Music-Inspired Texture Representation

Ben Horsburgh and **Susan Craw** and **Stewart Massie**
IDEAS Research Institute
Robert Gordon University, Aberdeen, UK

Abstract

Techniques for music recommendation are increasingly relying on hybrid representations to retrieve new and exciting music. A key component of these representations is musical content, with texture being the most widely used feature. Current techniques for representing texture however are inspired by speech, not music, therefore music representations are not capturing the correct nature of musical texture. In this paper we investigate two parts of the well-established mel-frequency cepstral coefficients (MFCC) representation: the resolution of mel-frequencies related to the resolution of musical notes; and how best to describe the shape of texture. Through contextualizing these parts, and their relationship to music, a novel music-inspired texture representation is developed. We evaluate this new texture representation by applying it to the task of music recommendation. We use the representation to build three recommendation models, based on current state-of-the-art methods. Our results show that by understanding two key parts of texture representation, it is possible to achieve a significant recommendation improvement. This contribution of a music-inspired texture representation will not only improve content-based representation, but will allow hybrid systems to take advantage of a stronger content component.

Introduction

Over the last decade the way in which people find and enjoy music has completely changed. Traditionally a listener would first hear about new music either through their friends or mass-media, such as radio and magazines. They would then visit a record store and buy a hard copy of the music. It is now equally as likely that a listener will have been recommended new music by an algorithm, and that they will either stream or buy a soft copy of the music. There are possibly millions of tracks on a website, with only a limited number being of interest to the user. This presents an interesting new challenge: how to decide which tracks should be recommended to a user.

Many current state-of-the-art techniques for providing music recommendations are based on the idea of similarity. Given one or more songs that the listener likes, recommender systems must provide further songs that the listener will like. One of the most popular ways to do this is to compare meta-data about the songs. Such meta-data often includes textual tags, and audio descriptors, which are combined into a hybrid representation for each song. When examining these hybrid representations it becomes clear that tags offer much greater accuracy than audio descriptors. For this reason a lot of current research is focussing on how such audio descriptors can be improved, and how these can be better integrated with tags.

There are two approaches to describing musical audio: in terms of its structure, such as rhythm, harmony and melody; or in terms of its feel, such as texture. The choice of which approach to use can differ depending on the task. However, in almost all tasks, texture has proven to be very popular and successful. The most widely used texture descriptor is the mel-frequency-cepstral-coefficient (MFCC) representation. Although originally developed for speech recognition, MFCCs have proven to be robust across many musical tasks, including recommendation. This foundation in speech processing however can make texture difficult to understand in a musical sense.

We take a novel approach to examining the MFCC representation in a musical context, and make two important observations. The first observation is related to the resolution used at a key step in the algorithm, and how this corresponds to music. The second observation is that in the case of music, summarising the perceptual spectrum used by MFCC is undesirable. Based on these observations we develop a novel approach to music-inspired texture representation, and evaluate its performance using the task of music recommendation. This evaluation also shows that no single fixed-parameter texture representation works best for all recommendation models.

The paper is structured as follows. We first discuss related work on the MFCC representation, how it has been used for music, and some observations others have made on its suitability to music. We then develop our new texture representation by examining the MFCC representation, and discussing key steps. The following section describes our experimental design, and the three recommendation approaches with which we evaluate our approach. We then discuss the results obtained from our experiments, and finally we draw some conclusions from this work.

