	0.26
		LDA Simple Gaussian	26.14	0.36
		MVCART0.001	**41.29**	**1.02**
		MVCART0.0001	39.83	0.95
		SMO Poly1	27.05	0.60
Rhythm	9 second windows	**CART**	**38.84**	**0.21**
Log-scale after 2 Hz		Simple Gaussian	21.40	0.20
		IBk	35.96	0.51
		J48	35.47	0.37
		LDA	27.04	0.38
		LDA Simple Gaussian	21.31	0.14
		MVCART0.001	38.32	0.44

Continued on next page

Table A.1 – concluded from previous page

Feature set	Segmentation/summary	Classifier	Acc	St.dev.
		MVCART0.0001	37.20	0.87
		SMO Poly1	26.51	0.54
Rhythm	9 second windows	**CART**	**38.84**	**0.60**
Log-scale after 3 Hz		Simple Gaussian	22.24	0.20
		IBk	35.39	0.38
		J48	35.90	0.57
		LDA	27.66	0.14
		LDA Simple Gaussian	22.05	0.22
		MVCART0.001	37.78	0.38
		MVCART0.0001	36.05	0.68
		SMO Poly1	25.86	0.34
Rhythm	9 second windows	CART	37.68	0.38
Log-scale after 4 Hz		Simple Gaussian	18.74	0.38
		IBk	37.68	0.45
		J48	34.87	0.38
		LDA	18.49	1.96
		LDA Simple Gaussian	23.10	0.38
		MVCART0.001	**41.75**	**0.67**
		MVCART0.0001	40.86	0.48
		SMO Poly1	26.38	0.56
Rhythm	9 second windows	CART	28.70	0.53
Log-scale after 4 Hz	Mean	LDA Simple Gaussian	24.66	0.32
		Simple Gaussian	20.86	0.30
		IBk	33.64	0.27
		J48	28.99	0.17
		LDA	20.32	1.52
		MVCART0.001	**34.19**	**1.18**
		SMO Poly1	25.27	0.55
Rhythm	9 second windows	CART	28.71	0.31
Log-scale after 4 Hz	Mean and variance	LDA Simple Gaussian	26.20	0.43
		Simple Gaussian	19.97	0.35
		IBk	33.69	0.14
		J48	29.16	0.37
		LDA	18.36	1.26
		MVCART0.001	**34.33**	**0.48**
		SMO Poly1	25.49	0.57

Appendix B

Retrieval Results

B.1 Pseudo-Objective Statistics

Table B.1 shows the pseudo-objective statistics for each search system. The statistics include the 'neighbourhood clustering' produced by each system, the percentage of orphans (tracks that were not recommended for any query), the size of the worst hub (track appearing in top N results for most number of queries) and the percentage of track tuples for which the triangular inequality holds.

Table B.1: Pseudo-objective statistics of search results

Feature set	Retrieval system	Evaluation metric	Result level 5	10	20	50	-
Random baseline	Random baseline	album matches	0.15%	0.15%	0.15%	0.15%	-
		artist matches	7.74%	7.74%	7.74%	7.74%	-
		genre matches	9.07%	9.07%	9.07%	9.07%	-
		Triangular ineq. holds	-	-	-	-	-
Timbre frames	KL divergence	album matches	15.06%	10.72%	7.22%	4.13%	-
		artist matches	39.52%	36.05%	32.92%	29.55%	-
		artist-filtered genre	33.30%	31.36%	29.33%	26.70%	-
		Worst 'hub'	61	90	152	316	-
		% orphans	16.15%	8.61%	4.28%	1.70%	-
		Triangular ineq. holds	-	-	-	-	40.009%
Timbre frames mean and covariance	Euclidean distance	album matches	5.49%	4.07%	3.00%	1.93%	-
		artist matches	25.65%	23.25%	21.16%	18.56%	-
		artist-filtered genre	24.64%	23.32%	21.90%	20.15%	-
		Worst 'hub'	34	61	93	195	-
		% orphans	10.66%	4.97%	2.29%	0.81%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre frames mean and covariance	LDA Euclidean distance	album matches	5.38%	4.57%	3.65%	2.58%	-
		artist matches	30.59%	29.74%	28.81%	27.69%	-
		artist-filtered genre	35.84%	34.98%	33.83%	31.74%	-
		Worst 'hub'	23	45	82	196	-
		% orphans	6.79%	2.51%	1.06%	0.34%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre frames mean and covariance	SMO likelihoods Euclidean distance	album matches	2.96%	2.45%	2.06%	1.62%	-
		artist matches	27.86%	27.24%	26.90%	26.28%	-
		artist-filtered genre	29.37%	28.42%	26.70%	24.87%	-
		Worst 'hub'	59	69	93	171	-
		% orphans	13.61%	6.92%	2.51%	0.02%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre frames mean and covariance	SMO likelihoods Cosine distance	album matches	2.94%	2.46%	2.07%	1.64%	-
		artist matches	27.82%	27.26%	26.94%	26.35%	-
		artist-filtered genre	29.20%	28.14%	26.61%	24.91%	-
		Worst 'hub'	59	66	93	170	-
		% orphans	13.51%	6.89%	2.56%	0.05%	-
		Triangular ineq. holds	-	-	-	-	49.096%
Timbre frames mean and covariance	LDA likelihoods Euclidean distance	album matches	0.76%	0.72%	0.76%	0.93%	-
		artist matches	0.87%	0.70%	4.83%	15.05%	-
		artist-filtered genre	18.60%	22.26%	25.22%	27.62%	-

