MUSIC CLUSTERING WITH CONSTRAINTS

Wei Peng **Tao Li**
School of Computer Science
Florida International University
{wpeng002,taoli}@cs.fiu.edu

Mitsunori Ogihara
Department of Computer Science
University of Rochester
ogihara@cs.rochester.edu

ABSTRACT

This paper studies the problem of building clusters of music tracks in a collection of popular music in the presence of constraints. The constraints come naturally in the context of music applications. For example, constraints can be generated from the background knowledge (e.g., two artists share similar styles) and the user access patterns (e.g., two pieces of music share similar access patterns across multiple users). We present an approach based on the generalized constraint clustering algorithm by incorporating the constraints for grouping music by "similar" artists. The approach is evaluated on a data set consisting of 53 albums covering 41 popular artists. The "correctness" of the clusters generated is tested using artist similarity provided by All Music Guide.

1 INTRODUCTION

For those who listen to music through the Internet, how to navigate in the ocean of on-line music is an important issue. Nowadays, everything about music is on the web — audio, lyrics, artist discographies, artist biographies, reviews, and discussions. This raises an issue of whether the on-line music data can be efficiently accessed so that the user can benefit from the existence of such large volumes of data. A solution to the issue can be given by developing efficient music assistance programs, which integrate techniques for analyzing, summarizing, indexing, classifying, and grouping music data.

This paper addresses the issue of clustering pop music into groups with respect to the artists. Clustering is the standard, effective tool for efficient organization, summarization, navigation and retrieval of a large amount of data. Information navigation by browsing through data clusters is more suitable for users who have vague information need and/or just wish to discover general contents of the data set.

The use of instance-level constraints as the background information to improve data clustering has been widely studied in machine learning in the past few years. Instance-level constraints are generally pairwise and they are of two types: the *positive constraint* is one that specifies that two instances must belong to the same cluster, and the *negative constraint* is one that specifies that two instances must belong to different clusters. These instance-level constraints have been used in learning distance/dissimilarity measures [3,4,7,10,18], modifying objective criteria for cluster evaluation [1], and improving optimization procedures [2,16,17].

The constraints come naturally in the context of music applications. For example, constraints can be generated from the background knowledge (e.g., two artists share similar styles) and the user access patterns (e.g., two pieces of music share similar access patterns across multiple users). In this paper, we investigate the problem of content-based music clustering with such instance-level constraints. In particular, we adapt a generalized constraint clustering algorithm based on K-means and discuss approaches to automatically generate constraints. The rest of the paper is organized as follows: Section 2 introduces the algorithm for constraint-based clustering, Section 3 discusses various approaches to generate constraints in music applications, Section 4 describes the content-based feature extraction, Section 5 show our experimental results, and Section 6 provides conclusions and presents open questions.

2 CONSTRAINT-BASED CLUSTERING

This section provides some background on the K-means algorithm and then discusses the constraint-based clustering algorithm following the exposition in [7].

2.1 K-means Clustering

The problem of clustering data arises in many disciplines and has a wide range of applications. Intuitively, clustering is the problem of partitioning a finite set of points in a multi-dimensional space into classes (called *clusters*) so that (i) the points belonging to the same class are "similar" and (ii) the points belonging to different classes aren't [9].

K-means is a popular clustering algorithm where the input data set is partitioned into K groups, where the number K is specified by the user. The quality of partition into K clusters can be viewed as the *quantization error* as follows:

$$E = \frac{1}{2} \sum_{j=1}^{K} \sum_{s \in C_j} (\bar{c}_j - s)^2. \quad (1)$$

Here $C_1, \ldots, C_K$ are the K clusters and and $\bar{c}_1, \ldots, \bar{c}_K$ their centroids. The goal of K-means is to minimize this quantization error, which is accomplished iteratively by alternating between the *allocation step* and the *evaluation step*. In the former each data point is allocated to the cluster whose centroid is the closest to it so as to minimize the quantization error with respect to the current centroids, while in the latter, the centroid of each cluster is updated based on the new allocation. The solution of $\frac{\delta E}{\delta \bar{c}_j} = 0, 1 \leq j \leq K$ provides the rule that sets the centroid $\bar{c}_j$ to be the mean of the data points in C_j for all $j, 1 \leq j \leq K$. Repetition of these alternating steps monotonically decreases the average distance of data points from their corresponding centroid. The algorithm converges when there is no change in the data allocation.

