A Content-based Similarity Search for Monophonic Melodies

(sequential classification by SVM's, subsequence matching and use of Levenshtein distance for pruning)

Shashank Srivastava(Y5429), Snigdha Chaturvedi(Y5452)

Guide: Dr. Arnab Bhattacharya
Dept of Computer Science and Engineering
IIT Kanpur

November 10, 2008

Contents

ABSTRACT

Feature extraction based methods have been used for identifying genres in musical pieces, and there are also attempts to predict the popularity of songs by statistical methods such as clustering. Transportation distances have been extensively used, especially in image searches. Advantages of such methods are their incorporation of the notions of continuity and partial matching. We proposed to combine both methods: we test different kinds of feature representations to cluster songs in a database through global level parameters. At the time of search, the input Midi sequence is classified to one of the clusters by an SVM, and the songs in the sequence with the minimum Levenshtein distances are returned. The sequential approach is seen to yield encouraging results on two standard datasets, for Midi inputs by amateur players.

1 Introduction

Data-mining on time series data affords issues such as correlation, high dimensionality, missing values. Also, it has tremendous applicability in examples such as stock markets, ECG and weather prediction. We took the interesting case of similarity search in monophonic music to learn about the same.

Similarity detection is the essence of all genres of multimedia, and especially musical searches in particular: query by humming, query-by-example, query-by-style, genre or artist.

Feature extraction based methods have been used for purposes of identifying genres, artists and Query by example retrievals for musical pieces. Transportation distances such as EMD have been extensively used, especially in image searches. Advantages of such methods are their incorporation of the notions of continuity and partial matching.

We chose the simple case of monodic melodies like a user might input from a MIDI keyboard, as this avoids intricacies like identifying the soprano and melody matching, the which have not been solved reliably. The data is essentially close to a time series.

2 Time series Representation

For the purpose of our study, we considered using midi versions of audio files. Several datasets in other formats exist, such as other representations as pitch vectors(pv), *.krn etc. However, it was felt that most of these notations were unnecessarily complicated for a study of monophonies, and called for considerable in-depth knowledge of domain.

Also, an existing Matlab toolkit for Midi, developed by the University of Jyvaskyla, could greatly expedite visualization of data, and feature extraction. The midi format contains data to recreate a given musical piece.

A midi file for our case, contains the following data which can be extracted into matrices by Matlab:

ONSET (BEATS)	DURATION (BEATS)	MIDI CHANNEL	MIDI PITCH	VELOCITY	ONSET (SEC)	DURATION (SEC)
0	0.9000	1.0000	64.0000	82.0000	0	0.5510
1.0000	0.9000	1.0000	71.0000	89.0000	0.6122	0.5510
2.0000	0.4500	1.0000	71.0000	82.0000	1.2245	0.2755
2.5000	0.4500	1.0000	69.0000	70.0000	1.5306	0.2755
3.0000	0.4528	1.0000	67.0000	72.0000	1.8367	0.2772
3.5000	0.4528	1.0000	66.0000	72.0000	2.1429	0.2772
4.0000	0.9000	1.0000	64.0000	70.0000	2.4490	0.5510
5.0000	0.9000	1.0000	66.0000	79.0000	3.0612	0.5510
6.0000	0.9000	1.0000	67.0000	85.0000	3.6735	0.5510
7.0000	1.7500	1.0000	66.0000	72.0000	4.2857	1.0714

A 5 second sample of Midi Data

The first column indicates the onset of the notes in beats, and the second column the duration of the notes in these same beat-values. The third column denotes the MIDI channel (1-16), which is constant in the case of monophonic melodies, as all notes are played on the same channel. However, the converse is not true, and removing other channels from a polyphonic midifile, even if it still contains the principal melody, does not make it monophonic.

The fourth column the MIDI pitch, where the middle C note (C4: 261.63 Hz) is denoted as 60 (see Appendix). The fifth column is the velocity describing how loud the note is played (0-127). The last two columns correspond to the first two (onset in beats, duration in beats) except that seconds are used instead of beats.

