Pattern Recognition Letters 32 (2011) 1768–1777

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Music classification via the bag-of-features approach

Zhouyu Fu [*], Guojun Lu, Kai Ming Ting, Dengsheng Zhang

Gippsland School of Information Technology, Monash University, Northways Rd., Churchill, VIC 3842, Australia

ARTICLE INFO

Article history:
Received 6 September 2010
Available online 18 July 2011
Communicated by O. Siohan

Keywords:
Music classification
Bag-of-features
Pattern classification
Multiple codebook model
Support Vector Machines

ABSTRACT

A central problem in music information retrieval is audio-based music classification. Current music classification systems follow a frame-based analysis model. A whole song is split into frames, where a feature vector is extracted from each local frame. Each song can then be represented by a set of feature vectors. How to utilize the feature set for global song-level classification is an important problem in music classification. Previous studies have used summary features and probability models which are either overly restrictive in modeling power or numerically too difficult to solve. In this paper, we investigate the bag-of-features approach for music classification which can effectively aggregate the local features for song-level feature representation. Moreover, we have extended the standard bag-of-features approach by proposing a multiple codebook model to exploit the randomness in the generation of codebooks. Experimental results for genre classification and artist identification on benchmark data sets show that the proposed classification system is highly competitive against the standard methods.

1. Introduction

Music information retrieval is an important emerging research area in multimedia. With the rapid development of information technologies, digital music has become widely available over the Internet and digital storage media. This has created both challenges and opportunities for music management and retrieval. Currently, music management is shifting from manual labeling to automatic music categorization and retrieval based on the standard taxonomies of genre, mood, titles, composers and/or performers, etc.

Music classification is a central problem in music information retrieval. Many applications can be naturally cast into a classification setting, such as genre classification, artist identification and music similarity retrieval. A main challenge in music classification is how to efficiently and effectively utilize low level audio features for high level classification. Current music classification systems follow a frame-based model for audio feature extraction by splitting a song into frames and extracting features from each frame. Consequently, a set of feature vectors are extracted from a single song. Hence an important question arises here on how to use the feature set obtained from frame-level feature extraction for global song-level classification. A standard approach is to use simple summary features that describe the distributions of local feature vectors, such as the mean, variance and covariance of them. This produces a song-level feature vector explicitly for classifier training. Alternatively, one can adopt a similarity based classification scheme for song classification by defining a similarity measure between feature sets without explicitly representing each song with a feature vector. This is usually achieved by modeling the distribution of local feature vectors in the feature set with some probability model, such as Gaussian model and mixture of Gaussians. Different songs can then be compared by their underlying probability models.

Both of the feature representations mentioned above can be regarded as compressing the local information to facilitate song-level classification. However, the way compression is done is susceptible to lose information for classification. By using a single summary feature, one cannot capture all relevant information at different stages of the song. Alternatively, using probability models to represent the feature distribution is either overly restrictive such as the Gaussian assumption or numerically unstable for model comparison such as the mixture of Gaussian.

In this paper, we investigate a *bag-of-features* model to organize the local features extracted at frame level for classification. A global codebook is first constructed by applying vector quantization to all local features in the data set. The codebook vectors correspond to the representative music patterns shared by different songs. Local features in a song are then mapped to the corresponding codebook vectors. A song-level feature representation is then obtained for each song by the histogram vector counting the frequencies of occurrences of the codebook vectors in the song. The bag-of-features approach has been widely used and established itself as the state-of-the-art in the related areas of document analysis and image classification (Nigam et al., 1999; Zhang et al., 2007; Grauman and Darrell, 2007). It is not until recently that the approach has attracted some attention in the music information retrieval community mainly in retrieval (Seyerlehner et al., 2008; Hoffman et al., 2008) and annotation (Hoffman et al., 2009). Here,

[*] Corresponding author.
E-mail address: zhouyu.fu@monash.edu (Z. Fu).

we explore the bag-of-features model for music classification and evaluate it in the classification setting. This is mainly inspired by its success in image classification (Zhang et al., 2007; Grauman and Darrell, 2007). Moreover, we have proposed an extension of the standard bag-of-features approach by exploiting a multiple codebook model to further improve the classification performance.

The remainder of this paper is structured as follows: Section 2 provides a brief review of related work for music classification. Section 3 describes the details of the proposed system, including the bag-of-features representation, a multiple codebook model and implementation details. Experimental results for music genre classification and artist identification on benchmark data sets are reported in Section 4, followed by conclusions in Section 5.

2. Related work

2.1. Audio features for music classification

To distinguish between different categories of music, some discriminating audio features have to be first extracted from the raw audio signal. By the taxonomy of Weihs (Weihs et al., 2007), the audio features used for music classification can be organized into three main categories: short-term, long term, and semantic features. Short-term features are related to the timbre characteristics of the audio signal that describe the properties of instrumentation and sound sources. These features are usually extracted by analyzing the spectra of signals from local windows using tools like short time Fourier transform. The most widely used timbre features include zero crossing rate, spectral centroid, spectral flux (Tzanetakis and Cook, 2002) and Mel-Frequency Cepstrum Coefficients (MFCC) (Logan, 2000; Tzanetakis and Cook, 2002; Berenzweig et al., 2004; Mandel and Ellis, 2005), wavelet transform coefficients (Lin et al., 2005; Li and Ogihara, 2006), MPEG spectral feature descriptors (Kim et al., 2004). Long-term features capture the temporal information of the music and timbre dynamics. The most widely used long-term features include features derived from time series analysis (Xu et al., 2005; Morchen et al., 2006; Meng et al., 2007) and modulation spectral analysis of short-term features extracted from local texture windows (Lidy and Rauber, 2005; Pampalk, 2006; West, 2008). Semantic features pertain to the intrinsic properties that are related to the human perception of music, like melody, rhythm, tempo and pitches. Those features, while being most interpretable by humans, are quite difficult to capture accurately. Many of the semantic features are extracted on top of the short-term features, such as pitch features derived from the pitch histogram (Tzanetakis and Cook, 2002; Tzanetakis et al., 2002).

Notice that both long-term and short-term features are local features in the sense that they are extracted from local windows of the audio signal. We focus on short-term features in this paper, since they are the building blocks for audio features in music classification. Long-term and semantic features are usually extracted by aggregating short-term features. An important problem arising in feature extraction is how to utilize the local information provided by short-term features for global classification. Specifically, we need to derive a feature representation over the set of short-term features that carries the long-term dynamics as well as the semantics of the piece of music being analyzed. This can either be in the form of a global song-level feature vector or a similarity measure between any two songs being compared. In the following discussion, without loss of clarity, we will use the terms short-term features and local features interchangeably.

2.2. Feature representation models

With the short-term features extracted from frames, the next important problem is to derive a global feature representation for song level classification. Depending on the forms of representation, which can either be a global feature vector for each single song or similarity between any two songs, previous feature representation models can be categorized as summary feature based and probability model based representations.

2.2.1. Summary feature based representation

In summary feature based representation, a song is described by a global summary feature vector. This is usually achieved by aggregating the local features extracted from each frame and summarizing them with a single vector. Two levels of aggregation are usually involved here. Due to the non-stationary nature of polyphonic music, the audio features may exhibit very different statistical properties at different stages. Hence it is undesirable to summarize the local features from the whole signal especially for music with long durations. To remedy this problem, larger texture windows (Tzanetakis and Cook, 2002) are usually taken to split the music into several larger segments, each of which still contains a large number of frames and is assumed to have stable sound 'texture' over the segment. Summary features are first derived from each texture window and combined globally to form a single feature vector for each song. Specifically, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ denote the set of local features extracted from local frames. At the first level of aggregation, a single feature F_j is extracted from the jth texture window based on the mapping $f(\mathbf{x}_{(j-1)p+1}, \ldots, \mathbf{x}_{jp})$, where p is the number of frames contained in each texture window. At the second level of aggregation, the average or median of feature vectors from all texture windows is taken to form the song-level feature vector. The feature map f used at the first level of aggregation is essential for feature discriminability and classification performance. Next we provide a brief overview on some previously proposed common representations.

2.2.1.1. Mean–Var summary feature (Tzanetakis and Cook, 2002). The summary feature derived from each texture window is the concatenation of the mean and variance of short-term features extracted from all local frames over the texture window. Specifically, the feature map f for the Mean–Var summary feature can be written as

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_p) = \left[\bar{\mathbf{x}}^T, diag\left(\frac{1}{p}\sum_{i=1}^{p}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\right)\right]^T \quad (1)$$

where $\bar{\mathbf{x}} = \frac{1}{p}\sum_{i=1}^{p}\mathbf{x}_i$ is the mean vector of local features $\mathbf{x}_1, \ldots, \mathbf{x}_p$, and *diag* is an operator that takes the diagonal elements from a matrix to form a column vector. For d-dimensional local features, the above feature map generates a summary feature with $2d$ dimensions for each texture window.