2.2 Constraint-based Clustering

Following [4] we define the concept of constraint-based clustering for music similarity. We deal with positive constraints, which are given as a list of pairs that are expected to belong to the same cluster, and negative constraints, which are given as a list of pairs that are expected to belong to different clusters. We modify the the objective function so that penalty is added for each constraint that is not satisfied. For a positive constraint (s_i, s_j) (that is, s_i and s_j must be in the same cluster), the penalty (in the case where they go to different clusters) is the squared distance between their cluster centroids. For a negative constraint (s_i, s_j) (that is, s_i and s_j must be in different clusters), the penalty (in the case where they go to the same clusters) is the squared distance between the centroids that are the closest and the second closest to either s_i or s_j. In both cases, we use the centroids to determine the penalty so as to treat equally constraint violations within a cluster, and we use squared distance since the quantization error is based on squared distance. Also, our choice of the two centroids in the penalty for an unsatisfied negative constraint is based on the idea that the constraint would be satisfied if one vector were assigned to the cluster of the closest centroid and the other were to the cluster of the other centroid. Note that the closest centroid closest is the centroid of the cluster to which s_i and s_j belong.

The exact formula for the objective function is given bellow:

$$\begin{aligned} CE &= \frac{1}{2}(E + PM + PC) && (2) \\ &= \frac{1}{2}\left(\sum_{j=1}^{K}\sum_{s \in C_j}(\bar{c}_j - s)^2 + PM + PC\right), \\ PM &= \sum_{(s_i,s_j)\in M} p_{ij}^{m}(1 - \Delta(y(s_i), y(s_j))), && (3) \\ PC &= \sum_{(s_i,s_j)\in C} p_{ij}^{c}\Delta(y(s_i), y(s_j)), && (4) \\ p_{ij}^{m} &= (\bar{c}_{y(x_i)} - \bar{c}_{y(x_j)})^2, && (5) \\ p_{ij}^{c} &= (\bar{c}_{y(x_i)} - \bar{c}_{ij}^{*})^2. && (6) \end{aligned}$$

Here M and C respectively represent the set of positive constraints and the set of negative constraints, p_{ij}^m and p_{ij}^c are respectively penalty parameters for the positive and for negative constraints, and the value of $y(s_i)$ is the index of the cluster to which the data point s_i belongs. Also, Δ is the Kronecker delta function defined by: $\Delta(x, y) = 1$ if $x = y$ and 0 otherwise. That is, the penalty p_{ij}^m is added only if $(s_i, s_j) \in M$ but s_i and s_j do not belong to different clusters and the penalty p_{ij}^c is added only if $(s_i, s_j) \in C$ but s_i and s_j belong to the same cluster. Furthermore, $\bar{c}_{ij}^*$ is the centroid that is the next closest to either s_i and s_j.

Like K-means, the constraint-based clustering algorithm is iterative, alternating between the allocation step and the centroid update step. In the allocation step, the goal is to minimize the generalized constrained vector quantization error in Eq. 3. This is achieved by assigning instances so as to minimize the proposed error term. For pairs of instances in the constraint set, the quantization error CE is calculated for each possible combination of cluster assignments, and the instances are assigned to the clusters so that CE is minimized. In the update step, the centroids are cluster centroids. As in K-means, the first order partial derivatives of CE with respect to each centroid is evaluated and the solution that makes all these derivatives equal to zero are obtained. Figure 1 presents the procedure of the constraint-based clustering algorithm (see [4] for more detail).

```
ALGORITHM Constraint-based Clustering
Input: Data set $S$, positive constraint set $M$,
       negative constraint set $C$, number of clusters $K$
Output: Cluster assignment $Y$

1:   Initialization:
     a. Create the $m$ neighborhoods from $M$ and $C$
     b. if $m \geq K$, initialize using weighted farthest-first
     traversal starting from the largest neighborhood
     else initialize with centroids of neighborhood sets
     and remaining clusters at random

2:   Iteration:
     while stopping criterion is not met
2.1:     Step I: Assign each data point to either cluster or
         noise set such that it minimizes $CE$
2.2:     Step II: Update the centroids for each cluster

3:   Return $Y$
```

Figure 1. The Algorithm Description.