3 Datasets

Monophonic data from two different sources was used for the project. Initially, the only dataset consisted of 48 midi-files containing Chinese and English melodies,from the QBSH corpus owned by MIR (Multimedia Information Retrieval) Lab at National Tsing Hua University, Taiwan.

The Finnishfolk song dataset containing 8613 midi sequences, which was promised earlier, was procured with delay due to technical difficulties from the University of Jyvaskyla, Finland.

Dataset	Instances	Labeled
QSBH midi	48	Yes
Finnish Folk Songs	8613	No
Midi keyboard inputs	4200	No

Also, recordings from Midi keyboards played by people with different levels of proficiency were used to test the similarity search. Most of these songs were those already existing in the database.

4 Feature Extraction and Clustering

Audio classification systems generally combine two processing stages: feature extraction followed by classification. In our case, a classification would help detecting similar melodies as a preliminary step, as similar musical pieces would lie in the same class. For the purpose of feature extraction from audio signals, a variety of low-level signal features have been used, including parameters such as the zero-crossing rate, signal bandwidth, spectral centroid, and signal energy. However, these approaches are characterized as "timbre similarity" to emphasize that they are based on distributions of short-term features and ignore most temporal structure. Also, in our case, the data in the form of note sequences is highly discrete, and most of these features don't make sense.

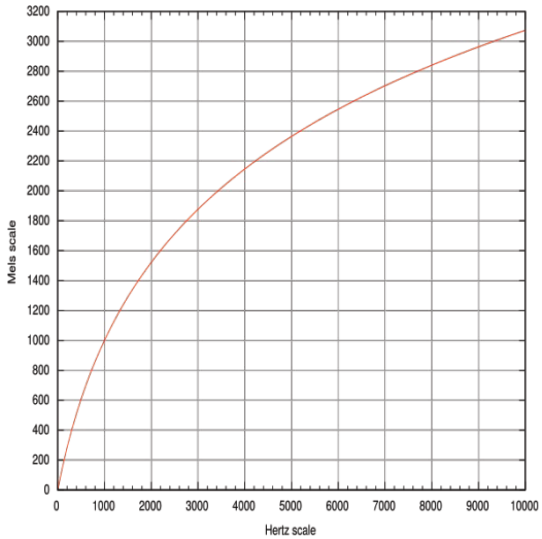
Instead, a feature vector representative of the entire series is needed. There are a number of papers that classify music on the basis of 'timbre similarity' judged from short window-frames (1ms) alone. However, this is because extraction of melody for polyphonic music is a non-trivial task, and the temporal structure for polyphonies is much more complex.

For monophonic melodies, the representation of the overall structure is more feasible as the time-series involves fewer dimensions. Frequency was first converted from Midi format to its physical value using the conversion described in Appendix 1. The Holy Grail of feature extraction was the acquirement of a meaningful feature vector, that could be used for clustering of songs without pondering on detail:

1. DFT's and DWT's have been frequently used in other time-series domains with considerable success.
2. Another approach was to not depend on the frequency waveform alone, but represent a song by a feature vector contain statistical information about the notes and beat-patterns.

Mel Scale: Humans have an auditory range of 20-20000Hz, however human perception of auditory signals doesn't correspond exactly to frequency .i.e. two notes that 'sound' equally similar to another may not be equidistant on the logarithmic frequency spectra.

Many musicians and psychologists prefer the Mel Scale, a perceptual scale of pitches judged by listeners to be equal in distance from one another.



Source: http://en.wikipedia.org/wiki/Mel_scale

The frequency values were converted to Mels, as domain knowledge suggested that this would be more representative, and assist statistical analysis:

$$m = 1127.01048 \log_e \left(1 + \frac{f}{700}\right)$$

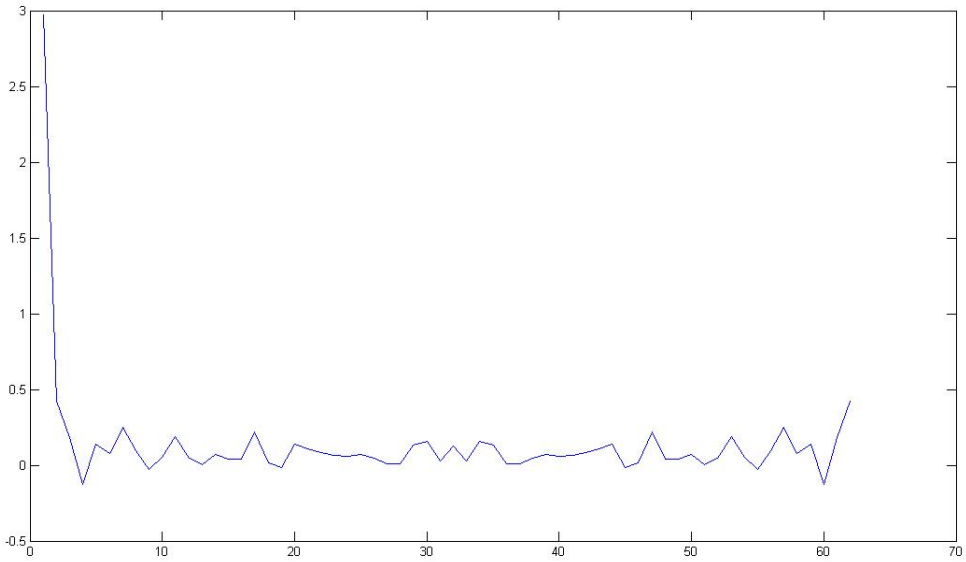
4.1 DFT's

MFCC coefficients were first proposed by Foote as an improvement on DFT's. Different from previous representations that tried to estimate parameters of an assumed production model such as Gaussian mixtures, cepstral analysis tries to estimate the model directly using homomorphic filtering. First, the audio segment is multiplied by a Hamming window to smooth out discontinuities at the beginning and end of the segment. Then, the Fast Fourier Transform (FFT) of the windowed segment is computed. We then compute the logarithm followed by the inverse FFT, or a Discrete cosine transform in certain cases.

Diagram from (3) :Foote, J.T.: Content-based retrieval of music and audio (1997)

MFCC coefficients from 1 to 5 were used to represent each melody of typical length 30-80 notes. The method was used to cluster and classify on a dataset of 48 Chinese and English melodies described in the next section in detail. However, the transformation was found to be not useful at all for the purposes of clustering (66% accuracy on the classification task of section 4.3), and the vector was dominated entirely by its first term.

Without using the normalized first coefficient(that corresponds to the sum of the series), the accuracy was slightly worse at 62%.



MFCC coefficients for a song from Finnish Folk Song database

In retrospection, MFCC's are generally used to extract low level features with small window resolution. At such a low resolution, the waveform was not smooth, and didn't yield any meaningful results. Also, in course of study, we found that results were not counter-intuitive as DFT's look for strong periodic global patterns, instead of localized details that are typical of our Midi datasets.

4.2 Statistical Features

Music classification has also been approached by accumulating statistical features that are representative of the overall 'genre' or 'type' of song. Such features have especially been used in Genre and artist prediction problems.

Sometimes, datamining through such statistical parameters has yielded important and non-intuitive insights into musical compositions and distinguishing features of various artists, such as Bach's chorals, as different individuals can have predilections for different notes. Such parameters take into consideration the pitch, beats, tempo and variations in a piece. We experimented with 12 such features, and chose the best through combinations of various features and cross-validation.

- Avg number of notes/ time for a song
- Average pitch of all notes used
- Maximum difference in pitch
- Variance of frequency
- Fraction of notes outside 1 SD of frequency
- Starting note family
- Fraction of 'sharp' notes
- Beat duration
- Avg duration of a note
- Total silent time in composition/composition length
- Avg note velocity
- Song length

The features were tested on a dataset of 48 English and Chinese songs. (courtesy QBSH(Query By Singing/Humming corpus) by MIR (Multimedia Information Retrieval) Lab at CS Dept. of National Tsing Hua University, Taiwan, as described in section 4.3.

4.3 Classification by feature representation on Chinese-English Dataset:

The two classes were represented fairly equally (25 Chinese and 23 English songs), and the problem seemed amenable to classification by binary classifiers such as Support Vector Machines, to verify if the proposed feature-vectors were adequately representative of song-content.