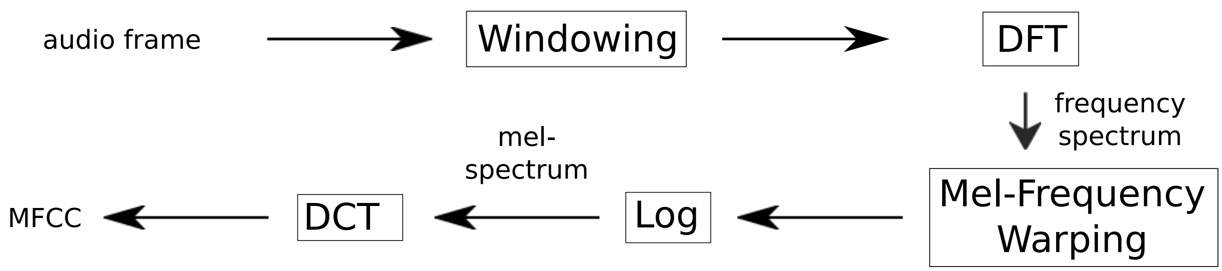

Figure 1: Block diagram of MFCC extraction algorithm

Related Work

One of the most important content-based representations for music recommendation is MFCC (Celma 2010). First developed for speech recognition (Mermelstein 1976), the MFCC representation is based on warping a frequency spectrum to human perception using the mel scale (Stevens, Volkmann, and Newman 1937).

One important difference between practical evaluations of MFCC for speech and music, is that speech can be evaluated in an objective manner (Zheng, Zhang, and Song 2001), while many music tasks are evaluated subjectively (Ellis et al. 2002). However, many authors have successfully evaluated the use of MFCC subjectively for genre classification (Lee et al. 2009) and music recommendation (Yoshii et al. 2008).

Logan (2000) investigates the suitability of the mel scale for musical texture, concluding that the scale is at least not harmful for speech / music discrimination, but perhaps further work is required to examine only music. Logan also concludes that at a theoretical level DCT is an appropriate transform to decorrelate the mel-frequency spectrum, but the parameters used require more thorough examination. We question whether the mel-frequency spectrum should be decorrelated at all, and evaluate the suitability of DCT for describing a music-inspired texture.

Several investigations have concluded that resolution is an important factor affecting MFCCs. It has been shown that computing MFCCs on MP3 audio is only robust if the audio is encoded at 128kbps or higher (Sigurdsson, Petersen, and Lehn-Schiler 2006). At lower encoding resolutions, the MFCCs extracted are not able to describe texture reliably. Pachet and Aucouturier (2004) investigate several parameters of MFCC and how they relate to music. Their first observation is closely linked to that of Sigurdsson, Petersen, and Lehn-Schiler; increasing the input resolution by increasing the sample rate improves the effectiveness of MFCC. In further work Aucouturier, Pachet, and Sandler (2005) examine the relationship between how many coefficients are retained after DCT, and the number of components used in a Gaussian Mixture Model. It is found that increasing the number of coefficients beyond a threshold is harmful for this model.

The MFCC representation involves smoothing the input data based on the mel scale, introducing a further step where resolution may be important. Most commonly 40 mel filters are used for smoothing, often described as 13 linear and 27 logarithmic filters, and 13 coefficients are retained after DCT. One popular toolbox for extracting MFCCs also uses these fixed parameters (Slaney 1998).

Yan, Zhou, and Li (2012) investigate the placement of mel filters for the application of Chinese speech recognition. When analysing MFCC across various regional accents, they discovered that the first two formants are most sensitive to accents. These formants are the lower-frequency section of the mel-spectrum, and so the authors re-distribute the mel filters used to provide a higher resolution in these critical bands.

Li and Chan (2011) also investigate the placement of mel filters while considering genre classification. The authors find that the MFCC representation is affected by the musical key that a track is played in. Rather than modifying the MFCC representation to resolve this issue, they instead normalise the key of each track before extracting MFCCs. In this paper we investigate the relationship between the mel scale and pitch, and develop our music-inspired texture representation from this.

Mel-Frequency Texture Representations

The MFCC representation attempts to describe the shape of a sound, with respect to how humans hear it. In this section we describe and examine this representation, and develop several modifications showing how this approach can be tailored to describe texture for music.