Continued on next page

Table B.1 – continued from previous page

Feature set	Retrieval system	Evaluation metric	Result level 5	10	20	50	-
		Worst 'hub'	506	506	506	506	-
		% orphans	98.38%	97.02%	94.31%	86.19%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre frames mean and covariance	LDA likelihoods Cosine distance	album matches	0.76%	0.72%	0.76%	0.93%	-
		artist matches	0.87%	0.70%	4.83%	15.05%	-
		artist-filtered genre	18.60%	22.26%	25.22%	27.62%	-
		Worst 'hub'	506	506	506	506	-
		% orphans	98.38%	97.02%	94.31%	86.19%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre events	MVCART likelihoods Euclidean distance	album matches	5.73%	4.77%	3.86%	2.73%	-
		artist matches	30.90%	30.23%	29.50%	28.48%	-
		artist-filtered genre	32.66%	30.98%	29.73%	28.76%	-
		Worst 'hub'	14	29	49	113	-
		% orphans	3.00%	0.96%	0.34%	0.15%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre events	MVCART likelihoods Cosine distance	album matches	5.99%	4.91%	3.94%	2.77%	-
		artist matches	31.07%	30.24%	29.48%	28.41%	-
		artist-filtered genre	32.74%	30.73%	29.48%	28.51%	-
		Worst 'hub'	15	32	57	120	-
		% orphans	3.22%	1.01%	0.32%	0.12%	-
		Triangular ineq. holds	-	-	-	-	56.43%
Timbre events	KL divergence	album matches	14.62%	10.36%	6.94%	3.95%	-
		artist matches	39.21%	35.73%	32.70%	29.53%	-
		artist-filtered genre	31.29%	29.53%	27.44%	24.90%	-
		Worst 'hub'	87	131	206	425	-
		% orphans	19.69%	12.08%	6.45%	2.49%	-
		Triangular ineq. holds	-	-	-	-	37.589%
Timbre events mean and covariance	Euclidean distance	album matches	4.44%	3.34%	2.46%	1.65%	-
		artist matches	23.24%	21.08%	18.96%	16.67%	-
		artist-filtered genre	23.15%	21.58%	20.29%	18.50%	-
		Worst 'hub'	39	63	112	214	-
		% orphans	11.62%	5.49%	2.54%	0.69%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre events mean and covariance	LDA Euclidean distance	album matches	4.12%	3.62%	3.00%	2.21%	-
		artist matches	28.91%	28.38%	27.56%	26.26%	-
		artist-filtered genre	34.29%	33.44%	32.24%	30.41%	-
		Worst 'hub'	22	40	76	190	-
		% orphans	6.84%	2.81%	0.96%	0.37%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre events	MVCART 738 terms - TF	album matches	9.74%	7.63%	5.77%	3.72%	-
		artist matches	33.80%	32.20%	30.67%	28.81%	-
		artist-filtered genre	31.35%	30.03%	29.28%	28.04%	-