Weka's SVM based on sequential minimum Optimization was used to classify the dataset. Performance was evaluated by 10-fold cross-validation and 66-33 split.

Total Instances	48
Correctly Classified	82.35%
Incorrectly classified	17.65%

	Chinese	English
Chinese	4	3
English	0	10

Sample Confusion matrix

Classification by Weka's SVM using final set of attributes

There was a high correlation of the class with Song length, which seemed coincidental within this sparse dataset. Hence, it was not used in further training and testing. It was felt that a knowledge-bling algorithm such as an SVM would erringly give higher weight to these features, which wont generalize.

The following features were found to be the best for classification and clustering:

1. Avg number of notes/ time for a song
2. Average pitch of all notes used
3. Variance of frequency
4. Beat duration
5. Avg duration of a note
6. Total silent time in composition/composition length
7. Avg note velocity

The current scheme of feature representation achieved an accuracy of around 80% against labeled data, which was good for a preliminary clustering according to song types, especially as some of the songs could'nt be discerned even by a human test subject.

4.4 Clustering

The proposed set of features found in the above section is used by us to cluster the entire database of all songs (8661) by Expectation Maximization clustering, as it doesn't need a predefined number of clusters. Songs within a cluster, based on the performance on the Chinese/English song dataset, would be expected to be similar.

An input query is classified by the trained SVM to assign it to one of the clusters, and a transportation distance is used to suggest the closest matches from within the cluster.

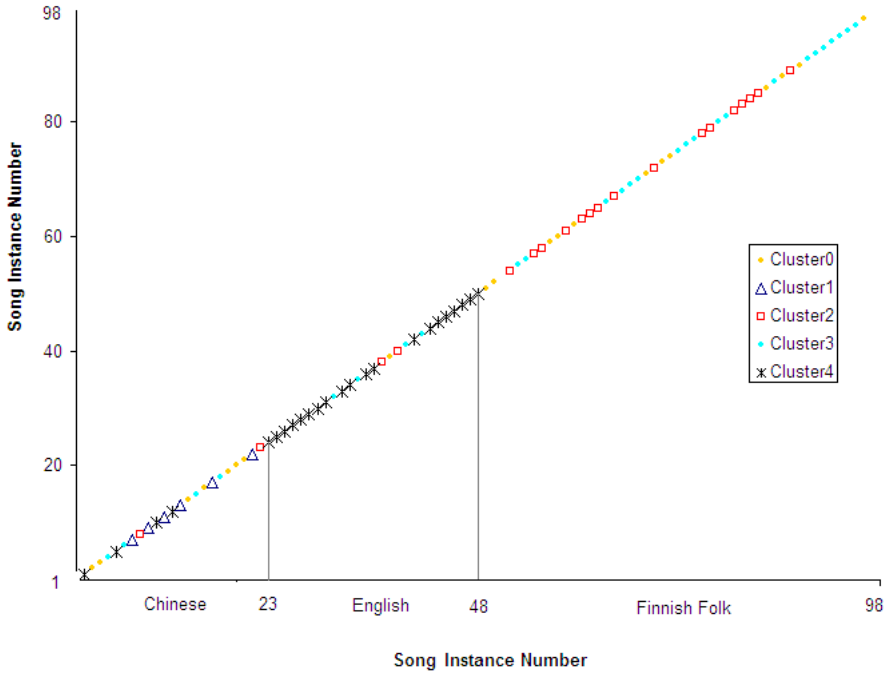


Illustration of clustering efficiency: 23 chinese, 25 English and 50 Finnish melodies.