Frequency Spectrum

The MFCC representation is generated from a short time-sample of audio, as illustrated in Figure 1. The input is an audio frame of fixed sample rate and sample size. To reduce the effects of noise introduced by sampling, rather than having a continuous waveform, the initial step in Figure 1 is windowing. This windowed time-domain frame is then converted to a frequency-domain spectrum using the discrete-Fourier transform (DFT).

There are three properties of the frequency spectrum which are important:

- The size of the spectrum is half the size of the audio frame, and for computational efficiency is usually size 2^n.
- The maximum frequency is half the audio frame sampling rate.
- The spectrum bin resolution is the maximum frequency divided by the spectrum size.

For an audio frame with sample rate 44.1kHz and sample size 2^{10} (23ms), the size of the spectrum is 2^9, the maximum frequency is 22.05kHz, and the bin resolution is 43.1Hz.

The windowing and DFT steps are common to many content-based representations, and so this paper focusses on the later steps in Figure 1, which are more specific to texture.

Mel-Frequency Spectrum

The frequency spectrum output by the DFT describes the intensity of discrete frequency ranges. In order to correctly describe texture, one must warp this output so that it describes how humans hear frequencies and their intensities. This is achieved using two operations; frequency warping and intensity rescaling. Intensity rescaling, applied after the frequency warping, is achieved by taking the $\log$ of all frequency values. In this section we will focus on how the frequency warping is achieved.

Frequency is the physical scale describing sound oscillations, and must be warped into the perceptual mel scale (Stevens, Volkmann, and Newman 1937), thus mimicking humans interpret frequency. The following equation describes the conversion from frequency f to mel ϕ

$$\phi(f) = 2595 \log_{10}\left(\frac{f}{700} + 1\right) \tag{1}$$

and is illustrated by the curve in Figure 2. The horizontal axis is f, and the left vertical axis is $\phi(f)$.

Frequency warping is not achieved as a direct conversion from f to $\phi(f)$, but as a discretization of f based on $\phi(f)$. To avoid confusion, we will refer to frequency ranges defined by the DFT as bins, and to the ranges used for frequency warping as buckets.

M buckets are equally sized and spaced along $\phi(f)$, as illustrated by the triangles in Figure 2. The buckets on the f axis are equivalent to those on $\phi(f)$. The value of each bucket is a weighted sum of all its frequency bins, where the weights are defined by the triangular filter.

After discretization, the frequency spectrum has been warped into M mel-scaled values. After the $\log$ of each value has been taken, this set of values is known as the Mel-Frequency Spectrum (MFS).

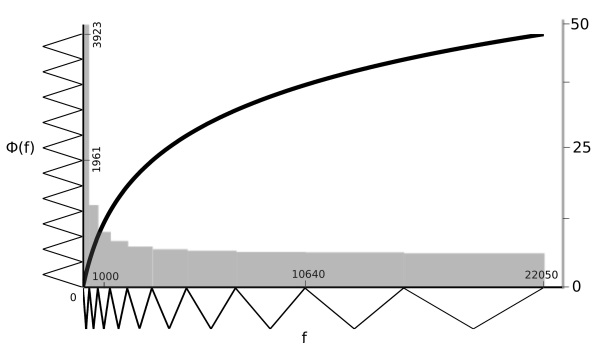

Figure 2: Frequency - mel curve

Mel-Frequency Filter Resolution

When MFCCs are discussed in the literature it is usually in terms of the frequency and mel scales. In this section we first make observations of the MFS in terms frequency and mel, and then in terms of a musical scale, in an attempt to examine its suitability for musical texture.

Continuing with our example using Figure 2, when 10 triangular mel-filters are used for frequency warping, each bucket is 392 mels wide. Converted to f, this means the smallest bucket is 291Hz, and the largest is 6688Hz. Examined as a percentage value, each bucket covers 10% of the mel scale, or between 0.013% and 30% of the frequency scale. At a first glance the smallest bucket perhaps seems too small, and the largest is perhaps too large, however, we must now examine the ranges from a musical point of view.