3 CONSTRAINTS GENERATION

The constraints come naturally in the context of music applications. The following gives a summary on different approaches to generating constraints:

- *Background Knowledge*: Constraints can be generated from the background knowledge. If we already know that two songs are of the same styles, or formally, if we know two songs have the same cluster labels, then they must be in the same cluster (e.g., a positive constraint). Similarly, if it is known that two songs are of different styles, then they should be in different clusters (e.g., a negative constraint).
- *User-access Patterns*: It is well known that the perception of music is subjective to individual users. Different users can have totally different opinion for the same pieces of music. If two pieces of music share similar access patterns across multiple users, they should be similar in human perception, and consequently should be put in the same cluster. Thus user-access patterns can be used to generate constraints for music clustering.
- *Subjective Similarity Measures*: In the context of music information retrieval, when the initial retrieval results are unsatisfied, relevance feedback methods will be applied to improve the quality of retrieval results through iteration of user relevance feedback. User feedback can be regarded as a way of providing subjective similarity measures to the music and this can also be used to generate constraints.
- *Complementary and Diverse Music Information Sources*: Music data are naturally multi-modal, in the sense that they are represented by multiple sets of features. For example, the personnel-related features (the producer, the supporting musicians, and the record label), the acoustic features (which summarize the voice and the background audio), and the text features (.e.g., the song lyrics, the reviews). These features are complementary and diverse, and can be used to generate constraints for clustering. For example, if two piece of music have the same personnel-related features, then they can be considered to be similar based on content.

In the experiments in this paper, the constraints are generated by background knowledge, i.e., from the known class labels. Exploring the effects of various constraint generation methods is one of our future goals.

4 CONTENT-BASED FEATURE EXTRACTION

There has been a considerable amount of work in extracting descriptive features from music signals for music genre classification and artist identification [8, 11, 13–15]. In our study, we use timbral features along with wavelet coefficient histograms. The feature set consists of the following three parts and totals 35 features.

4.1 Mel-Frequency Cepstral Coefficients (MFCC)

MFCC is a feature set that is popular in speech processing and is designed to capture short-term spectral-based features. To obtain the feature, the logarithm of the amplitude spectrum is computed for each frame based on short-term Fourier transform, where the frequencies are divided into thirteen bins using the Mel-frequency scaling. (The "cepstrum" is the name coined for this logarithm.) After taking the logarithm of the amplitude spectrum, the frequency bins are grouped and smoothed according to Mel-frequency scaling, which is design to agree with perception. MFCC features are generated by decorrelating the Mel-spectral vectors using discrete cosine transform. In this study, we use the first five bins, and compute the mean and variance of each over the frames.

4.2 Short-Term Fourier Transform Features (FFT)

This is a set of features related to timbral textures and is not captured using MFCC. It consists of the following five types. More detailed descriptions can be found in [15]. *Spectral Centroid* is the centroid of the magnitude spectrum of short-term Fourier transform and is a measure of spectral brightness. *Spectral Rolloff* is the frequency below which 85% of the magnitude distribution is concentrated. It measures the spectral shape. *Spectral Flux* is the squared difference between the normalized magnitudes of successive spectral distributions. It measures the amount of local spectral change. *Zero Crossings* is the number of time domain zero crossings of the signal. It measures noisiness of the signal. *Low Energy* is the percentage of frames that have energy less than the average energy over the whole signal. It measures amplitude distribution of the signal.

4.3 Daubechies Wavelet Coefficient Histograms (DWCH)

Daubechies wavelet filters are ones that are popular in image retrieval (see [5]). To extract DWCH features, the db_8 filter with seven levels of decomposition is applied to thirty seconds of sound signals. After the decomposition, the histogram of the wavelet coefficients is computed at each subband. Then the first three moments of a histogram, i.e., the average, the variance, and the skewness, are used [6, 13] to approximate the probability distribution at each subband. In addition, the subband energy, defined as the mean of the absolute value of the coefficients, is also computed at each subband. A few trials reveal that of the seven subbands of db_8 (1: 11025–22050 Hz, 2: 5513–11025Hz, 3: 2756–5513Hz, 4: 1378–2756Hz, 5: 689–1378Hz, 6: 334–689Hz, 7: 0–334Hz), subbands 1, 2, and 4 show little variation. We thus choose to use only the remaining four subbands, 3, 5, 6, and 7, for our experiments. In fact, the subbands match the models of sound octave-division for perceptual scales [12].