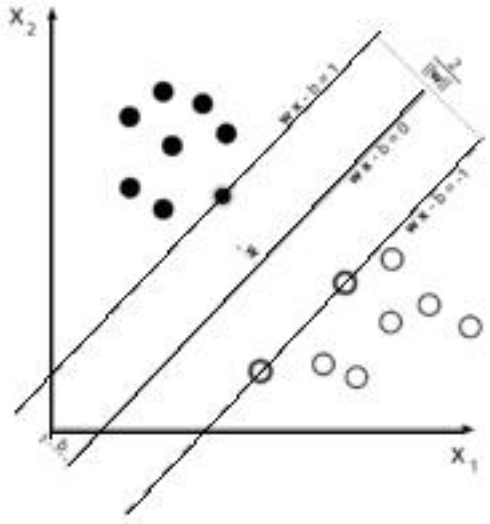
On a sample dataset containing 98 melodies, EM clustering lead to 5 clusters, as depicted in the figure above. Cluster 4 is seen to correspond almost exactly to English songs in the dataset, however Chinese Finnish songs dont have such a distinct correpondence to single clusters. Cluster 1, containing only Chinese melodies, was seen to correspond to to typically Chinese tunes. Cluster 2, on the other hand, consists of principally Finnish folk tunes.

Clusters 0 & 3 dont have a strong correlation to either class. A featurewise analysis of clusters is part of future work.

**Logan, Salomon et al proposed a novel method of extracting the global feature vector by clustering features(cepstral coefficients) of all frames of songs of 25.6 ms (sampled at), and storing specifications of clusters. As a line of enquiry, we simulated the same approach on windows of 10 notes overlapping on 2 notes each for a dataset of 48 songs, but preliminary K-means clustering (k=2), on the machine learning toolkit Weka, didn't yield encouraging results.*

4.5 Support Vector Machines

Support Vector Machines, proposed by Vladimir Vapnik, is a maximum margin classifier that minimizes an upper bound on its expected classification error. It attempts to find the hyperplane separating two classes of data that will generalize best to future data.



http://en.wikipedia.org/wiki/Support_vector_machine

For data points X, and binary classes, $y_i = \pm 1$; any hyperplane separating the two data classes has the form

$$y_i(w^T X_i + b) > 0 \ \forall i$$

The maximum margin hyperplane is defined by

$$w = \sum_{i=0}^{N} \alpha_i y_i X_i$$

Where b is set by the KKT conditions to maximize L_D

$$L_D = \sum_{i=0}^{N} \alpha_i - \frac{1}{2} \sum_{i=0}^{N} \sum_{j=0}^{N} \alpha_i \alpha_j y_i y_j X_i^T X_j$$

subject to the constraint

$$\sum_{i=0}^{N} \alpha_i y_i = 0 \ \alpha_i \geq 0 \ \forall i$$

5 Transport Distance Matching

The crux of our approach is a two-level utilization of data. The high level statistics get matched in the clustering phase for a preliminary pruning, whereas the temporal structure is not ignored, and gets matched in the detail in the pruning.

For search, the input melody is classified to one of the clusters, and the scores with the minimum transportation distances, EMD or LD, WITHIN the cluster are returned as similar songs.

5.1 Earth Mover Distance

Logan, Saloman et al and others have previously used Earth Mover's Distance as a metric for similiarity search for polyphones, using Cepstral coefficients. For monophonic pitch-matching, EMD has a still simpler formulation, where ground distances correspond to difference between notes, and the Frequency difference is to be transferred.

EMD is formalized as a LP problem. Let

$$P = \{(p_1, w_{p_1}), ..., (p_m, w_{p_m})\}$$

be the first signature with m 'heap's, where p_i is the heap representative and w_{pi} is the weight of the heap. Let

$$Q = \{(q_1, w_{q_1}), ..., (q_m, w_{q_m})\}$$

be the second signature with n clusters; and D be the ground distance matrix where d_{ij} is the ground distance between clusters p_i and q_j.

We want to find a flow $\mathbf{F}$, with f_{ij} the flow between p_i and q_j, that minimizes the overall cost

$$Work(P, Q, \mathbf{F}) = \sum_{\mathbf{i=1}}^{\mathbf{m}} \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{f_{ij} d_{ij}}$$

Subject to constraints:

$$f_{ij} \geq 0 \ 1 \leq i \leq m \ , \ 1 \leq j \leq n$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{p_i} \ 1 \leq i \leq m$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{q_i} \ 1 \leq j \leq n$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j})$$

The first constraint allows moving s̈upplies" from P to Q and not vice versa. The next two constraints limits the amount of supplies that can be sent by the heaps in P to their weights, and the heaps in Q to receive no more supplies than their weights; and the last constraint forces to move the maximum amount of supplies possible. This formulation has been extensively been used in image processing databases, and also for music.