The most commonly used musical scale is the cents scale, which describes the distance in cents between two frequencies. The following equation shows how to calculate the number of cents between frequency f and reference frequency g.

$$c(f, g) = 1200 \log_2\left(\frac{f}{g}\right) \tag{2}$$

Using this scale, 100 cents is equal to one musical semitone, or adjacent piano key. For the scale to make sense g must correspond to the frequency of a musical note; for our examples we set $g = 16.35$, which is the musical note C. The lowest note on a standard piano is A at 27.5Hz.

If we consider our example of 10 filters again, but this time examining the size of the buckets in cents, we get a completely different picture. The smallest filter covers 4986 cents, which is just more than 49 musical notes, and the largest filter covers 626 cents, which is just more than 6 musical notes. This illustrates something completely different to examining discretization of f; musically the smallest bucket in f seems much too large in cents, and the largest bucket in f seems much too small in cents.

Figure 2 illustrates this conflict of resolution between f and cents. The light grey shading shows the number of musical notes for each triangular filter along f. The right vertical axis shows the scale for these bars. It is clear that the smallest triangular filter on f has the largest number of notes. As the filters on f become larger, the number of notes quickly becomes lower and flattens at around 6 notes.

There are 12477 cents between our reference frequency g and 22050Hz, which is just over 124 musical notes. Re-examining the discretization as a percentage, the smallest bucket covers 39.5% of the musical scale, and the largest bucket covers 0.048%.

In practice the most common value of M for MFCCs is 40. At this value the smallest filter covers 23.5 musical notes, and the largest covers 1 musical note. Intuitively the smallest filter still seems much too large. Most musical notes are played at a relatively low frequency; the higher frequencies often describe harmonics of the lower note. Texture should describe all the elements, which when put together form a sound. If 23.5 notes are all grouped together, one can argue these elements are too coarsely discretized, and thus the best texture description is not used.

We propose that M should be a more carefully considered parameter, and should be much larger than is typically used. There is one final consideration however; increasing M will decrease the size of the smallest filter. M is bounded by the point at which the size of the smallest filter becomes less than the resolution of the frequency spectrum, at which point duplicate and redundant information will be produced. The selection of M is therefore a balancing act between the resolution of the smallest and largest filters. As more filters are used, the resolution of higher filters will become much less than 1 musical note.

To denote the number of filters which are being used by any representation we will append the number in brackets. For example, when 60 filters are used the representation will be denoted as MFS(60).

Discrete Cosine Transform

The final step in Figure 1 is the Discrete Cosine Transform (DCT), which takes the mel-frequency spectrum as an input, and outputs a description of its shape, the mel-frequency cepstrum. The coefficients of this cepstrum are the MFCC representation.

The description provided by the DCT is a set of weights, each correlating to a specific cosine curve. When these curves are combined as a weighted sum, the mel-frequency spectrum is approximated. To calculate the mel-frequency cepstrum, X, the DCT is applied to the mel-frequency spectrum x as follows

$$X_n = \sum_{m=0}^{M-1} x_m \cos\left[\frac{\pi}{M}\left(m+\frac{1}{2}\right)n\right] \tag{3}$$

for $n = 0$ to $N-1$. Typically M is 40, and N is 13.

Discrete Cosine Transform Observations

The idea behind using DCT for texture representations comes from the speech processing domain. Speech has two key characteristics: formants are the meaningful frequency components which characterise a sound; and breath is the general noise throughout all frequency components, and thus much less meaningful.

For speech, DCT offers strong energy compaction. In most applications breath is undesirable, and so only the first few DCT coefficients need to be retained. This is because these coefficients capture low-frequency changes within the mel-frequency spectrum. The high frequency detail primarily characterised by noisy breath is not needed. Most commonly 13 coefficients are retained when 40 mel filters are used. Retaining 40 coefficients would give a very close approximation of the mel-frequency spectrum. Some authors do not retain the 0th, DC coefficient in their representation.