5 EXPERIMENTS

In this section, we perform experiments to evaluate whether the clustering algorithms based on minimizing

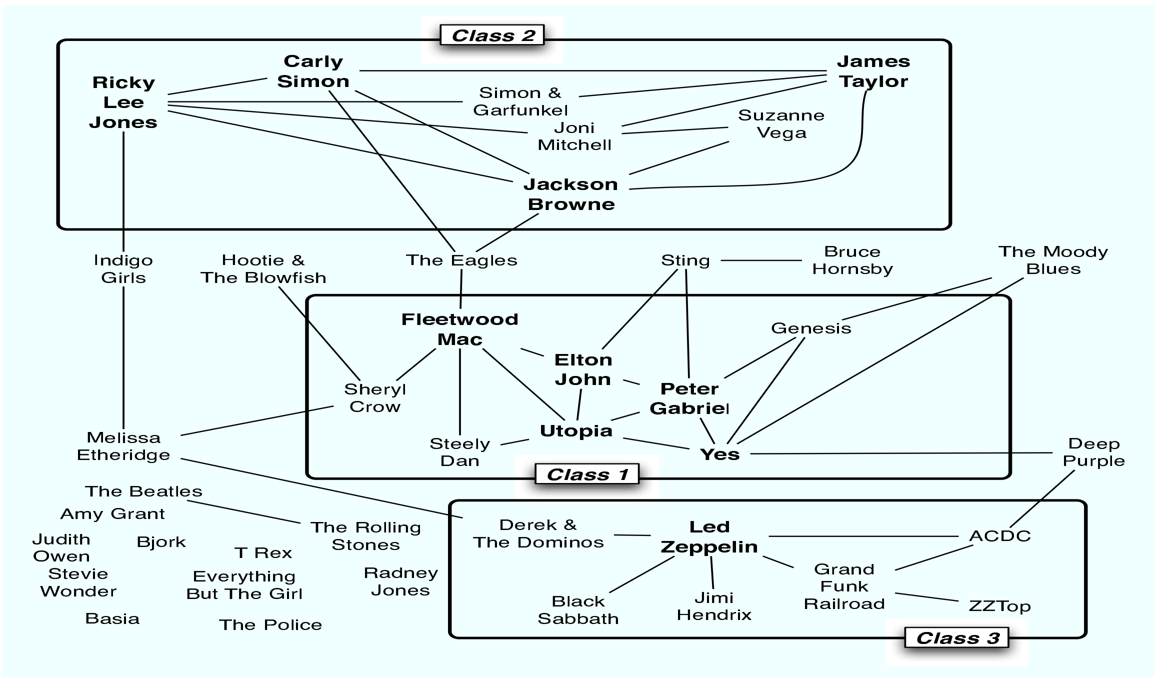

Figure 2. The artist similarity graph. The names in bold are "core" nodes.

disagreement can be more powerful than unimodal methods.

5.1 Data Description

Our experiments are performed on the dataset consisting of 300 songs from 53 albums of a total of 41 artists.

To divide songs into classes, we choose to use the similarity information among artists available at the All Music Guide artist pages (http://www.allmusic.com), assuming that this information is provided by experts. By examining the All Music Guide pages of the 41 artists, if the name of an artist X appears on the list of artists similar to Y, it is considered that X is similar to Y. To form classes artists having a large number of neighbors are selected as core nodes. Core nodes that are neighbors to each other are put into the same class. The other artists that are neighbors to each core nod are selected to be in the class of the core so that each class is separated by at least one node in between. All the remaining artists are put in a separate class. Table 1 shows the classes of the artists and Figure 2 shows the similarity graph along with the classes. Our goal is to classify each song out of the 300 into one of the four classes corresponding to the artist by analyzing its audio contents.

5.2 Evaluation Measures

As discussed above, we use the cluster structures obtained from All Music Guide as labels to evaluate the clustering performance. We use *purity*, *entropy*, and *accuracy* as our performance measures (see [19] for discussions of these).

To describe how these measures are calculated, let $C_1, \ldots, C_K$ be the input classes, $N_1, \ldots, N_K$ their size,

Class	Members
1	{ *Fleetwood Mac, Yes, Utopia, Elton John, Genesis, Steely Dan, Peter Gabriel* }
2	{ *Carly Simon, Joni Mitchell, Suzanne Vega, Ricky Lee Jones, Simon & Garfunkel, James Taylor* }
3	{ *AC/DC, Black Sabbath, ZZ Top, Led Zeppelin, Grand Funk Railroad, Derek & The Dominos* }
4	All the remaining artists

Table 1. Artist classes.

and $N = N_1 + \cdots + N_K$. Also, let $D_1, \ldots, D_K$ be the output clusters and $M_1, \ldots, M_K$ their size. For each $(i, o), 1 \leq i, o \leq K$, let $H_{i,o} = \|C_i \cap D_o\|$.