2.2.1.2. Mean–Cov summary feature (Mandel and Ellis, 2005). Alternatively, one can stack the summary feature vector with means and covariance values computed from frame-level features within the texture window. This leads to the following representation of Mean–Cov summary feature

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_p) = \left[\bar{\mathbf{x}}^T, svec\left(\frac{1}{p}\sum_{i=1}^{p}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\right)\right]^T \quad (2)$$

where *svec* is the vectorization operator for symmetric matrix that stacks the lower-triangular part of a symmetric matrix into a single vector by columns. For d-dimensional features $\mathbf{x}_i$, the Mean–Cov representation scheme produces a higher $\frac{d(d+1)}{2}$-dimensional summary feature vector.

2.2.1.3. Modulation spectra analysis (Pampalk, 2006; West, 2008). The local features themselves can be regarded as vector valued time series data. Hence one can further apply spectral analysis to the local features and represent the global song-level feature using the features derived from the spectral analysis. Specifically, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$ be the data matrix with local features stacked in columns, and $\mathbf{x}^{jT}$ be the jth row of $\mathbf{X}$ corresponding to the time series given by the jth attribute of the local features for $j = 1, \ldots, D$, the modulation spectrum is then obtained by applying Fourier transform to signal $\mathbf{x}^{jT}$ and the coefficients are then taken to form the song-level feature. This is the general paradigm followed by modulation spectra analysis, although the implementation details may differ. Examples of modulation features include Fluctuation Patterns (FP) (Pampalk, 2006) and Rhythmic Cepstrum Coefficients (RCC) (West, 2008).

2.2.2. *Probability model based representation*

Many pattern classifiers can use pairwise distance or similarity values directly for classification, like the K-Nearest Neighbour (KNN) classifier. Hence, instead of explicitly representing each song with a single feature vector, one can alternatively define similarity measures between any two feature sets for each pair of songs. This can be achieved by modeling the distribution of local feature vectors in each feature set with a probability model, which can be either parametric or non-parametric. Let $\mathcal{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$ and $\mathcal{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$ denote the local features extracted from two songs, generated from two probability distributions $P(X)$ and $Q(Y)$ respectively. We can then compare the two songs by comparing their underlying distributions $P(X)$ and $Q(Y)$. This is usually estimated by using certain divergence measures proposed for the comparison of probability distributions, such as the symmetrized Kullback–Leibler (KL) divergence defined below:

$$\mathrm{KL}(p(x), q(x)) = \int (p(x) - q(x)) \log \frac{p(x)}{q(x)} dx \tag{3}$$

The choice of probability models is essential here, as different models give rise to different song-level similarities as well as different degrees of computational complexity. The following models have been proposed in the past for similarity computation.

2.2.2.1. Single Gaussian model (Pampalk, 2006; Mandel and Ellis, 2005). The single Gaussian model assumes that the local features extracted from each song are governed by the normal distribution with the following probability density function (*pdf*)

$$p(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right) \tag{4}$$

The main advantage for the single Gaussian representation is its low computational complexity. The symmetrized KL divergence between two Gaussian models $p(X)\widetilde{N}(\mu_\mathcal{X}, \Sigma_\mathcal{Y})$ and $q(Y)\widetilde{N}(\mu_\mathcal{Y}, \Sigma_\mathcal{Y})$ can be rewritten in the closed form as follows:

$$\begin{aligned}\mathrm{KL}(p(X), q(Y)) =& tr\left(\Sigma_\mathcal{X}^{-1}\Sigma_\mathcal{Y} + \Sigma_\mathcal{Y}^{-1}\Sigma_\mathcal{X}\right) \\ &+ (\mu_\mathcal{X} - \mu_\mathcal{Y})^T \left(\Sigma_\mathcal{X}^{-1} + \Sigma_\mathcal{Y}^{-1}\right)(\mu_\mathcal{X} - \mu_\mathcal{Y})\end{aligned} \tag{5}$$

2.2.2.2. Gaussian mixture models (Berenzweig et al., 2004; Pampalk, 2006). The local features can also be modeled using mixture of Gaussian models (GMM), where each feature is generated from one of the Gaussian component in the mixture model. The *pdf* of the GMM is given by

$$p(\mathbf{x}) = \sum_j \alpha_j f(\mathbf{x}|\mu_j, \Sigma_j) \quad \text{s.t.} \quad \sum_j \alpha_j = 1 \tag{6}$$

The single Gaussian model is a special case of GMM with only a single Gaussian component. Hence GMM is more expressive than the single Gaussian model. However, this is at the cost of computational complexity. In fact, there is no straightforward way to compare two Gaussian mixture models as there is no closed form derivation of KL divergence for GMMs. A Monte-Carlo sampling approach is usually taken for the computation of KL divergence, which is much more computationally expensive.

2.2.2.3. Clustering based models (Logan and Salomon, 2001; Pampalk et al., 2003; Berenzweig et al., 2004). Alternatively, the distributions of local features can also be modeled in a non-parametric fashion. In (Logan and Salomon, 2001; Pampalk et al., 2003; Berenzweig et al., 2004), K-means clustering is applied to local features extracted from each song to generate a compact summary by a few cluster centers. Song-level similarities can then be derived by comparing the similarity between two sets of cluster centers using the earth mover distance (EMD) (Rubner et al., 1998). Again, EMD is quite computationally expensive compared to single Gaussian models as each computation of one EMD involves solving a transportation problem.

Both summary feature and probability model based representation schemes have their respective advantages and drawbacks. Summary features are simple and efficient to compute, but they do not fully exploit the underlying distributions of short-term local features. Probability models, on the other hand, are conceptually more appropriate with more discriminative information in the representation. Nevertheless, due to numerical difficulties arising in distance computation, it is usually quite hard to evaluate the similarity based on probability modeling except for very simple models like the single Gaussian, which is incapable of capturing the differences between complex distributions of non-Gaussian nature.

3. Proposed system for music classification

In this section, we introduce a bag-of-features (BOF) model based system for music classification that overcomes the respective drawbacks of feature representations discussed in the previous section. The BOF model does not make any assumption on the underlying probability distribution and hence can better preserve the discriminative information in feature representation. We then discuss the important issue of similarity comparison based on the BOF representation. Finally, we have also investigated the possibility of combining multiple codebooks in the BOF model for the reduction of variance in codebook construction.

3.1. *Bag-of-features representation*

The bag-of-features (BOF) model is originated from the bag-of-words (BOW) model, which was first proposed for document classification. The BOW model basically provides a document-level feature representation scheme by viewing each document as an order-less collection of words. The assumption is that words in the document are a strong indicator of its categorical nature. For instance, words like "democratic" and "republican" are most likely to appear in political articles, whereas "drop-kick" and "touch-down" are mostly found in sports news. Hence a BOW representation can be adopted where each document is represented by a histogram vector counting the frequency of the occurrence for each word in the vocabulary. Despite its simplicity, the BOW model is effective and has been widely used in document classification. A similar idea has been adopted for image classification (Zhang et al., 2007; Grauman and Darrell, 2007) and recently introduced to music similarity retrieval (Seyerlehner et al., 2008; Hoffman et al., 2008) by representing the target object with a collection of local features. The major difference here is that the space of local features for image and music is dense and contains infinite number of points in it, unlike the case of document analysis, where the words are already defined from a vocabulary with fixed size. Hence, for the application of BOF model to image and music analysis, a vocabulary has to be learned. This is normally achieved through vector quantization by compressing the local features into a fixed dictionary.

Here we focus on the Mel-Frequency Cepstrum Coefficients (MFCC) extracted from each frame as local features used for BOF feature representation. However, the BOF representation is quite general in nature and can accommodate for other choices of local features. MFCC features are used because they have demonstrated superior performance in speech recognition (Rabiner and Juang, 1993) and have been recently successfully applied to music classification tasks (Logan, 2000; Tzanetakis and Cook, 2002; Mandel and Ellis, 2005).

Fig. 1 illustrates the process for obtaining the BOF representation in the context of music classification. It is achieved in the following two steps.