For music, the concepts of formants and breath do not apply. It is true that a vocalist is common, meaning formants and breath are present, however, music is also present. If only a few coefficients are retained from the DCT, then much information about percussion and the general feel of the music is lost. In music the mel-frequency spectrum is much more rich, and this detail is important to describing texture.

One could argue then that for music more coefficients should be retained from the DCT. The primary reason for using the DCT however still stems from the idea of separating formants and breath, or information from noise. We propose that the DCT should not be used for a mel-frequency based texture representation for music. All of the frequency information is potentially meaningful, and therefore should not be summarised. The special case against this argument is live music, where noise may still be present. However, for general music retrieval, most music is studio recorded, where a lot of effort is taken to remove all noise.

When no DCT is used for describing texture we denote the representation mel-frequency spectrum (MFS). When the DCT is used we denote the representation MFCC. When 40 mel filters are used we denote the representations as MFS(40) or MFCC(40).

Experiment Design

We evaluate how MFS performs at the task of music recommendation. We construct this as a query-by-example task, where 10 recommendations are provided for any given query. 10-fold cross validation is used, and our results are compared to those achieved by MFCC.

Dataset

The dataset we use consists of 3174 tracks by 764 artists, and are from 12 distinct super-genres. The most common genres are Alternative (29%) and Pop (25%). Each track in our collection is sampled at 44.1kHz, and processed using a non-overlapping samples of size of 2^{13} (186ms). Each frequency spectrum computed has a maximum frequency of 22.05kHz, and a bin resolution of 5.4Hz. For each sample we use a Hamming window before computing the DFT, and then extract each of the texture representations.

Each model is constructed using texture vectors, extracted from each sample in a given track. We extract texture vectors for both MFS and MFCC using 40, 60, 80, 100, and 120 filters. For 40 filters the smallest bucket contains just over 25 notes, and the largest contains just over 1 note. For 120 filters the smallest bucket contains 4 notes, and the largest contains 0.5 notes.

Recommendation Models

We construct three well-known models to avoid drawing conclusions specific to one type of recommender model.

Latent-Semantic-Indexing of Mean-Vector (LSA) - A mean texture vector is first computed for each track, where each dimension corresponds to the mean value across all of the track's texture vectors. We then construct a track-feature matrix using these mean texture vectors. The JAMA package (Hicklin et al. 2000) is used to generalise this matrix by LSI, and each query is projected into this generalised search space. Recommendations are made based on Euclidean distance as in previous work (Horsburgh et al. 2011).

Vocabulary-Based Model (VOC) - Vocabulary-Based methods are often found in hybrid recommender systems, and so we examine this popular model. To generate a vocabulary we use the k-means algorithm to cluster 20000 texture vectors selected at random from all tracks. For each track we count the number of samples which are assigned to each cluster, and construct a cluster-track matrix. A TF-IDF weighting is applied, and cosine similarity used to make recommendations.

Gaussian Mixture Model Approach (GMM) - A GMM models the distribution of a tracks' texture vectors as a weighted sum of K more simple Gaussian distributions, known as components. Each weighted component in the GMM is described by its mean texture vector and covariance matrix (Aucouturier, Pachet, and Sandler 2005). We learn a GMM for each track using the Weka implementation of the EM algorithm (Hall et al. 2009).

If each track were represented by a single Gaussian distribution, recommendations can be made using Kullback-Leibler divergence. With GMMs however each track is represented by K weighted Gaussian distributions, and so we make recommendations based on an approximation of Kullback-Leibler divergence (Hershey and Olsen 2007). For each component in a query track's GMM, we compute the minimum Kullback-Leibler divergence to each component in a candidate track's GMM. The estimated Kullback-Leibler divergence between the query track and the candidate recommendation is calculated as the weighted average of the minimum divergence between all components. Recommendations are ranked using this estimated Kullback-Leibler divergence.