Continued on next page

Table B.1 – continued from previous page

Feature set	Retrieval system	Evaluation metric	Result level 5	10	20	50	-
		Worst 'hub'	25	41	71	162	-
		% orphans	8.61%	2.46%	0.49%	0.00%	-
		Triangular ineq. holds	-	-	-	-	99.995%
Timbre events	MVCART 738 terms - TF/IDF	album matches	9.68%	7.67%	5.83%	3.73%	-
		artist matches	33.97%	32.37%	30.82%	28.84%	-
		artist-filtered genre	31.37%	30.18%	29.40%	28.11%	-
		Worst 'hub'	27	39	73	162	-
		% orphans	8.74%	2.34%	0.42%	0.00%	-
		Triangular ineq. holds	-	-	-	-	99.995%
Timbre events	MVCART 2048 terms - TF	album matches	10.61%	8.21%	6.11%	3.92%	-
		artist matches	34.99%	33.17%	31.35%	29.22%	-
		artist-filtered genre	31.33%	30.50%	29.57%	28.32%	-
		Worst 'hub'	33	49	93	184	-
		% orphans	9.70%	2.76%	0.47%	0.02%	-
		Triangular ineq. holds	-	-	-	-	100.00%
Timbre events	MVCART 2048 terms - TF/IDF	album matches	10.78%	8.32%	6.23%	3.99%	-
		artist matches	35.10%	33.19%	31.40%	29.22%	-
		artist-filtered genre	31.28%	30.42%	29.36%	28.19%	-
		Worst 'hub'	31	49	89	175	-
		% orphans	9.03%	2.81%	0.34%	0.02%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre events	MVCART 4091 terms - TF	album matches	10.33%	8.08%	6.04%	3.80%	-
		artist matches	34.45%	32.75%	31.10%	28.97%	-
		artist-filtered genre	31.27%	30.07%	29.09%	27.88%	-
		Worst 'hub'	30	44	79	150	-
		% orphans	8.81%	2.22%	0.25%	0.00%	-
		Triangular ineq. holds	-	-	-	-	99.998%
Timbre events	MVCART 4091 terms - TF/IDF	album matches	10.28%	8.05%	6.04%	3.84%	-
		artist matches	34.33%	32.70%	31.12%	29.01%	-
		artist-filtered genre	31.34%	30.15%	29.21%	27.98%	-
		Worst 'hub'	26	46	85	151	-
		% orphans	8.69%	2.26%	0.22%	0.00%	-
		Triangular ineq. holds	-	-	-	-	99.998%
Timbre events	MVCART 6210 terms - TF	album matches	10.17%	7.85%	5.95%	3.91%	-
		artist matches	34.73%	32.71%	31.09%	29.05%	-
		artist-filtered genre	31.18%	30.38%	29.63%	28.44%	-
		Worst 'hub'	40	62	108	197	-
		% orphans	11.99%	3.35%	0.47%	0.00%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Timbre events	MVCART 6210 terms - TF/IDF	album matches	10.44%	8.19%	6.15%	3.98%	-
		artist matches	34.95%	32.98%	31.20%	29.09%	-
		artist-filtered genre	31.10%	30.18%	29.56%	28.28%	-

Continued on next page

Table B.1 – continued from previous page

Feature set	Retrieval system	Evaluation metric	Result level 5	10	20	50	-
		Worst 'hub'	42	68	119	192	-
		% orphans	10.88%	3.05%	0.37%	0.00%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Rhythm log scale	MVCART 602 terms - TF	album matches	1.49%	1.27%	1.16%	1.06%	-
		artist matches	18.68%	18.00%	16.62%	15.44%	-
		artist-filtered genre	24.06%	23.81%	23.20%	22.41%	-
		Worst 'hub'	467	626	807	960	-
		% orphans	17.67%	7.58%	4.45%	2.09%	-
		Triangular ineq. holds	-	-	-	-	99.924%
Rhythm log scale	MVCART 602 terms - TF/IDF	album matches	1.47%	1.31%	1.21%	1.08%	-
		artist matches	19.71%	19.27%	18.52%	17.78%	-
		artist-filtered genre	23.61%	23.33%	22.83%	22.17%	-
		Worst 'hub'	29.20%	34.80%	40.70%	57.50%	-
		% orphans	11.52%	5.29%	3.03%	1.03%	-
		Triangular ineq. holds	-	-	-	-	99.976%
Rhythm log scale	MVCART 1555 terms - TF	album matches	1.25%	1.11%	1.09%	0.99%	-
		artist matches	17.53%	16.46%	14.98%	16.05%	-
		artist-filtered genre	24.21%	23.74%	23.02%	21.51%	-
		Worst 'hub'	521	697	874	1,029	-
		% orphans	16.37%	7.73%	3.72%	1.99%	-
		Triangular ineq. holds	-	-	-	-	99.955%
Rhythm log scale	MVCART 1555 terms - TF/IDF	album matches	1.24%	1.14%	1.10%	1.00%	-
		artist matches	18.99%	18.47%	17.97%	17.64%	-
		artist-filtered genre	23.58%	22.99%	22.63%	21.44%	-
		Worst 'hub'	304	353	460	683	-
		% orphans	11.69%	5.09%	3.08%	1.35%	-
		Triangular ineq. holds	-	-	-	-	99.986%
Rhythm log scale	mean Euclidean distance	album matches	2.26%	1.97%	1.65%	1.23%	-
		artist matches	20.87%	20.31%	19.64%	18.91%	-
		artist-filtered genre	20.65%	19.91%	19.06%	17.64%	-
		Worst 'hub'	21	31	57	141	-
		% orphans	3.37%	0.79%	0.17%	0.05%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Rhythm log scale	mean LDA Euclidean distance	album matches	1.38%	1.13%	0.92%	0.68%	-
		artist matches	19.08%	18.48%	17.92%	16.97%	-
		artist-filtered genre	15.11%	13.83%	12.91%	11.60%	-
		Worst 'hub'	17	30	58	133	-
		% orphans	2.98%	0.89%	0.25%	0.10%	-
		Triangular ineq. holds	-	-	-	-	100.000%
Rhythm log scale after 4Hz	MVCART 352 terms - TF	album matches	1.25%	1.18%	1.06%	0.94%	-
		artist matches	18.01%	17.73%	17.09%	16.53%	-
		artist-filtered genre	21.24%	21.14%	20.62%	19.42%	-