1. *Codebook Construction*
   First, local features are extracted from each song in the training data set, where each song is represented by a varying number of feature vectors. All local features in the training data set are then collected on top of which a codebook is built via vector quantization. This is achieved by running K-means clustering on them. In practice, due to the huge number of local features in the training data set, we subsampled the feature set and randomly selected a subset of feature vectors taken from different local frames for clustering. The number of cluster centers is fixed, which leads to a codebook with a fixed size. The resulting codebook, being a compact representation of all local features, is the dictionary of "audio-words". Each cluster center is a codebook vector that represents an "audio-word". Notice that the codebook is only constructed once in the training stage.
2. *Assignment*
   With the learned codebook at hand, we can obtain the BOF representation for each song by assigning each local feature with a codebook vector and calculating the normalized histogram for the frequency of occurrences of each codebook vector in the codebook. Specifically, let $\mathcal{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_m]$ be the codebook vectors, For a song with the set of local features $\mathcal{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, the BOF representation is a histogram vector represented by $[h(1), \ldots, h(m)]$ with $h(k)$ given by

$$h(k) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{d(\mathbf{x}_i,\mathbf{c}_k) \leqslant \max_{j \neq k} d(\mathbf{x}_i,\mathbf{c}_j)} \tag{7}$$

where $\mathbf{1}_A$ is the indicator function which is 1 if condition A is true and 0 otherwise, $d(\mathbf{x}_i, \mathbf{c}_k)$ denotes the distance between the ith local feature $\mathbf{x}_i$ in the song and the kth codebook vector $\mathbf{c}_k$. The kth bin is the number of local features matched by the kth codebook vector over the total number of local features. A local feature is matched by the kth codebook vector if and only if it has the closest distance to the codebook vector.

The BOF representation, despite being in the form of a global song-level histogram vector, has a probabilistic interpretation. Each entry in the histogram captures the frequency of occurrence for the corresponding codebook vector. Since the histogram is normalized with non-negative entries and unit sum, it is equivalent to a probability mass function (**pmf**) that specifies a multinomial distribution over a discrete random variable with K states, K being the number of clusters. The probability of choosing certain state is proportional to the frequency for observing the codebook vector. Moreover, unlike previous clustering based models (Logan and Salomon, 2001; Pampalk et al., 2003; Berenzweig et al., 2004), the BOF representation creates a global cluster model that summarizes the distributions of all local features in the data set, whereas previous approaches builds a separate cluster model for each song. This not only makes the comparison of cluster models for different songs difficult, but is computationally quite expensive as well, since clustering needs to be performed for each song to generate a summary given by the cluster centers. The BOF model, on the other hand, does not have such problems. The codebook is only trained once and there is no need for retraining the codebook at testing time.

The choice of K, the number of clusters, offers more flexibility in feature modeling. A compact codebook with small value of K has

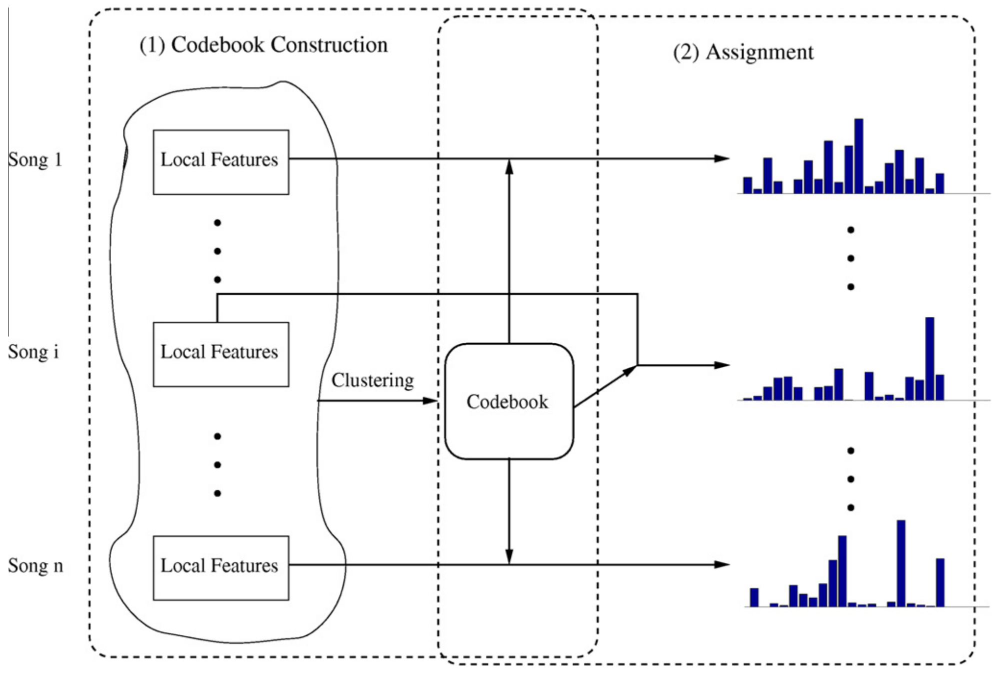


Fig. 1. Illustration of BOF computation.

higher degree of compression at the cost of lower discriminability and underfitting due to quantization effects. On the other hand, a fine codebook with large value of K is able to distinguish relatively small differences in feature values, but at the same time may introduce noises by mapping similar feature vectors into different clusters. This would inevitably leads to overfitting. An optimal sized codebook should well balance the trade-off between underfitting and overfitting.

3.2. *Comparing BOF representations*

The histogram based BOF representation has made the similarity comparison between pairs of songs easy. We can compute the similarity between two songs by comparing the distances between their underlying histograms. A natural distance measure to consider is the squared Euclidean distance. However, due to the special property of histograms, a number of alternative distance measures can be used instead such as the χ^2 distance defined below

$$d_{\chi^2}(h_i,h_j)=\sum_k\frac{(h_i(k)-h_j(k))^2}{h_i(k)+h_j(k)+\epsilon} \tag{8}$$

where ϵ is a small value added to avoid division by zero. The χ^2 measure is a more appropriate measure for histogram comparison, as compared with the conventional Euclidean distance which ignores the normalization term in the summation of the RHS of the above equation. The intuition behind this normalization is to amplify the differences for rare codewords over common codewords, since features corresponding to rare codewords usually carry more discriminative information, whereas common codewords are most likely to be associated with features that are shared across different categories. An analogy can be made with document analysis, where conjunctions like ‘a’, ‘an’, ‘the’, ‘and’, ‘but’ occur frequently in all documents without any discriminative information. Using a χ^2 distance can effectively suppress their influences. Alternatively, we can consider the inverse document frequency (IDF) of each histogram bin to counter with such artifacts. This leads to the following TF (term frequency)-IDF distance,

$$d_{\text{TF-IDF}}(h_i,h_j)=\sum_k((h_i(k)-h_j(k))\text{IDF}(k))^2 \tag{9}$$

$$\text{IDF}(k)=\log\frac{|\{\mathbf{X}\}|}{|\{\mathbf{X}:\mathbf{c}_k\in\mathbf{X}\}|}$$

where IDF(k) is a scale factor for the kth bin given by the logarithm of the ratio between the total number of songs and the number of songs containing local features represented by the kth codebook vector. We can see that the rare the codebook vectors, the larger the IDF value, and the greater degree of emphasis for the corresponding bin.

Alternatively, as mentioned earlier, the frequency histogram is basically a *pmf*, we can simply use a proper distance measure for comparing probability densities or mass such as the symmetrized KL divergence below:

$$d_{\text{KL}}(h_i,h_j)=\sum_i\left(h'_i(k)-h'_j(k)\right)\log\frac{h'_i(k)}{h'_j(k)} \tag{10}$$

$$h'_i(k)=\frac{h_i(k)+\epsilon}{1+n\epsilon}$$

here h'_i is the smoothed version of histogram h_i used simply for concerns of numerical stability.

One final similarity measure specific to histograms that has been widely used in image matching is the histogram intersection kernel (Grauman and Darrell, 2007) which counts the total overlap between two histograms

$$K_{HistInt}(h_i,h_j)=\sum_i\max(h_i(k),h_j(k)) \tag{11}$$

$$d_{HistInt}(h_i,h_j)=1-K_{HistInt}(h_i,h_j) \tag{12}$$

The larger the overlap, the more similar the two histograms are as given by larger value of $K_{HistInt}$, and the closer the distance between them as given by smaller value of $d_{HistInt}$.