Purity measures how large a portion of each output cluster comes from a single input class [19]. For each $o, 1 \leq o \leq K$, let $p(o)$ be the index $i, 1 \leq i \leq K$, that maximizes $H_{i,o}$, where a tie can be broken arbitrarily. The purity of D_o with respect to $C_1, \ldots, C_K$ is defined to be $\frac{H_{p(o),o}}{M_o}$. The purity of $D_1, \ldots, D_K$ with respect to $C_1, \ldots, C_K$ is then defined to be the sum of individual purity weighted proportionally to the size of output clusters. In other words,

$$\begin{aligned} \mathit{Purity} &= \sum_{o=1}^{K} \frac{M_o}{N} \cdot \frac{H_{p(o),o}}{M_o} \\ &= \frac{1}{N} \sum_{o=1}^{K} \max\{H_{i,o} \mid 1 \leq o \leq K\}. \quad (7) \end{aligned}$$

Note that $p(1), \ldots, p(K)$ are determined independently and thus $p(o)$ may have the same value for multiple values of o. Generally speaking, the higher the purity, the better the clustering quality.

Entropy measures how classes distributed on various clusters.

$$\begin{aligned} \mathit{Entropy} &= -\frac{1}{\log K}\sum_{i=1}^{K}\frac{N_i}{N}\sum_{o=1}^{K}\frac{H_{i,o}}{N_i}\log\frac{H_{i,o}}{N_i} \\ &= -\sum_{i=1}^{K}\sum_{o=1}^{K}\frac{H_{i,o}}{N\log K}\log\frac{H_{i,o}}{N_i}, \end{aligned} \quad (8)$$

where the logarithm is base 2. Generally speaking, the smaller the entropy value, the better the clustering quality.

Accuracy measures under the assumption that there is a one-to-one correspondence between the input classes and the output clusters, how accurately the data allocation is.

$$\begin{aligned} \mathit{Accuracy} &= \max_{\pi}\sum_{i=1}^{K}\frac{N_i}{N}\max_{q}\sum_{i=1}^{K}\frac{H_{i,q(i)}}{N_i} \\ &= \frac{1}{N}\max_{\pi}\sum_{i=1}^{K}H_{i,q(i)}, \end{aligned} \quad (9)$$

where q ranges over all permutations of $\{1,\ldots,K\}$. Generally speaking, the larger the accuracy value, the better the clustering quality.

Note that purity and accuracy do not necessarily agree with each other. Consider an example in which three classes of size 10 each are given, the first class is evenly split between first and second clusters, and and the whole second and third classes are allocated to the third cluster. The purity for the three clusters are 1.0, 1.0, and 0.5 respectively. Since the cluster sizes are 5, 5, and 20 respectively, the purity for the clustering is $1.0\cdot\frac{5}{30}+1.0\cdot\frac{5}{30}+0.5\cdot\frac{20}{30}=0.6667$. On the other hand, by letting class i correspond to cluster i maximizes the formula for the accuracy to $\frac{5+10}{30}=0.5$.

5.3 Analysis of the Results

30 constraints (including 10 positive constraints and 20 negative constraints) are randomly generated from the cluster labels. We compare the results of constraint clustering with the results obtained when clustering is applied on content without any constraints. Table 2 presents the experimental results over ten independent trials.

Measurement	Purity	Entropy	Accuracy
Without Constraints	0.436	0.731	0.438
With Constraints	0.471	0.723	0.472

Table 2. Performance comparison. The numbers are obtained by averaging over ten trials.

We observe that constraint-based clustering achieves better performance (i.e., higher purity and accuracy values and lower entropy values) than clustering without any constraints, and that the performance of purity, entropy, and accuracy relative to the other is consistent in our comparison, i.e., higher purity values correspond to lower entropy values, and to higher accuracy values. Note that different evaluation measures consider different aspects of the clustering results. For example, the entropy measure takes into account the entire distribution of the data in a particular cluster and not just the largest class as in the computation of the purity. The accuracy considers the relationships among all pair class-clusters. We hope that these different measures would provide enough information to understand the results of our experiments.

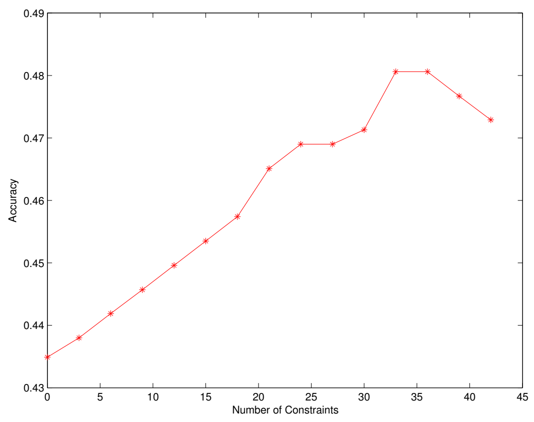

Figure 3. Comparisons of the clustering accuracy as a function of constraint size.