Continued on next page

Table B.1 – concluded from previous page

Feature set	Retrieval system	Evaluation metric	Result level 5	10	20	50	-
		Worst 'hub'	958	1,129	1,401	1,855	-
		% orphans	29.34%	20.50%	16.32%	12.50%	-
		Triangular ineq. holds	-	-	-	-	98.37%
Rhythm log scale after 4Hz	MVCART 352 terms - TF/IDF	album matches	1.26%	1.12%	1.04%	0.98%	-
		artist matches	18.98%	18.82%	18.89%	18.60%	-
		artist-filtered genre	21.03%	20.79%	20.51%	20.00%	-
		Worst 'hub'	649	664	728	1062	-
		% orphans	26.11%	18.68%	16.34%	12.72%	-
		Triangular ineq. holds	-	-	-	-	93.12%
Rhythm log scale after 4Hz	MVCART 1079 terms - TF	album matches	1.24%	1.20%	1.11%	0.98%	-
		artist matches	17.42%	17.30%	17.12%	16.76%	-
		artist-filtered genre	22.54%	21.24%	20.56%	18.90%	-
		Worst 'hub'	969	1,216	1,521	1,937	-
		% orphans	27.69%	18.02%	13.34%	10.14%	-
		Triangular ineq. holds	-	-	-	-	98.86%
Rhythm log scale after 4Hz	MVCART 1079 terms - TF/IDF	album matches	1.14%	1.10%	1.07%	1.02%	-
		artist matches	19.00%	19.20%	19.26%	19.08%	-
		artist-filtered genre	21.87%	21.02%	20.76%	19.82%	-
		Worst 'hub'	567	640	833	1,265	-
		% orphans	21.24%	14.92%	12.45%	9.01%	-
		Triangular ineq. holds	-	-	-	-	99.86%
Rhythm log scale after 4Hz	mean Euclidean distance	album matches	2.13%	1.75%	1.51%	1.21%	-
		artist matches	19.68%	18.80%	18.13%	17.18%	-
		artist-filtered genre	18.25%	18.15%	17.70%	16.82%	-
		Worst 'hub'	15	26	42	92	-
		% orphans	2.46%	0.62%	0.12%	0.02%	-
		Triangular ineq. holds	-	-	-	-	100.00%
Rhythm log scale after 4Hz	mean LDA Euclidean distance	album matches	1.80%	1.43%	1.12%	0.81%	-
		artist matches	19.41%	18.65%	17.89%	16.88%	-
		artist-filtered genre	16.90%	15.97%	14.64%	13.02%	-
		Worst 'hub'	15	30	55	128	-
		% orphans	3.10%	0.89%	0.17%	0.05%	-
		Triangular ineq. holds	-	-	-	-	100.00%

Appendix C

Combined Retrieval System Results

C.1 Pseudo-Objective Statistics

Table C.1 shows the pseudo-objective statistics for each search system based on the combination of other search systems. The statistics include the 'neighbourhood clustering' produced by each system, the percentage of orphans (tracks that were not recommended for any query), the size of the worst hub (track appearing in top N results for most number of queries) and the percentage of track tuples for which the triangular inequality holds.