With the distances defined between two histograms, we can then train classifiers over them. In this paper, we focus on two common classifiers, the K-Nearest Neighbor (K-NN) classifier as well as the Support Vector Machine (SVM) classifier. K-NN uses distances directly for classification. SVM uses kernels for classification, which is basically a nonlinear mapping from distances to the kernel values. We have employed the Radial Basis Function (RBF) kernel for all distance measures mentioned above except histogram intersection where the kernel is already defined in Eq. (11). Given the distance $d(h_i,h_j)$ between histograms h_i and h_j, the RBF kernel is an exponential map given by,

$$K(h_i,h_j)=\exp\left(-\gamma d(h_i,h_j)\right) \tag{13}$$

3.3. *A multiple codebook model*

In the standard BOF representation, only a single codebook has been created and used. The performance of such models is much influenced by the quality of the codebook used. Due to random subsampling of data and initial selection of cluster centers, a different codebook is constructed each time by repeating the construction process multiple times even for the same training data set. To handle this problem, we have adopted a multiple codebook model by generating multiple codebooks with different random selection of subsamples for clustering as well as initial cluster centers. Hence a different codebook is created with each random initialization. Let $\mathcal{C}^j=\left[\mathbf{c}^j_1,\ldots,\mathbf{c}^j_m\right]$ be the jth codebook with $j=1,\ldots,|\mathcal{C}|$, the distance between two songs represented by two feature sets $\mathcal{X}$ and $\mathcal{Y}$ based on the multiple codebook model is simply given by the average distance over all codebooks

$$d(\mathcal{X},\mathcal{Y})=\frac{1}{|\mathcal{C}|}\sum_{j=1}^{|\mathcal{C}|}d(h(\mathcal{X};\mathcal{C}^j),h(\mathcal{Y};\mathcal{C}^j)) \tag{14}$$

where $h(\mathcal{X};\mathcal{C}^j)$ denotes the BOF representation for $\mathcal{X}$ using the jth codebook which can be computed via Eq. (7).

The motivations for exploiting multiple codebooks are twofold. Firstly, K-means clustering focuses more on regions with high density and may easily lose track over sparse regions, which may carry some discriminative information. This is especially true for data with no clear cluster structures, which is usually the case for local feature distributions. Hence a single codebook may not be able to cover all regions in the feature space. By using multiple codebooks, we can somehow counteract this negative effect. Secondly, each codebook is differed randomly and explains different facets of the underlying data distributions, by aggregating them we can reduce the variance in the BOF representation and achieve a more stable classification performance. This is similar in spirit to the bagging based ensemble models used in classification (Breiman, 1996). Note the above two problems cannot be simply resolved by either increasing the subsample size for clustering or the number of clusters in the single codebook model. We have empirically observed that with a sufficiently large subsample, further increasing the sample size would not much effect the clustering quality. Increasing number of clusters, on the other hand, is likely to introduce bias in the representation and leads to overfitting.

An example of multiple codebooks is shown in Fig. 2, where the local features extracted from a subset of the GTZAN data set are plotted in small light-colored dots, and the codebook vectors from different codebooks are represented by larger marks in different color and shape combinations. For visualization purposes, we have first applied Principal Component Analysis (PCA) to the local feature vectors and keep the top two principle components with the largest variances. The same PCA projection is applied to the codebook vectors accordingly. The results of codebooks with size of 50 are shown here for clarity. From the figure, we can see that the codebook vectors obtained from different clustering processes are somehow different. As we can see, the feature distribution is highly inhomogeneous and not a single codebook can capture all information represented in the data distribution. However, the coverage of codebook vectors from all codebooks generated are better than the coverage of any single codebook.

3.4. Implementation details

The input audio signals are split into 23 ms frames with 50% overlapping between adjacent frames. We followed the standard routines for MFCC feature extraction in each frame. Thirty-six Mel filters have been used covering the Mel scale of the spectral range between 0 Hz and the Nyquist frequency with 50% overlapping between adjacent filters. The output power spectrum over the Mel filters are mapped from raw magnitudes to decibel values. Finally, DCT is applied to the filter output and only the top 20 DCT coefficients were kept (excluding the DC component) to form the 20-dimensional MFCC feature for each local frame.

Three songs were selected from each music class and all local features extracted from the selected songs were used as the input data for codebook construction. Euclidean distance is used for computing the distance between local MFCC features in both K-means clustering and BOF feature computation in Eq. (7). To accommodate for different dynamic ranges of the input attributes, we have rescaled all attributes to the range of $[0,1]$ before codebook construction. The local features for each song are then rescaled accordingly before the computation of BOF vector. We have set ϵ to $1e{-}6$ in both Eqs. (8) and (10) to avoid numerical problems for the distance computation between histogram vectors.

We have adopted the χ^2 distance defined in Eq. (8) as the distance measure for histogram comparison. The codebook size is fixed to 3000 and the number of codebooks used in the multiple codebook model is empirically set to 20. However, as noted in the experimental section, the performance of our model is not sensitive to the selection of these parameters as long as the appropriate distance measure is chosen.

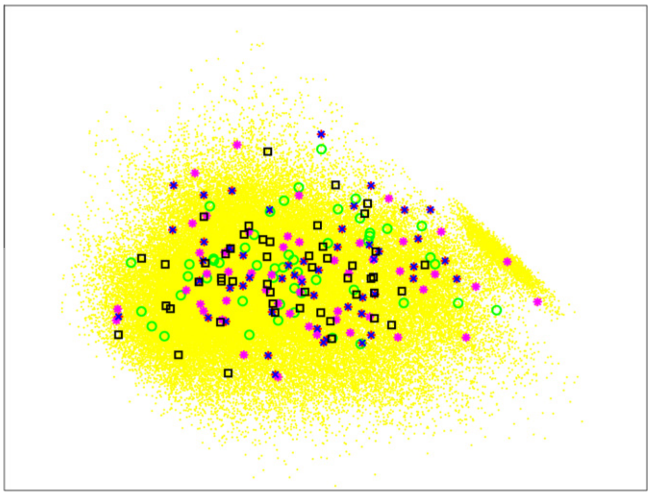


Fig. 2. Illustration of codebook uncertainty and multiple codebook model.

4. Experimental results

In this section, we present the experiments performed to evaluate the proposed BOF approach on two music classification tasks, namely genre classification and artist identification. Firstly, the experimental setup is described. Next, we report the classification results comparing BOF to the alternative feature representation models discussed in Section 2.2. Finally, we discuss issues related to hyperparameter selection, including the choice of distance measure, codebook size, and number of codebooks used in the multiple codebook framework.

4.1. Experimental setup

Two data sets are used in our experiments, the GTZAN data set for genre classification (Tzanetakis and Cook, 2002) and the USPOP2002 data set for artist identification (Mandel and Ellis, 2005). Both data sets are publicly available and serve as benchmarks for different algorithms to be compared. The GTZAN data set contains 1000 songs uniformly distributed over the following 10 genres – blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. The USPOP2002 data set contains songs from over 1000 albums of 400 pop/rock artists from which a subset of artists is used for the experiment. We have covered the same 18 artists reported in the experiments of Mandel and Ellis (Mandel and Ellis, 2005) to ensure that a sufficient number of songs is available for each artist. Table 1 presents the details of the above two data sets used in our experiment. It is worth notice that for USPOP2002 data set, only the MFCC features extracted from 23 ms local frames are available. The GTZAN data set contains the raw audio data for all songs in WAV format and the MFCC features were extracted based on the details discussed in Section 3.4.

For the GTZAN data set, we adopted stratified 10-fold cross validation as our evaluation scheme by splitting the data set into 10 equal folds preserving the proportion of data from different classes for each fold. In each training–testing round, we used 9 folds of data for training and the remaining fold for testing. For the USPOP2002 data set, we adopted a similar evaluation setting as in (Mandel and Ellis, 2005) by applying an album filter to the partition of training and testing sets. In each round, we randomly selected one album from each artist and used all tracks from the selected album for testing. Tracks from the other albums are used as training data. This partition scheme ensures that training and testing data sets do not cross over songs in the same album and thus minimizes the album effect that songs from the same album tend to sound more similar than those from different albums even for the same artist. The album effect would positively bias the classification performance as noted in (Mandel and Ellis, 2005) and need be removed by careful data partition for training and testing.