Figure 3 illustrates the effects of the constraint size. The X-axis of figure shows the number of constraints while the Y-axis shows the clustering accuracy. Here different constraint sizes are tested to investigate the effect of the size of the constraint on the overall clustering performance. An approximate $1:2$ ratio of the number of positive constraints to the number of negative constraints is maintained throughout the experiment. We observe that as the constraint set size increases, the accuracy measures steadily improves and flattens out after 40. Then, after that, it looks as if the accuracy beings to decrease. This may suggest that two many constraints may force our clustering algorithm to over-fit.

The total number of constraints to specify relations between data elements in an N-element data set is $N(N+1)/2$. In our case, $N=300$ so the number is $44,850$. The number of constraints we used is less than 0.1% of this and thus may look very small. However, the total number of class relations four a K-class data set is K^2, which is in our case 16. Thus, with 40 constraints we can expect that the class relations are represented at least twice on average. The conspicuous decline in accuracy may suggest that adding more than three constraints per class relation can lower the performance.

6 DISCUSSIONS AND OPEN QUESTIONS

In this paper, we study the problem on clustering music songs in the presence of constraints. In particular, we present a constraint-based clustering framework and discusses various approaches to generate constraints. Experimental results on a data set consisting of 300 songs from 41 artists of 53 albums show the effectiveness of our approach.

There are several natural avenues for future research. The first natural direction is to investigate different approaches for constraints generation. Second, another interesting direction is on music annotation. How can we automatically and efficiently generate music style or similarity information? Note we did not agree completely with the artist similarity obtained from All Music Guide, but nonetheless used it as the ground truth to evaluate our algorithms in the experiments. Can we incorporate the opinions from music experts or take into account the views from individual users? Third, it would also be interesting to evaluate the quality of the generated constraints. Finally, can we determine the number of constraints?

Acknowledgments

Tao Li is partially supported by NSF grants HRD-0317692 and IIS-0546280. Wei Peng is supported by a Florida International University Presidential Graduate Fellowship.

7 REFERENCES

[1] K. P. Bennett A. Demiriz and M. J. Embrechts. Semi-supervised clustering using genetic algorithms. In *Proceedings of the 5th Conference on Artificial Neural Networks in Engineering (ANNIE'99)*, pages 809–814, 1999.

[2] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*, pages 27–34, 2002.

[3] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 59–68, ACM, 2004.

[4] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, pages 81–88, 2004.

[5] I. Daubechies. *Ten lectures on wavelets*. SIAM, Philadelphia, PA, 1992.

[6] A. David and S. Panchanathan. Wavelet-histogram method for face recognition. *Journal of Electronic Imaging*, 9(2):217–225, 2000.

[7] I. Davidson and S. S. Ravi. Clustering with constraints: feasibility issues and the k-means algorithm. In *Proceedings of the 5th International Conference on Data Mining (SDM'05)*, SIAM, 2005.

[8] J. Foote and S. Uchihashi. The beat spectrum: a new approach to rhythm analysis. In *Proceedings of the 2nd International Conference on Multimedia and Expo (ICME'01)*, IEEE, 2001.

[9] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[10] D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*, pages 307–314, 2002.

[11] J. Laroche. Estimating tempo, swing and beat locations in audio recordings. In *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA'01)*, 2001.

[12] G. Li and A. A. Khokhar. Content-based indexing and retrieval of audio data using wavelets. In *Proceedings of the International Conference on Multimedia and Expo (ICME'00)*, pages 885–888, IEEE, 2000.

[13] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *Proceedings of the 26th International Conference on Development in Information Retrieval (SIGIR'03)*, pages 282–289, ACM, 2003.

[14] L. Rabiner and B. H. Jiang. *Fundamentals of Speech Recognition*. Prentice-Hall, NJ, 1993.

[15] G. Tzanetakis and P. E. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[16] K. Wagsta and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the 17th International Conference of Machine Learning (ICML'00)*, pages 1103–1110, 2000.

[17] K. Wagsta, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference of Machine Learning (ICML'01)*, pages 577–584, 2001.

[18] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Neural Information Processing 15 (NIPS'03)*, pages 505–512, 2003.

[19] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.