5.2 Levenshtein distance

The Levenshtein distance (more popularly called the 'Edit distance') is a metric for measuring the amount of difference between two sequences. The Levenshtein distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. Levenshtein distance uses a simple incremental programming algorithm. It has previously been used in speech recognition, DNA analysis and plagiarism detection systems.

Eg.

kitten -> sitten (substitution of 's' for 'k') LD=1
sitten -> sittin (substitution of 'i' for 'e') LD=2
sittin -> sitting (insert 'g' at the end). LD=3

We use Levenshtein distance to measure distances between the input wave and all songs in the cluster to which it is assigned in the pruning step of our similarity search. Sequence matching has previously been tried for music retrieval by Ghias et al who used an approximate Baeza Yates algorithm. It has been suggested that change in note frequency can be a good estimator to compare two musical pieces.

Parson encoding: Pitch transitions are encoded as melodic contours. Eg. Midi note sequence [62, 63, 64, 64, 66, 65, 62] is coded according to the direction of pitch movement as [up up same up down down].

Implementation detail

A window of size 15 is used by for matching in case the input melody is long, else it is matched in whole. To allow for artistic variations, five LD distances are computed for N-grams of lengths n-3 to n+3 of the dataset melodies for each input case, and the minimum is taken.

A crucial domain specific detail includes attaching a very high replacement cost for the LD formulation, as a replacement of an 'up' symbol by a 'down' symbol would make the metric meaningless in terms of human perception. Insertion and deletion costs are roughly the same. Various combinations of weights were experimented with and the following chosen for the test:

Insertion cost	1
Deletion cost	1
Replacement cost	5

Through such a matching, temporal structure can be matched in addition to the preliminary classification according to cluster based on general characterstics of the musical piece. This new approach in this domain has shown promising results as described below, although the distance calculation is less accurate in some sense than the EMD.

However, the advantages are that it foolproof to poor playing, or occasional mistakes in the input through the Midi Keyboard.

6 Preliminary Results

The combined dataset of monophonies were converted into features selected in section 4.2. Expectation-maximization clustering was done ignoring datalabels.

Each input melody was classified into one of the 5 clusters so generated, and the **best three matches** by Edit Distance were reported.

Since a similarity search would have no objective evaluators of performance, 3 independent **human test subjects** were made to listen to similarity results for 5 **random queries**. The results were very encouraging, and on an average **2.1 out of 3** results were judged as similar to the Midi input.

	Result 1	Result 2	Result 3
Query 1	Similar	Similar	Similar
Query 2	Similar	Not Similar	Not Similar
Query 3	Similar	Similar	Similar
Query 4	Similar	Not Similar	Similar
Query 5	Similar	Similar	Similar

Sample Response for Test User 1

In case of inputs trying to play an existing file in the database, the file was reported **among the top 3 in 6 out of 10** cases.(though most players don't play very well) In 3 cases, the file is reported as the first hit.

References

[1] R. Typke, R. Veltkamp, and F. Wiering. Searching notated polyphonic music using transportation distances. In Proc. multimedia, pages 128-135 (2004).

[2] Oppenheim, A.V.: A speech analysis-synthesis system based on homomorphic filtering. Journal of the Acoustical Society of America 45, 458-465 (1969)

[3] Foote, J.T.: Content-based retrieval of music and audio. In: C.C.J.K. et al. (ed.) Proc. Storage and Retrieval for Image and Video Databases (SPIE), vol. 3229, pp. 138-147 (1997)

[4] Efficient repeating pattern finding in music databases - Hsu, Liu, et al.;Proceedings of the seventh international conference on Information and knowledge management (1998);Pages: 281 - 288;ISBN:1-58113-061-9

[5] Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines. Cambridge University Press, New York, NY (2000)

[6] Burgess, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2(2), 121-167 (1998)

[7] Logan, B., Salomon, A.: A music similarity function based on signal analysis. In: Proc. IEEE Intl. Conf. on Multimedia Expo., Tokyo, Japan (2001)

[8] Logan, B.: Mel frequency cepstral coefficients for music modelling. In: Proc. Intl. Conf. on Music Info. Retrieval (2000)

[9] Ghias, A., Logan, J., Chamberlin, D., and Smith, B. (1995).Query by humming musical information retrieval in an audio database. In Proc. ACM International Multimedia Conference.