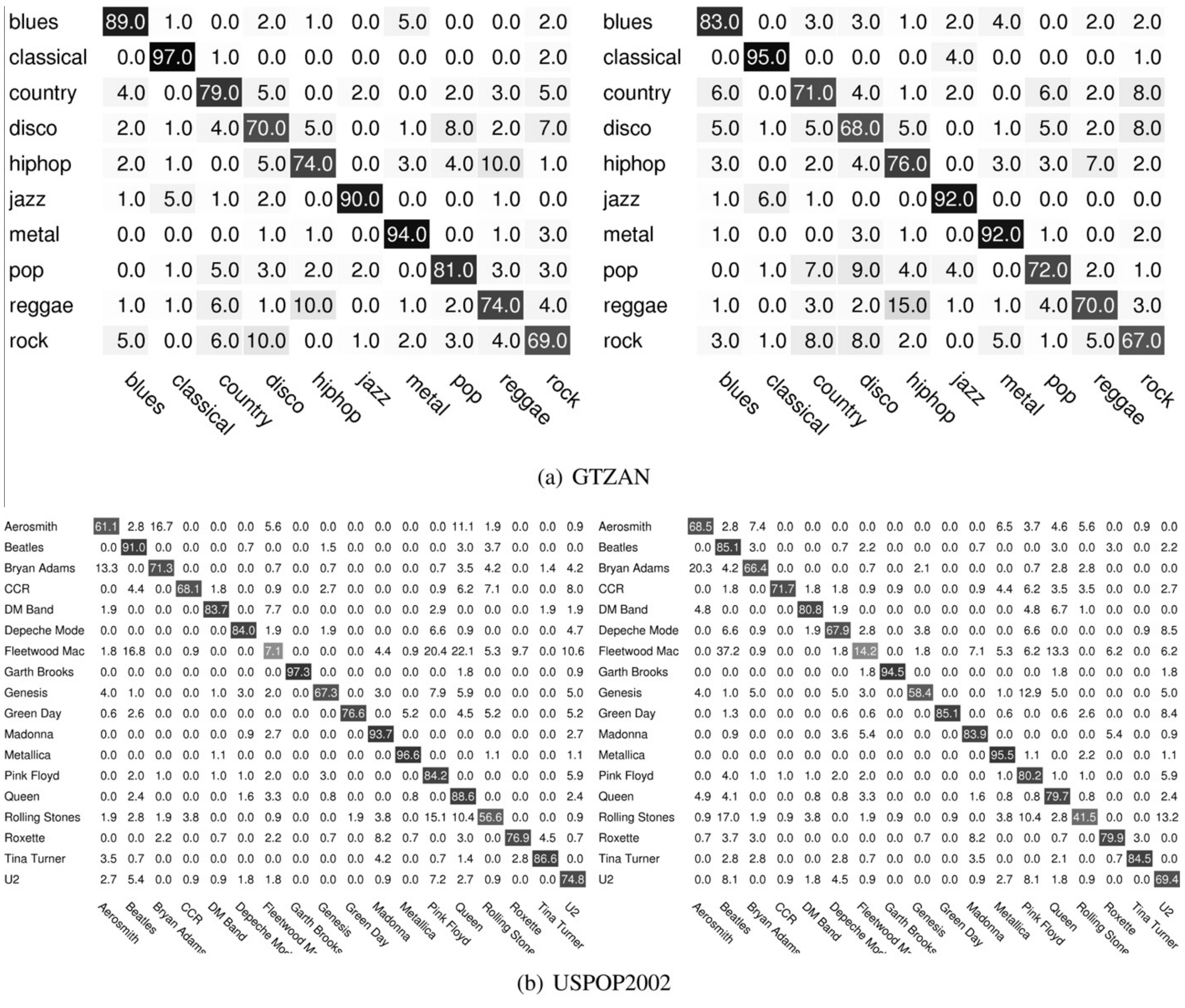
For each data set, we have repeated the classification experiment 10 times with different random partitions of training and testing data sets. All models being compared were applied to the same data partitions to ensure a fair comparison. The optimal classifier parameters were selected via 5-fold cross validation for the GTZAN data set and leave-one-out validation at album level for the USPOP2002 data set, same as the partitioning of training/testing data. For the two classifiers tested in our experiment, namely the K-NN classifier and the RBF kernel based SVM classifier, the classifier parameters are K, the number of nearest neighbors for K-NN, and C and γ, the regularization and the scale parameters for the RBF-SVM. Since SVM is inherently a binary classifier and both data sets contain multiclass data, we have used the pairwise fusion framework for multiclass classification using SVM. Binary SVM classifiers are trained using the LibSVM package available online (Chang and Lin, 2001). For feature based representation

Table 1
Statistics of databases used in experiment.

	Songs	Classes	Track length	Audio info	Data format
GTZAN	1000	10	30.41 s	22050 Hz, mono	Raw audio data
USPOP2002	1322	18	94.57 s	22050 Hz, mono	MFCC features

Table 2
Performance comparison of the proposed BOF representation vs. alternative models for music classification.

	Mean–Var	Mean–Cov	FP	Gaussian	BOF
(a) GTZAN					
K-NN	67.20 ± 2.60%	69.90 ± 3.51%	69.90 ± 3.91%	70.60 ± 3.10%	**73.10 ± 3.18%**
SVM	75.10 ± 4.32%	78.60 ± 2.42%	77.70 ± 2.83%	70.40 ± 3.17%	**81.70 ± 1.85%**
(b) USPOP2002					
K-NN	45.76 ± 4.32%	52.04 ± 5.09%	43.49 ± 4.00%	52.70 ± 5.19%	**54.60 ± 4.51%**
SVM	58.58 ± 3.22%	73.10 ± 2.81%	64.81 ± 3.78%	67.89 ± 6.10%	**76.03 ± 3.24%**



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	89.0	1.0	0.0	2.0	1.0	0.0	5.0	0.0	0.0	2.0
classical	0.0	97.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
country	4.0	0.0	79.0	5.0	0.0	2.0	0.0	2.0	3.0	5.0
disco	2.0	1.0	4.0	70.0	5.0	0.0	1.0	8.0	2.0	7.0
hiphop	2.0	1.0	0.0	5.0	74.0	0.0	3.0	4.0	10.0	1.0
jazz	1.0	5.0	1.0	2.0	0.0	90.0	0.0	0.0	1.0	0.0
metal	0.0	0.0	0.0	1.0	1.0	0.0	94.0	0.0	1.0	3.0
pop	0.0	1.0	5.0	3.0	2.0	2.0	0.0	81.0	3.0	3.0
reggae	1.0	1.0	6.0	1.0	10.0	0.0	1.0	2.0	74.0	4.0
rock	5.0	0.0	6.0	10.0	0.0	1.0	2.0	3.0	4.0	69.0

	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	83.0	0.0	3.0	3.0	1.0	2.0	4.0	0.0	2.0	2.0
classical	0.0	95.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	1.0
country	6.0	0.0	71.0	4.0	1.0	2.0	0.0	6.0	2.0	8.0
disco	5.0	1.0	5.0	68.0	5.0	0.0	1.0	5.0	2.0	8.0
hiphop	3.0	0.0	2.0	4.0	76.0	0.0	3.0	3.0	7.0	2.0
jazz	1.0	6.0	1.0	0.0	0.0	92.0	0.0	0.0	0.0	0.0
metal	1.0	0.0	0.0	3.0	1.0	0.0	92.0	1.0	0.0	2.0
pop	0.0	1.0	7.0	9.0	4.0	4.0	0.0	72.0	2.0	1.0
reggae	1.0	0.0	3.0	2.0	15.0	1.0	1.0	4.0	70.0	3.0
rock	3.0	1.0	8.0	8.0	2.0	0.0	5.0	1.0	5.0	67.0