Table C.1: Pseudo-objective statistics of search results for combined search systems

Feature sets	Retrieval systems	Evaluation metric	Result level 5	10	20	50	-
Random baseline	Random baseline	album matches	0.15%	0.15%	0.15%	0.15%	-
		artist matches	7.74%	7.74%	7.74%	7.74%	-
		genre matches	9.07%	9.07%	9.07%	9.07%	-
		Triangular ineq. holds	-	-	-	-	-
	Product combination of:	album matches	2.74%	2.27%	1.81%	1.29%	-
Timbre frames	KL divergence	artist matches	20.90%	20.19%	19.16%	18.11%	-
Rhythm log scale	MVCART transcription - 602 terms - TF/IDF	artist-filtered genre	23.06%	22.92%	22.38%	21.53%	-
		Worst 'hub'	59	110	198	458	-
		% orphans	6.74%	1.11%	0.12%	0.00%	-
		Triangular ineq. holds	-	-	-	-	99.97%
	Rank normalised product combination of:	album matches	6.67%	5.20%	3.92%	2.54%	-
Timbre frames	KL divergence	artist matches	31.40%	29.63%	27.67%	23.85%	-
Rhythm log scale	MVCART transcription - 602 terms - TF/IDF	artist-filtered genre	32.17%	30.59%	29.22%	27.04%	-
		Worst 'hub'	74	155	286	577	-
		% orphans	14.00%	6.55%	2.49%	0.71%	-
		Triangular ineq. holds	-	-	-	-	91.13%
	Weight average (0.7, 0.3) combination of:	album matches	2.75%	2.28%	1.81%	1.30%	-
Timbre frames	KL divergence	artist matches	21.02%	20.30%	19.31%	18.28%	-
Rhythm log scale	MVCART transcription - 602 terms - TF/IDF	artist-filtered genre	23.15%	22.99%	22.46%	21.65%	-
		Worst 'hub'	59	110	198	459	-
		% orphans	6.82%	1.06%	0.12%	0.00%	-
		Triangular ineq. holds	-	-	-	-	99.70%
	Product combination of:	album matches	6.82%	5.45%	3.99%	2.48%	-
Timbre events	MVCART transcription - 2048 terms - TF/IDF	artist matches	32.06%	30.51%	28.72%	26.14%	-
Rhythm log scale	MVCART transcription - 602 terms - TF/IDF	artist-filtered genre	32.81%	31.43%	30.00%	27.58%	-
		Worst 'hub'	40	71	112	196	-
		% orphans	4.23%	0.79%	0.07%	0.00%	-
		Triangular ineq. holds	-	-	-	-	100.00%
	Rank normalised product combination of:	album matches	6.05%	5.04%	3.91%	2.57%	-
Timbre events	MVCART transcription - 2048 terms - TF/IDF	artist matches	30.96%	29.83%	28.15%	25.11%	-
Rhythm log scale	MVCART transcription - 602 terms - TF/IDF	artist-filtered genre	33.52%	32.18%	30.82%	29.02%	-
		Worst 'hub'	96	123	205	437	-
		% orphans	6.60%	1.90%	0.37%	0.02%	-
		Triangular ineq. holds	-	-	-	-	89.65%
	Weight average (0.7, 0.3) combination of:	album matches	10.16%	8.14%	6.14%	3.88%	-
Timbre events	MVCART transcription - 2048 terms - TF/IDF	artist matches	34.89%	33.05%	31.28%	29.03%	-
Rhythm log scale	MVCART transcription - 602 terms - TF/IDF	artist-filtered genre	33.45%	32.64%	31.68%	30.46%	-