[10] Mandel, Poliner, Ellis: Support Vector Machine Active Learning for Music Retrieval; Multimedia Systems, Vol. 12, No. 1. (2006), pp. 3-13

[11] Eerola, Tuomas and Toiviainen, Petri; University of Jyvaskyla: MIDI Toolbox: MATLAB Tools for Music Research: www.jyu.fi/musica/miditoolbox/ (2004)

[12] Weka3:Data Mining Software in Java; http://www.cs.waikato.ac.nz/ml/weka/

[13] Wikipedia, the free online encyclopedia; http://en.wikipedia.org/wiki/

[14] www.hitsongscience.com

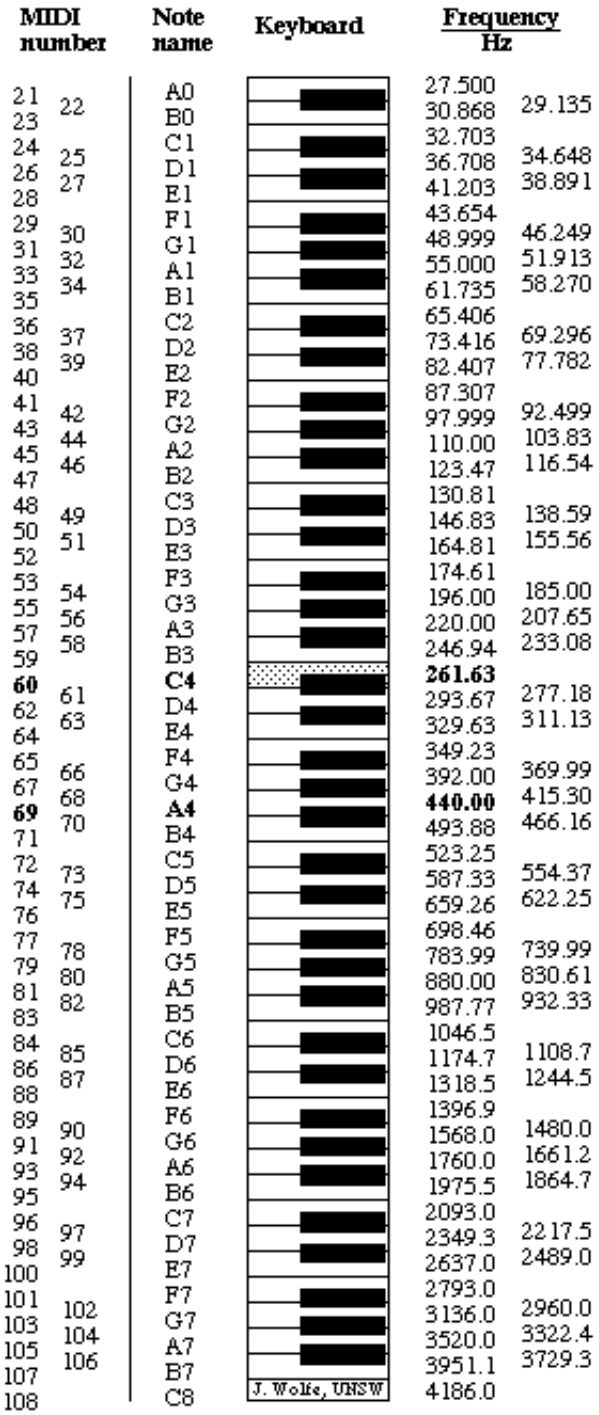
[15] http://www.cs.waikato.ac.nz/ ml/weka/

[16] www.jyu.fi/musica/miditoolbox/

[17] http://www.phys.unsw.edu.au/jw/notes.html

Appendix 1

Midi note mapping



Midi number to frequency Mapping on a Midi Keyboard (Base A4=440 Hz)
Source: www.phys.unsw.edu.au/jw/notes.html