(a) GTZAN

	Aerosmith	Beatles	Bryan Adams	CCR	DM Band	Depeche Mo	Fleetwood M	Garth Brooks	Genesis	Green Day	Madonna	Metallica	Pink Floyd	Queen	Rolling Stone	Roxette	Tina Turner	U2
Aerosmith	61.1	2.8	16.7	0.0	0.0	0.0	5.6	0.0	0.0	0.0	0.0	0.0	0.0	11.1	1.9	0.0	0.0	0.9
Beatles	0.0	91.0	0.0	0.0	0.0	0.7	0.0	0.0	1.5	0.0	0.0	0.0	0.0	3.0	3.7	0.0	0.0	0.0
Bryan Adams	13.3	0.0	71.3	0.0	0.0	0.0	0.7	0.0	0.7	0.0	0.0	0.0	0.7	3.5	4.2	0.0	1.4	4.2
CCR	0.0	4.4	0.0	68.1	1.8	0.0	0.9	0.0	2.7	0.0	0.0	0.0	0.9	6.2	7.1	0.0	0.0	8.0
DM Band	1.9	0.0	0.0	0.0	83.7	0.0	7.7	0.0	0.0	0.0	0.0	0.0	2.9	0.0	0.0	0.0	1.9	1.9
Depeche Mode	0.0	0.0	0.0	0.0	0.0	84.0	1.9	0.0	1.9	0.0	0.0	0.0	6.6	0.9	0.0	0.0	0.0	4.7
Fleetwood Mac	1.8	16.8	0.0	0.9	0.0	0.0	7.1	0.0	0.0	0.0	4.4	0.9	20.4	22.1	5.3	9.7	0.0	10.6
Garth Brooks	0.0	0.0	0.0	0.0	0.0	0.0	0.0	97.3	0.0	0.0	0.0	0.0	0.0	1.8	0.0	0.0	0.0	0.9
Genesis	4.0	1.0	0.0	0.0	1.0	3.0	2.0	0.0	67.3	0.0	3.0	0.0	7.9	5.9	0.0	0.0	0.0	5.0
Green Day	0.6	2.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	76.6	0.0	5.2	0.0	4.5	5.2	0.0	0.0	5.2
Madonna	0.0	0.0	0.0	0.0	0.0	0.9	2.7	0.0	0.0	0.0	93.7	0.0	0.0	0.0	0.0	0.0	0.0	2.7
Metallica	0.0	0.0	0.0	0.0	1.1	0.0	0.0	0.0	0.0	0.0	0.0	96.6	0.0	0.0	1.1	0.0	0.0	1.1
Pink Floyd	0.0	2.0	1.0	0.0	1.0	1.0	2.0	0.0	3.0	0.0	0.0	0.0	84.2	0.0	0.0	0.0	0.0	5.9
Queen	0.0	2.4	0.0	0.0	0.0	1.6	3.3	0.0	0.8	0.0	0.0	0.8	0.0	88.6	0.0	0.0	0.0	2.4
Rolling Stones	1.9	2.8	1.9	3.8	0.0	0.0	0.9	0.0	0.0	1.9	3.8	0.0	15.1	10.4	56.6	0.0	0.0	0.9
Roxette	0.0	0.0	2.2	0.0	0.7	0.0	2.2	0.0	0.7	0.0	8.2	0.7	0.0	3.0	0.0	76.9	4.5	0.7
Tina Turner	3.5	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.2	0.0	0.7	1.4	0.0	2.8	86.6	0.0
U2	2.7	5.4	0.0	0.9	0.9	1.8	1.8	0.0	0.0	0.0	0.9	0.0	7.2	2.7	0.9	0.0	0.0	74.8

	Aerosmith	Beatles	Bryan Adams	CCR	DM Band	Depeche Mo	Fleetwood M	Garth Brooks	Genesis	Green Day	Madonna	Metallica	Pink Floyd	Queen	Rolling Stone	Roxette	Tina Turner	U2
Aerosmith	68.5	2.8	7.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.5	3.7	4.6	5.6	0.0	0.9	0.0
Beatles	0.0	85.1	3.0	0.0	0.0	0.7	2.2	0.0	0.0	0.0	0.7	0.0	0.0	3.0	0.0	3.0	0.0	2.2
Bryan Adams	20.3	4.2	66.4	0.0	0.0	0.0	0.7	0.0	2.1	0.0	0.0	0.0	0.7	2.8	2.8	0.0	0.0	0.0
CCR	0.0	1.8	0.0	71.7	1.8	1.8	0.9	0.9	0.0	0.0	0.9	4.4	6.2	3.5	3.5	0.0	0.0	2.7
DM Band	4.8	0.0	0.0	0.0	80.8	1.9	0.0	0.0	0.0	0.0	0.0	0.0	4.8	6.7	1.0	0.0	0.0	0.0
Depeche Mode	0.0	6.6	0.9	0.0	1.9	67.9	2.8	0.0	3.8	0.0	0.0	0.0	6.6	0.0	0.0	0.0	0.9	8.5
Fleetwood Mac	0.0	37.2	0.9	0.0	0.0	1.8	14.2	0.0	1.8	0.0	7.1	5.3	6.2	13.3	0.0	6.2	0.0	6.2
Garth Brooks	0.0	0.0	0.0	0.0	0.0	0.0	1.8	94.5	0.0	0.0	0.0	0.0	0.0	1.8	0.0	0.0	0.0	1.8
Genesis	4.0	1.0	5.0	0.0	0.0	5.0	3.0	0.0	58.4	0.0	0.0	1.0	12.9	5.0	0.0	0.0	0.0	5.0
Green Day	0.0	1.3	0.0	0.0	0.0	0.6	0.6	0.0	0.0	85.1	0.0	0.6	0.0	0.6	2.6	0.0	0.0	8.4
Madonna	0.0	0.9	0.0	0.0	0.0	3.6	5.4	0.0	0.0	0.0	83.9	0.0	0.0	0.0	0.0	5.4	0.0	0.9
Metallica	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	95.5	1.1	0.0	2.2	0.0	0.0	1.1
Pink Floyd	0.0	4.0	1.0	1.0	1.0	2.0	2.0	0.0	0.0	0.0	0.0	1.0	80.2	1.0	1.0	0.0	0.0	5.9
Queen	4.9	4.1	0.0	0.0	0.8	0.8	3.3	0.0	0.0	0.0	1.6	0.8	0.8	79.7	0.8	0.0	0.0	2.4
Rolling Stones	0.9	17.0	1.9	0.9	3.8	0.0	1.9	0.9	0.0	0.9	0.0	3.8	10.4	2.8	41.5	0.0	0.0	13.2
Roxette	0.7	3.7	3.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	8.2	0.0	0.0	0.7	0.0	79.9	3.0	0.0
Tina Turner	0.0	2.8	2.8	0.0	0.0	2.8	0.7	0.0	0.0	0.0	3.5	0.0	0.0	2.1	0.0	0.7	84.5	0.0
U2	0.0	8.1	0.0	0.9	1.8	4.5	0.9	0.0	0.0	0.0	0.9	2.7	8.1	1.8	0.9	0.0	0.0	69.4

(b) USPOP2002

Fig. 3. Confusion matrices of the proposed BOF representation vs. the Mean–Cov representation for music classification using the SVM classifier. Left column: confusion matrices of BOF; right column: confusion matrices for Mean–Cov.

models like Mean–Var, Mean–Cov, and FP, the squared Euclidean distance is adopted for comparing the distance between two feature vectors. To accommodate for the different dynamic ranges of feature variables, each feature variable was rescaled to zero mean and unit standard deviation for the training data set and the same parameters were kept to rescale testing data before taking distance computation. This is equivalent to computing the Mahalanobis distance with diagonal covariance matrix between input features (Mandel and Ellis, 2005).

4.2. Result analysis

We now describe the experimental results achieved by the proposed BOF model using the hyperparameters identified in the last section and compare it with alternative models for two music classification tasks. The alternative models selected for comparison include three summary feature based models, Mean–Var, Mean–Cov, and FP, and one probability model, the single Gaussian model. The average accuracy together with the standard deviation from 10-fold cross validation for the K-NN and SVM classifiers are reported in Table 2. It can be seen that the proposed BOF feature representation model has the best performance with both K-NN and SVM classifiers. It outperforms the conventional Mean–Var feature aggregation method by a large margin and achieves a gap of improvement over Mean–Cov and FP, the state-of-the-art representations, in terms of classification accuracy. The differences in accuracy rates between the BOF model and these two methods are statistically significant for both data sets according to the outcomes of paired *t*-tests at significance level 5%. The single Gaussian aggregation scheme, on the other hand, although being competitive with the proposed BOF scheme when combined with the K-NN classifier, does not perform as well as for SVM classifier.

To examine the classification accuracies more closely, we have also plotted the average confusion matrices generated by the two best-performing models with the SVM classifier, the proposed BOF model and the alternative Mean–Cov representation, for both GTZAN and USPOP2002 data sets in Fig. 3. The left-hand column shows the confusion matrices for the BOF representation, whereas the right-hand column displays the confusion matrices for the Mean–Cov representation. From the figure, it can be easily seen that our BOF model not only achieves a higher overall classification accuracy, but also performs better for the majority of classes for each classification task, as indicated by larger values in the diagonal of the confusion matrices as compared to those from the Mean–Var model. Specifically, our representation model can better classify 8 of 10 genres than the alternative Mean–Cov model for genre classification, and 13 out of 18 artists for artist identification.