Evaluation Method

Our evaluation method uses data collected from over $175,000$ user profiles, extracted from Last.fm using the AudioScrobbler API[1] over 2 months. For each user we record tracks which they have said they like. On average, each user in our collection likes 5.4 songs. To measure recommendation quality we use the association score that we developed in previous work (Horsburgh et al. 2011).

$$\text{association}(t_i, t_j) = \frac{\text{likes}(t_i, t_j)}{\text{listeners}(t_i, t_j)} \qquad (4)$$

where t_i and t_j are tracks, $\text{listeners}(t_i, t_j)$ is the number of people who have listened to both t_i and t_j, and $\text{likes}(t_i, t_j)$ is the number of listeners who have liked both t_i and t_j. The number of listeners is estimated using statistics from Last.fm, and assumes listens are independent. Using this evaluation measure allows us to understand the likelihood that someone who likes track t_i will also like t_j.

Results

We compare our MFS music-inspired texture with MFCCs for the task of music recommendation using 40 filters. We then investigate the effects of increasing the number of filters used, and compare the best MFS and MFCC representations for each model.

MFS vs MFCC

Figure 3 shows the results using MFS and MFCC for each model. The vertical axis shows the association score, and the horizontal axis shows the number of recommendations evaluated. Each value shown is the average association score at the number of recommendations made. All error bars show significance at a 95% confidence interval.

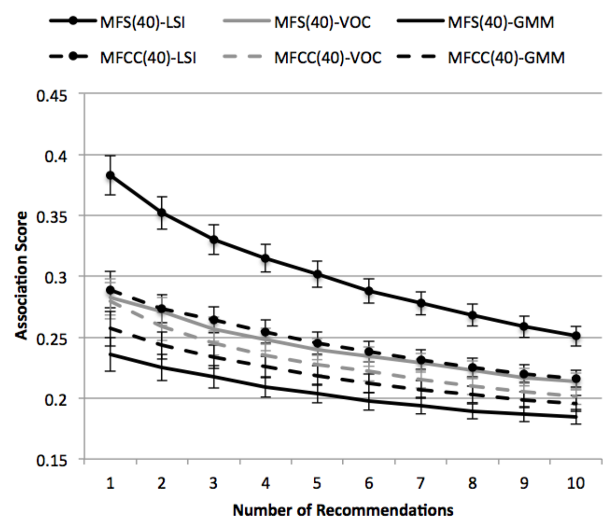

Figure 3: Comparison of MFS and MFCC using 40 filters

MFS-LSI achieves a significantly better association score than all other models and representations. MFS-VOC provides a significant quality increase over MFCC-VOC when 2 or more recommendations are made. MFS out performs MFCC with LSI and VOC because both models group data at the collection level; LSI finds associations between dimensions in the texture vectors, and VOC groups texture vectors into clusters. With MFCC, each texture vector is first summarised by DCT without respect to the collection, and therefore LSI and VOC are not able to model the data effectively.

Unlike LSI and VOC, for 40 filters MFCC-GMM is significantly better than MFS-GMM. The reason for this is that GMM behaves differently; each model is constructed for a single track. We learned each GMM using a diagonal covariance matrix, which has the effect of making dimensions in the texture vector independent. This means associations between dimensions are not learned.

Effect of Number of Filters

We want to explore the effect of increasing the number of filters on recommendation, and so examine a more simple recommendation task. Figure 4 shows the average association score of the top 3 recommendations for each model. The horizontal axis is grouped by the model used, and each bar corresponds to a given number of filters.