Continued on next page

Table C.1 – concluded from previous page

Feature sets	Retrieval systems	Evaluation metric	Result level 5	10	20	50	-
		Worst 'hub'	44	79	124	214	-
		% orphans	8.29%	1.70%	0.17%	0.00%	-
		Triangular ineq. holds	-	-	-	-	100.00%
	Combined feature vectors	album matches	5.62%	4.64%	3.70%	2.64%	-
Timbre frames	mean (rhythm) - LDA - Euclidean distance	artist matches	31.02%	30.08%	29.08%	27.90%	-
Rhythm log scale		artist-filtered genre	36.34%	35.32%	34.20%	32.39%	-
		Worst 'hub'	21	40	77	187	-
		% orphans	7.19%	2.83%	1.18%	0.37%	-
		Triangular ineq. holds					100.00%
	Combined feature vectors	album matches	5.35%	4.25%	3.25%	2.16%	-
Timbre frames	mean (rhythm) - Euclidean distance	artist matches	26.96%	24.97%	23.24%	21.08%	-
Rhythm log scale		artist-filtered genre	26.84%	25.86%	24.54%	22.87%	-
		Worst 'hub'	25	42	79	180	-
		% orphans	7.58%	3.17%	1.21%	0.27%	-
		Triangular ineq. holds	-	-	-	-	100.00%

References

[1] Chris Anderson. The long tail. http://www.thelongtail.com, April 2006. 1, 9

[2] J.-J Aucouturier. *Ten experiments on the modelling of polyphonic timbre*. PhD thesis, University of Paris 6, France, June 2006. 22, 35, 124, 125, 144, 147, 171

[3] J-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In *Proceedings of ISMIR 2002 Third International Conference on Music Information Retrieval*, September 2002. 1, 11, 12, 41, 54, 129

[4] J.J. Aucouturier and F. Pachet. Improving timbre similarity: How high is the sky. *Journal of Negative Results in Speech and Audio Sciences*, **1**(1):1–13, 2004. 54

[5] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing Company, 1999. 70, 72

[6] JP Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and MB Sandler. A Tutorial on Onset Detection in Music Signals. *Speech and Audio Processing, IEEE Transactions on*, **13**(5):1035–1047, 2005. 37

[7] J.P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of ISMIR 2005 Sixth International Conference on Music Information Retrieval*, 2005. 35

[8] K.P. Bennett and C. Campbell. Support vector machines: hype or hallelujah? *ACM SIGKDD Explorations Newsletter*, **2**(2):1–13, 2000. 73

[9] M.L. Berenson, D.M. Levine, and M. Goldstein. *Intermediate statistical methods and applications: a computer package approach*. Prentice-Hall, 1983. 79, 158

[10] A. Berenzweig, D. Ellis, and S. Lawrence. Anchor space for classification and similarity measurement of music. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 2003. 27, 132

[11] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kegl. Meta-features and adaboost for music classification. *Machine Learning Journal: Special Issue on Machine Learning in Music*, 2006. 54, 55

[12] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and Regression Trees*. Wadsworth and Brooks/Cole Advanced books and Software, 1984. 58, 59

[13] Paul Brossier. The aubio library at mirex 2006. http://www.music-ir.org/evaluation/MIREX/2006_abstracts/AME_BT_OD_TE_brossier.pdf, October 2006. 38

[14] J.J. Burred and A. Lerch. A hierarchical approach to automatic musical genre classification. In *Proc. of the International Conf. on Digital Audio Effects*, 2003. 54

[15] M. Casey and M. Slaney. Fast Recognition of Remixed Music Audio. *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, **4**, 2007. 24

[16] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 426–435, 1997. 127, 128, 144

[17] W.J. Conover. *Practical Nonparametric Statistics*. [Sl: sn, 1980. 80

[18] M. Cooper and J. Foote. Automatic music summarization via similarity analysis. In *Proceedings of ISMIR 2002 Third International Conference on Music Information Retrieval*, September 2002. 16

[19] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, **20**(3):273–297, 1995. 74

[20] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **ASSP-28, No. 4**:357–366, August 1980. 27

[21] S. Dixon. Onset Detection Revisited. In *Proceedings of the 9th International Conference on Digital Audio Effects, Montreal.*, pages 133–137, september 2006. 37, 38

[22] Simon Dixon. A beat tracking system for audio signals. In *in Proceedings of the 1999 Diderot Forum on Mathematics and Music, Austrian Computer Society.*, pages 101–110. Austrian Computer Society, 1999. 37

[23] Simon Dixon. Simple spectrum-based onset detection. http://www.music-ir.org/evaluation/MIREX/2006_abstracts/OD_dixon.pdf, October 2006. 38

[24] Simon Dixon, Elias Pampalk, and Gerhard Widmer. Classification of dance music by periodicity patterns. In *Proceedings of the Fourth International Conference on Music Information Retrieval (ISMIR) 2003*, pages 159–166, Austrian Research Institute for AI, Freyung 6/6, Vienna 1010, Austria, 2003. 43, 48, 53, 55, 166, 170