The proposed BOF model also compares favorably over previous methods for genre classification on the GTZAN data set with the same experimental setup. Table 3 reports the performance comparison results with our result highlighted achieving the highest classification accuracy rate. More importantly, the other methods reported here have used multiple features for classification, whereas our method has only focused on the MFCC features with a different feature representation. This further validates the effectiveness of the proposed BOF feature representation model. For artist identification, the previously reported best performances were obtained by using the Mean–Cov feature representation with MFCC features (Mandel and Ellis, 2005), which, as we have shown in Table 2, performs inferior to our proposed method.

Compared to complex probability based models like the GMM and the clustering model mentioned in Section 2.2, our method has a lower computational complexity. Notice that clustering only needs to be applied to the training data set which can be done offline. Given the learned codebooks, the prediction efficiency at the testing stage largely depends on the calculation of frequency histograms. For a 30 s track with approximately 2500 MFCC features and a codebook with size of 3000, calculation of a single histogram takes about 0.06 second, and the calculation of multiple histograms can be done in parallel.

Table 3
Comparison with other approaches on the GTZAN data set with the same experimental setup of 10-fold cross validation.

References	Accuracy
The proposed approach	**81.70 ± 1.85%**
Panagakis et al. (2008)	78.2 ± 3.82%
Li and Ogihara (2006)	78.5 ± 4.07%
Lidy and Rauber (2005)	74.9%
Tzanetakis and Cook (2002)	61.0%

4.3. Hyperparameter selection

In this section, we examine the influence of the hyperparameters on the performance of the proposed BOF representation model. Three hyperparameters that may potentially impact the performance of our method are distance measure for histogram

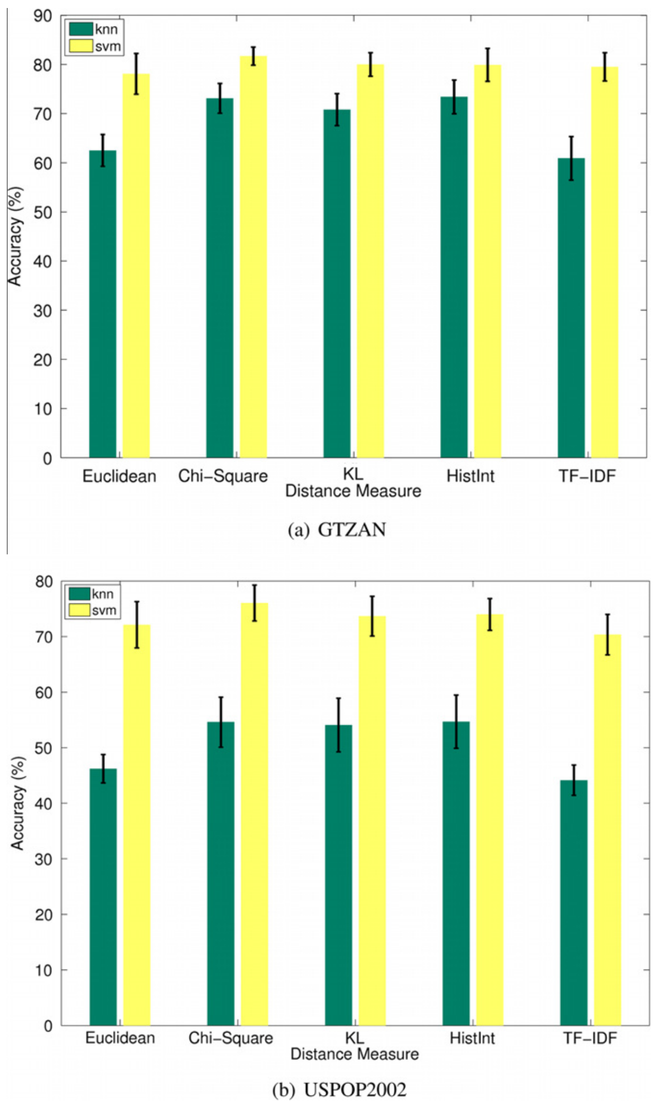


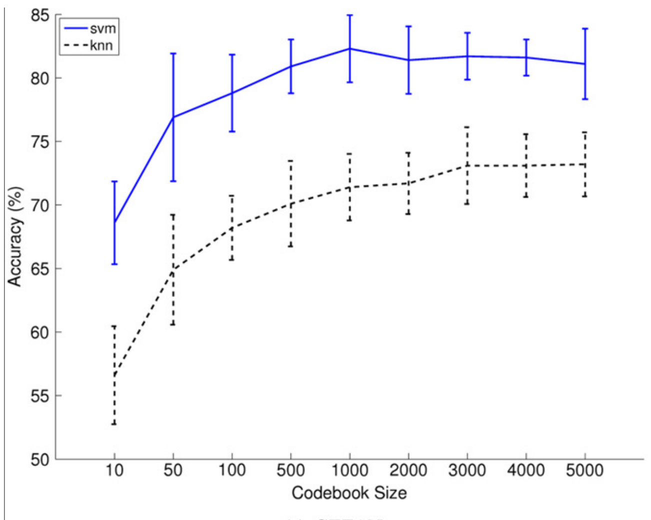
Fig. 4. Comparison of different distance measures for BOF model.

comparison, codebook size, and number of codebooks used, which will be examined one by one in the following. The change in classification accuracy is monitored by varying the hyperparameter being studied and fixing the other two hyperparameters.

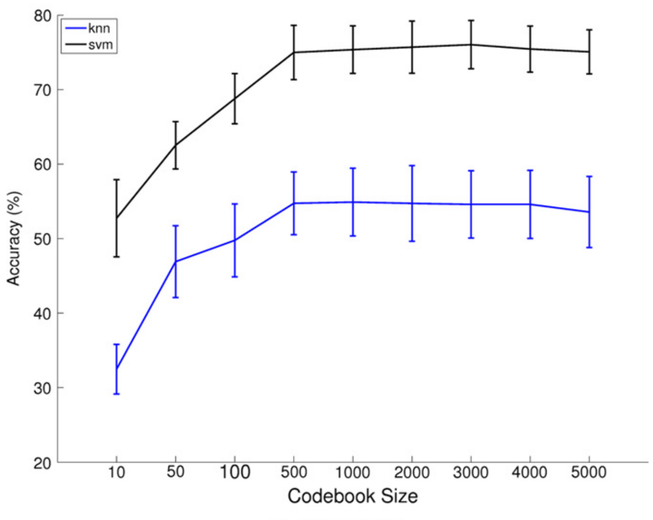
First, we compare the classification performance achieved by different distance measures discussed in Section 3.1 for comparing histograms. Fig. 4 shows the bar plots of classification accuracies on two data sets using different distance measures. From the plots, we note that χ^2 distance achieves the best classification performance on average for both K-NN and SVM classifiers. The results obtained by KL divergence and histogram intersection are also competitive with χ^2 distance but still worse. The Euclidean distance and TF-IDF do not perform as well as other distance measures. This empirically justifies our choice of χ^2 distance for histogram comparison in the BOF approach.

Next, we examine the influence of codebook size on classification performance for our BOF model. The size of codebook is varied from 10 up to 5000 in approximately log scale for small codebook sizes and linear scale for larger sizes, and the classification tests are repeated for codebooks with different sizes using both K-NN and SVM classifiers. The classification results with varying sizes of codebooks are shown in Fig. 5, where the x coordinate represents the sizes of the codebook, and the y coordinate denotes the accuracy rates in percentage. It can be easily seen that the performance is monotonically increasing for small codebook sizes and becomes quite stable over a broad range of sizes, which is consistent across both K-NN and SVM. Hence, given that as sufficiently large codebooks are used, the proposed model is not very sensitive to the codebook sizes. From the figure, we can identify that the optimal size is roughly in the range from 500 to 5000. Classification accuracies are not much affected within this broad range for both classifiers tested.

Finally, we investigate how classification performance is affected by the number of codebooks used. The multiple codebook based BOF model have been trained with different numbers of codebooks, varying from a single codebook up to a total number of 20 codebooks with fixed codebook size of 3000. Fig. 6 shows the plots of classification accuracies as a function of number of codebooks for both K-NN and SVM classifiers. It can be seen that the use of multiple codebooks does lead to increased classification accuracy and reduced variance regardless of the classifiers used. The performance improvement is more obvious initially and stabilized after a sufficient number of codebooks is used. Again, the definitive number of codebooks to be used is not important here.