[1] http://www.last.fm/api

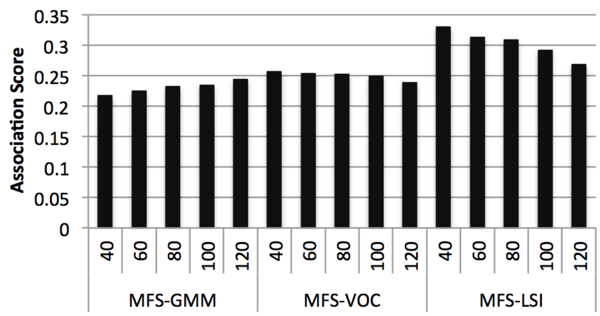

Figure 4: Effect of filters on MFS

Increasing the number of filters significantly increases the recommendation quality of MFS-GMM, does not significantly affect MFS-VOC, and significantly decreases the quality of MFS-LSI. GMM is improved because more independent information is available to the model, and so more meaningful distributions can be learned. VOC does not change because a similar vocabulary is formed regardless of the number of filters. The performance of LSI decreases, showing that the model can generalise a low number of filters more effectively. Figure 5 shows the effect changing the number of filters for MFCC. No correlation appears between the number of filters used and association score.

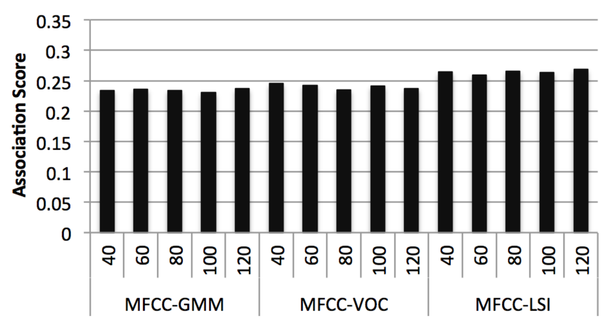

Figure 5: Effect of filters on MFCC

Figure 6 is in the same format as Figure 3, and shows MFS and MFCC when the best performing number of filters are used for each model. We do not show VOC because neither representation was improved by using more filters. Adding more filters improved the MFCC-LSI model, but is still outperformed by MFS-LSI.

The solid black line in Figure 6 shows MFS(40)-GMM, and the solid grey line shows MFS(120)-GMM. The MFS(40)-GMM results are included in the Figure to illustrate the improved recommendation quality achieved by increasing the number of filters for MFS-GMM. For the first 5 recommendations, MFS-GMM is now significantly better than MFCC-GMM. In comparison, there is only a small improvement of MFCC-GMM through increasing the filters used.

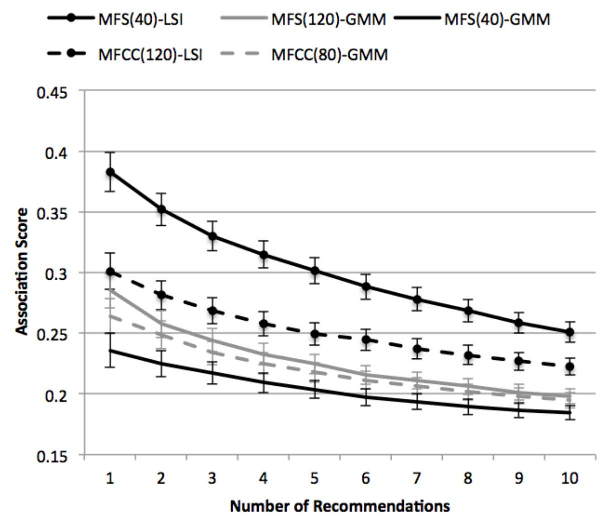

Figure 6: Comparison of best MFS and MFCC by model

Conclusion

The entire mel-frequency spectrum is important when describing the texture of music. When our MFS representation is used, all of the texture information is available to the models we use, leading to improved recommendation quality over MFCC. When MFCC is used, the DCT provides a summarised description of the mel-frequency spectrum, which does not allow a recommender model to learn what is important. Our results show that by not using the DCT, MFS achieves significantly better recommendation quality for each of the three models evaluated.