[25] J. Stephen Downie. Evaluating a simple approach to music information retrieval, 1999. 1, 10

[26] J. Stephen Downie, Paul Brossier, and Pierre Leveau. Mirex2006: Audio onset detection. http://www.music-ir.org/mirexwiki/index.php/Audio_Onset_Detection, October 2006. 37

[27] J. Stephen Downie, Kris West, Paul Lamere, and Elias Pampalk. Mirex2006: Audio music similarity and retrieval. http://www.music-ir.org/mirex2006/index.php/Audio_Music_Similarity_and_Retrieval, October 2006. 22, 81, 128, 129, 144, 170

[28] J.S. Downie. The Music Information Retrieval Evaluation eXchange (MIREX). *D-Lib Magazine*, **12**(12):1082–9873, 2006. 20, 55, 81, 170

[29] Yunfeng Du, Ming Li, and Jian Liu. Mirex 2006: Spectral-flux based musical onset detection. http://www.music-ir.org/evaluation/MIREX/2006_abstracts/OD_du.pdf, October 2006. 38

[30] A. Eronen. Chorus detection with combined use of mfcc and chroma features and image processing filters. In *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx-07)*, September 2007. 16

[31] L Gillick and Stephen Cox. Some statistical issues in the comparison of speech recognition algorithms. In *IEEE Conference on Accoustics, Speech and Signal Processing*, pages 532–535, 1989. 82

[32] F. Gouyon and S. Dixon. Dance music classification: A tempo-based approach. In *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR) 2004*, pages 501–504, 2004. 166, 170

[33] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, August 1999. 1, 8

[34] D. Huron and B. Aarden. Perceptual and cognitive applications in music information retrieval. In *Proceedings of ISMIR 2000, 1st International Symposium on Music Information Retrieval*, October 2000. 15

[35] J.H. Jensen, MG Christensen, DPW Ellis, and S.H. Jensen. A TEMPO-INSENSITIVE DISTANCE MEASURE FOR COVER SONG IDENTIFICATION BASED ON CHROMA FEATURES. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2209–2212, 2008. 170

[36] D.-N. Jiang, L. Lu, H.-J. Zhang, J.-H. Tao, and L.-H. Cai. Music type classification by spectral contrast feature. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 2002. 29

[37] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. Technical report, Department of Computer Science and Technology, Tsinghua University, China and Microsoft Research, Asia, 2002. 34

[38] M.C. Jones, J.S. Downie, and A.F. Ehmann. "human similarity judgements: Implications for the design of formal evaluations". In *Proceedings of ISMIR 2007 International Society of Music Information Retrieval*, 2007. 7, 20, 21

[39] Igor Karpov. Hidden Markov classification for musical genres. Technical report, Rice University, 2002. 54

[40] SS Keerthi, SK Shevade, C. Bhattacharyya, and KRK Murthy. Improvements to Platt's SMO Algorithm for SVM Classifier Design, 2001. 74

[41] Y.E. Kim, D.S. Williamson, and S. Pilli. Towards quantifying the album effect in artist identification. In *Proceedings of ISMIR 2006 Seventh International Conference on Music Information Retrieval*, September 2006. 18, 145

[42] L. Kuncheva. *Combining Pattern Classifiers, Methods and Algorithms*. Wiley-Interscience, 2004. 58, 61, 83, 106, 109, 110, 112, 113

[43] M. Levy and M. Sandler. Lightweight measures for timbral similarity of musical audio. *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 27–36, 2006. 125, 126, 127, 169

[44] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289, 2003. 34

[45] T. Lidy, A. Rauber, A. Pertusa, and J. M. Iesta. Improving genre classification by combination of audio and symbolic descriptors using a transcription system. In *Proceedings of ISMIR 2007 Sixth International Conference on Music Information Retrieval*, September 2007. 14, 34, 54

[46] B. Logan. Music summarization using key phrases. In *Proceedings of the International Conference on Audio, Speech and Signal Processing, ICASSP.*, 2000. 17

[47] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, August 2001. 1, 10, 21, 27, 34, 41, 125, 128, 143

[48] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the First International Symposium on Music Information Retrieval (ISMIR)*, October 2000. 27

[49] M. Mandel and D. Ellis. Song-Level Features and Support Vector Machines for Music Classification. In *Proceedings of ISMIR 2005 Sixth International Conference on Music Information Retrieval*, 2005. 34, 54, 55