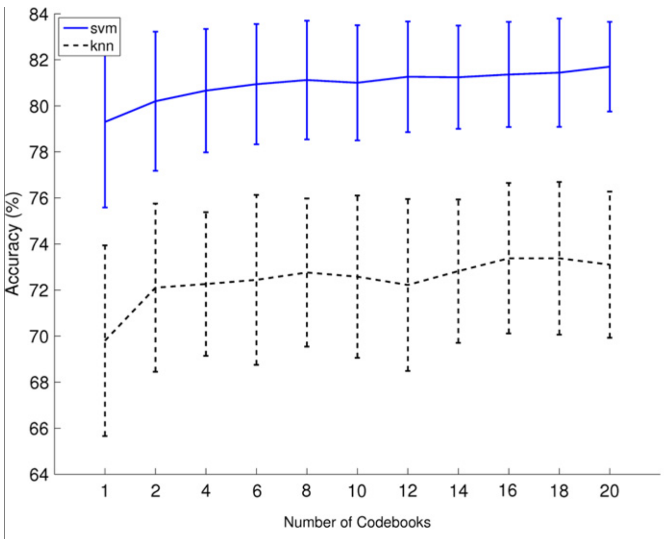


(a) GTZAN

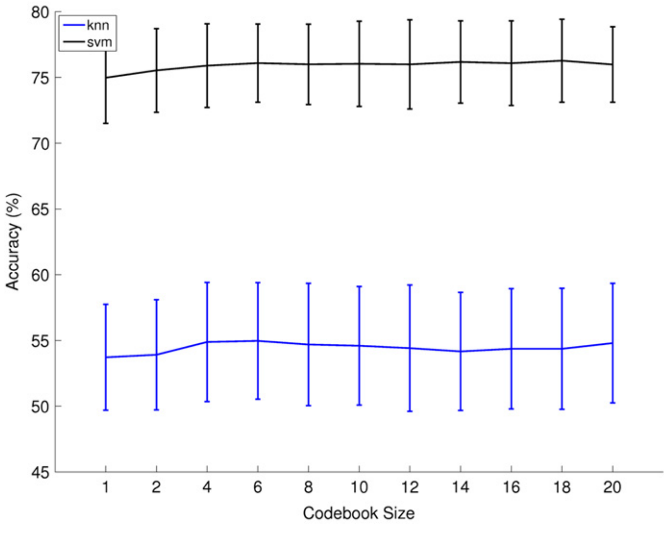


(b) USPOP2002

Fig. 5. Influence of codebook size for BOF model.



(a) GTZAN



(b) USPOP2002

Fig. 6. Influence of number of codebooks for BOF model.

Our model is not sensitive to the choice of the number of codebooks used either, as long as more than a couple of codebooks are used for obtaining the BOF representation. This is especially true for the GTZAN data set for genre classification.

In summary, it can be seen that our proposed model is not sensitive to the choice of parameters provided that an appropriate distance measure is used for histogram comparison such as the χ^2 distance adopted in our experiment. A wide range of values can be used for the other two parameters, namely the codebook size and the number of codebooks.

5. Conclusions

We have proposed an effective feature representation based on bag-of-features for music classification. The proposed model unifies previous feature representation models and has advantages in both modeling power and numerical computation. Experimental results for genre classification and artist identification on benchmark data sets have shown the superior performance of the proposed approach compared to the alternatives. In the future, we will focus on further improvement in the efficiency of the BOF model and explore the possibility of combining multiple feature types from different elements of music such as rhythm, harmony, etc., for improved classification performance. Another possible direction is the use of BOF model for music annotation (Turnbull et al., 2008). Unlike standard classification where each track is assigned to a single class and class labels are mutually exclusive, annotation is a multi-label classification problem, where each track can be assigned to any relevant label class. The proposed BOF model can be readily extended to the annotation setting and should gain an advantage over standard features being used for annotation.

Acknowledgment

This work is supported by the Australian Research Council under the Discovery Project (DP0986052) entitled “Automatic music feature extraction, classification and annotation”.

References

Berenzweig, A., Logan, B., Ellis, D., Whitman, B., 2004. A large-scale evaluation of acoustic and subjective music similarity measures. Comput. Music J. 28 (2), 63–76.

Breiman, L., 1996. Bagging predictors. Machine Learning 24 (2), 123–140.

Chang, C.-C., Lin, C.-J., 2001. Libsvm: A library for support vector machines, 2001. URL: <http://www.csie.ntu.edu.tw/cjlin/libsvm>.

Grauman, K., Darrell, T., 2007. The pyramid match kernel: Efficient learning with sets of features. J. Machine Learn. Res. 8, 725–760.

Hoffman, M., Blei, D., Cook, P., 2008. Content-based music similarity computation using the hierarchical Dirichlet process, in: Internat. Conf. on Music Information Retrieval.

Hoffman, M., Blei, D., Cook, P., 2009. Easy as cba: A simple probabilistic model for tagging music, in: Internat. Conf. on Music Information Retrieval.

Kim, H.G., Moreau, N., Sikora, T., 2004. Audio classification based on mpeg-7 spectral basis representation. IEEE Trans. Circuits Systems Video Technol. 14 (5), 716–725.

Li, T., Ogihara, M., 2006. Toward intelligent music information retrieval. IEEE Trans. Multimedia 8 (3), 564–573.

Lidy, T., Rauber, A., 2005. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification, in: Internat. Conf. on Music Information Retrieval.

Lin, C.C., Chen, S.H., Truong, T.K., Chang, Y., 2005. Audio classification and categorization based on wavelets and support vector machine. IEEE Trans. Speech Audio Process. 13 (5), 644–651.

Logan, B., 2000. Mel frequency cepstral coefficients for music modelling, in: Internat. Conf. on Music Information Retrieval.

Logan, B., Salomon, A., 2001. A music similarity function based on signal analysis, in: Internat. Conf. on Multimedia and Expo.

Mandel, M., Ellis, D., 2005. Song-level features and svms for music classification, in: Internat. Conf. on Music Information Retrieval.

Meng, A., Ahrendt, P., Larsen, J., 2007. Temporal feature integration for music genre classification. IEEE Trans. Audio Speech Lang. Process. 15 (5), 1654–1664.

Morchen, F., Ultsch, A., Thies, M., Lohken, I., 2006. Modeling timbre distance with temporal statistics from polyphonic music. IEEE Trans. Audio Speech Language Process. 14 (1), 81–90.

Nigam, K., Lafferty, J., McCallum, A., 1999. Using maximum entropy for text classification, in: IJCAI Workshop on Machine Learning for Information Filtering, pp. 61–67.

Pampalk, E., 2006. Computational Models of Music Similarity and their Application to Music Information Retrieval, Ph.D. Thesis, Vienna Inst. of Tech., Austria (2006).

Pampalk, E., Dixon, S., Widmer, G., 2003. On the evaluation of perceptual similarity measures for music, in: Internat. Conf. on Music Information Retrieval.

Panagakis, I., Benetos, E., Kotropoulos, C., 2008. Music genre classification: A multilinear approach, in: Internat. Conf. on Music Information Retrieval.

Rabiner, L., Juang, B., 1993. Fundamentals of Speech Recognition. Prentice-Hall.

Rubner, Y., Tomasi, C., Guibas, L.J., 1998. A metric for distributions with applications to image databases, in: Intenat. Conf. on Computer Vision, 1998.

Seyerlehner, K., Linz, A., Widmer, G., Knees, P., 2008. Frame level audio similarity – a codebook approach, in: Internat. Conf. on Digital Audio Effects.

Turnbull, D., Barrington, L., Torres, D., Lanckriet, G., 2008. Semantic annotation and retrieval of music and sound effects. IEEE Trans. Audio Speech Lang. Process. 16 (2), 467–476.

Tzanetakis, G., Cook, P., 2002. Musical genre classification of audio signals. IEEE Trans. Speech Audio Process. 10 (5), 293–302.

Tzanetakis, G., Ermolinskyi, A., Cook, P., 2002. Pitch histograms in audio and symbolic music information retrieval, in: Internat. Conf. on Music Information Retrieval.

Weihs, C., Ligges, U., Morchen, F., Mullensiefen, D., 2007. Classification in music research. Adv. Data Anal. Classification 1 (3), 255–291.

West, K., 2008. Novel Techniques for Audio Music Classification and Search, Ph.D. Thesis, Univ. of East Anglia, UK.

Xu, C., Maddage, N., Shao, X., 2005. Automatic music classification and summarization. IEEE Trans. Speech Audio Process. 13 (3), 441–450.

Zhang, J., Marszalek, M., Lazebnik, S., Schmid, C., 2007. Local features and kernels for classification of texture and object categories: A comprehensive study. Internat. J. Comput. Vision 73 (2), 213–238.