Traditional music texture representations are extracted with a standard number of filters, and therefore resolution. Our results have shown however that to extract a more meaningful music-inspired texture representation, one must also consider how the textures will be modelled. This link between representation and model is important, and there is no single MFS resolution which is optimal for the three models we have evaluated.

In each of the recommender models presented, the behaviour for MFS is more predictable than MFCC. With GMM more filters are best because the GMM is able to describe the information more meaningfully than DCT. With VOC there is no significant difference, and for LSI fewer filters are best. With MFCC, there are no relationships which emerge between the number of filters used and the recommender model.

The LSI model clearly outperforms both MFS and GMM for texture-based music recommendation. However, both VOC and GMM are commonly found in hybrid recommender systems. Future work therefore will explore how our novel approach to music texture contributes to hybrid recommender systems. It is hoped that by providing a stronger, predictable and robust texture component, increased recommendation quality may be achieved using hybrid representations.

References

Aucouturier, J.; Pachet, F.; and Sandler, M. 2005. The way it sounds: Timbre models for analysis and retrieval of music signals. *IEEE Transactions on Multimedia* 7(6):1028–1035.

Celma, O. 2010. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer-Verlag.

Ellis, D.; Whitman, B.; Berenzweig, A.; and Lawrence, S. 2002. The quest for ground truth in musical artist similarity. In *Proc. International Symposium on Music Information Retrieval ISMIR-2002*, 170–177.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. 2009. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter* 11(1):10–18.

Hershey, J., and Olsen, P. 2007. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume IV, 317–320.

Hicklin, J.; Moler, C.; Webb, P.; Boisvert, R.; Miller, B.; Pozo, R.; and Remington, K. 2000. Jama: A java matrix package. *URL: http://math. nist. gov/javanumerics/jama*.

Horsburgh, B.; Craw, S.; Massie, S.; and Boswell, R. 2011. Finding the hidden gems: Recommending untagged music. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2256–2261.

Lee, C.-H.; Shih, J.-L.; Yu, K.-M.; and Lin, H.-S. 2009. Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features. *Multimedia, IEEE Transactions on* 11(4):670 –682.

Li, T., and Chan, A. 2011. Genre classification and the invariance of mfcc features to key and tempo. *Advances in Multimedia Modeling* 317–327.

Logan, B. 2000. Mel frequency cepstral coefficients for music modelling. In *International Symposium on Music Information Retrieval*.

Mermelstein, P. 1976. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence* 116:91–103.

Pachet, F., and Aucouturier, J. 2004. Improving timbre similarity: How high is the sky? *Journal of negative results in speech and audio sciences* 1(1).

Sigurdsson, S.; Petersen, K. B.; and Lehn-Schiler, T. 2006. Mel frequency cepstral coefficients: An evaluation of robustness of MP3 encoded music. In *Proceedings of the Seventh International Conference on Music Information Retrieval*, 286–289.

Slaney, M. 1998. Auditory toolbox. *Interval Research Corporation, Tech. Rep* 10:1998.

Stevens, S.; Volkmann, J.; and Newman, E. 1937. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*. 8(3):185–190

Yan, Q.; Zhou, Z.; and Li, S. 2012. Chinese accents identification with modified mfcc. In Zhang, T., ed., *Instrumentation, Measurement, Circuits and Systems*, volume 127 of *Advances in Intelligent and Soft Computing*. Springer Berlin / Heidelberg. 659–666.

Yoshii, K.; Goto, M.; Komatani, K.; Ogata, T.; and Okuno, H. 2008. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *Audio, Speech, and Language Processing, IEEE Transactions on* 16(2):435–447.

Zheng, F.; Zhang, G.; and Song, Z. 2001. Comparison of different implementations of MFCC. *Journal of Computer Science and Technology* 16(6):582–589.