[50] Michael Mandel and Daniel Ellis. Song-level features and support vector machines for music classification. http://www.music-ir.org/evaluation/mirex-results/articles/audio_genre/mandel.pdf, October 2005. 41, 125

[51] Keith D Martin, Eric D Scheirer, and Barry L Vercoe. Music content analysis through models of audition. Technical report, MIT Media Laboratory Machine Listening Group, Cambridge MA USA, September 1998. 27

[52] C. McKay. *Automatic genre classification of MIDI recordings.* Master's thesis, McGill University, Canada., 2004. 10

[53] Martin F McKinney and Jeroen Breebaart. Features for audio and music classification. In *Proceedings of the Fourth International Conference*

on Music Information Retrieval (ISMIR) 2003, pages 151–158, Philips Research Laboratories, Prof. Holstlaan 4 (WY82, 5656 AA Eindhoven, The Netherlands, 2003. 27, 54

[54] A.V. Oppenheim and R.W. Schafer. *Discrete-time signal processing*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1989. 48, 49

[55] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Johannes Kepler University, Linz, March 2006. 18, 27, 34, 41, 45, 46, 54, 76, 125, 128, 144, 147

[56] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *Proceedings of ISMIR 2005 Sixth International Conference on Music Information Retrieval*, September 2005. 1, 11, 19, 34, 54, 133, 145

[57] J. Paulus and A. Klapuri. Measuring the similarity of rhythmic patterns. *Proc. ISMIR*, **2002**, 2002. 44

[58] G. Peeters. Musical key estimation of audio signal based on hidden Markov modeling of chroma vectors. In *Proceedings of the Ninth International Conference on Digital Audio Effects (DAFx-06)*, pages 127–131, 2006. 170

[59] J.C. Platt. 12 Fast Training of Support Vector Machines Using Sequential Minimal Optimization. *Advances in Kernel Methods: Support Vector Learning*, 1999. 74

[60] Axel Robel. Onset detection in polyphonic signals by means of transient peak classification. http://www.music-ir.org/evaluation/MIREX/2006_abstracts/OD_roebel.pdf, October 2006. 38

[61] A.I. Schein, A. Popescul, L.H. Ungar, and D.M. Pennock. Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, 2002. 3

[62] Paul Scott. Music classification using neural networks. Technical report, Stanford University, Stanford, CA 94305, 2001. 54

[63] J. Tague-Sutcliffe and J. Blustein. A Statistical Analysis of the TREC-3 Data. *Overview of the Third Text Retrieval Conference (Trec-3)*, 1995. 81, 158

[64] Rainer Typke. *Music Retrieval based on Melodic Similarity*. PhD thesis, Utrecht University, Netherlands, February 2007. 1, 10

[65] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, **10**(5):293–302, July 2002. 12, 27, 31, 44, 54, 67

[66] George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals. In *Proceedings of ISMIR 2001: The International Conference on Music Information Retrieval and Related Activities*, 2001. 27, 31, 34, 41, 54

[67] Andrew Webb. *Statistical Pattern Recognition*. John Wiley and Sons, Ltd, 2002. 56, 57

[68] K. West and S. Cox. Features and classifiers for the automatic classification of musical audio signals. In *Proceedings of ISMIR 2004 Fifth International Conference on Music Information Retrieval*, 2004. 54

[69] K. West and S. Cox. Finding an optimal segmentation for audio genre classification. In *Proceedings of ISMIR 2005 Sixth International Conference on Music Information Retrieval*, September 2005. 36, 64, 165

[70] K. West, S. Cox, and P. Lamere. Incorporating machine-learning into music similarity estimation. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 89–96. ACM Press New York, NY, USA, 2006. 129

[71] K. West and P. Lamere. A model-based approach to constructing music similarity functions. *EURASIP Journal on Advances in Signal Processing*, pages 1–10, 2007. 21, 29, 64, 128, 132, 133, 143

[72] K. West, E. Pampalk, and P. Lamere. Mirex 2006 - audio music similarity and retrieval. http://www.music-ir.org/mirex2006/index.php/Audio_Music_Similarity_and_Retrieval, April 2006. 146

[73] T. Westergren. The music genome project. http://www.pandora.com/mgp.shtml. 1, 9

[74] I.H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S.J. Cunningham. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. *ICONIP/ANZIIS/ANNES*, pages 192–196, 1999. 74

[75] Changsheng Xu, Namunu C Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical genre classification using support vector machines. Technical report, Laboratories for Information Technology, 21 Heng Mui Keng Terrace, Singapore 119613, 2003. 34, 54