Performance Evaluation of TreeQ and LVQ Classifiers for Music Information Retrieval

Matina Charami, Rami Halloush, Sofia Tsekeridou
Athens Information Technology (AIT)
0.8 km Markopoulo Ave.
GR - 19002 Peania, Athens, Greece
{scha, raha, sots}@ait.edu.gr

Abstract. Classification algorithms are gaining more and more importance in many fields such as Artificial Intelligence, Information Retrieval, Data Mining and Machine Vision. Many classification algorithms have emerged, belonging to different families, among which the tree-based and the clustering-based ones. Such extensive availability of classifiers makes the selection of the optimal one per case a rather complex task. In this paper, we aim to address this issue by conducting extensive experiments in a music information retrieval application, specifically with respect to music genre queries, in order to compare the performance of two state-of-the-art classifiers belonging to the formerly mentioned two classes of classification algorithms, namely, TreeQ and LVQ, respectively, using a variety of music features for such a task. The deployed performance metrics are extensive: accuracy, precision, recall, F-measure, confidence. Conclusions on the best performance of either classifier to support music genre queries are finally drawn.

1 Introduction

With the explosive amount of music data available on the Internet in recent years, there has been much interest in developing new ways to search and retrieve such data effectively. Most on-line music databases today, such as Napster and mp3.com, rely on file names or text labels to do searching and indexing, using traditional text searching techniques. Although this approach has proven to be useful and widely accepted in the past, there are many reasons this is not enough nowadays. As the amount of musical content increases and the Web becomes an important mechanism for distributing music, we expect to see a rising demand for music search services. It would be nice to have more sophisticated search capabilities, namely, searching by content.

Music Information Retrieval Systems can be classified into two types: i) systems that depend on human generated annotations and decisions (textual-based), and ii)

systems that depend on extracting information from the audio signal (content-based). The first type is manual and hence demands a lot of effort, it is time consuming and susceptible to errors. We will concentrate on content-based music information retrieval in the rest of the paper.

Content-based music information retrieval involves processes such as representative music feature extraction, classification in apriori known classes, usually deploying training and testing (supervised classification) and similarity-based querying. Thus, the challenging aspects of setting up an efficient music information retrieval are: i) what features to select as most representative on the types of similarity-based queries (e.g. music genre queries), ii) which classifiers will perform optimally for the application and types of queries at hand.

In this paper, we aim to address the issue of selecting the optimal combination of representative features and classifiers to address queries on music genre, by conducting extensive experiments in a music information retrieval application. The aim is to compare the performance of two state-of-the-art classifiers, namely, TreeQ [1] and LVQ [2] (Learning Vector Quantization), described briefly in the sequel, using a variety of music features for such a task. The deployed performance metrics are extensive: accuracy, precision, recall, F-measure, confidence. Conclusions on the best performance of either classifier combined with specific music feature vectors are finally drawn.

2 LVQ Classifier: a short overview

In general, LVQ [6] is a supervised version of vector quantization, which is applicable to pattern recognition, multi-class classification and data compression. LVQ algorithms directly define class boundaries based on prototypes, a nearest-neighbour rule and a winner-takes-it-all paradigm. The main idea, as shown in Figure 1, is to cover the input space of samples with 'codebook vectors' (CVs), each representing a region labeled with a class. A CV is localized in the centre of a decision region, called 'Voronoi cell', in the input space.

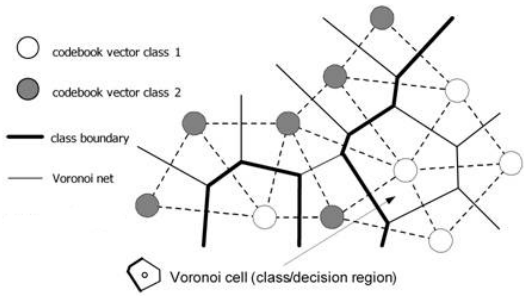

Fig. 1. LVQ space partitioning into decision regions by codebook vectors [2].

For the purpose of undertaken experiments, the LVQ software package of [7] has been used, which implements all algorithms necessary for statistical classification

and pattern recognition. The performance of LVQ depends on the algorithm implementation, as well as the data used for training and testing, in terms of size and the degree of representative features extracted from such data. The basic three implementations of LVQ are LVQ1, LVQ2 and LVQ3. In our experiments, we have used the optimized OLVQ1 implementation [1], an enhanced version of LVQ1.

The basic idea behind this algorithm is that each class is represented in terms of a set of codevectors m_i, each of which is a point in the D-Dimensional feature space. This set is called codebook. Several codebook vectors are assigned to each class. A feature vector x is then assigned to the same class to which the nearest m_i belongs:

$$m_c = \arg\min_i \{\| x - m_i \|\}, \text{ where } m_c \text{ is the nearest } m_i \text{ to } x$$

Values for m_i that approximately minimize the misclassification errors in the above nearest neighbour classification can be found as asymptotic values in the following learning process. Let $x(t)$ be a sample of input and let $m_i(t)$ represent sequences of m_i in the discrete time domain. Starting with properly defined initial values, the following equations define the basic LVQ1 process, where $0 < a(t) < 1$ and *t* is the iteration step:

- $m_c(t+1) = m_c(t) + a_c(t)[x(t) - m_c(t)]$, if x is classified correctly
- $m_c(t+1) = m_c(t) - a_c(t)[x(t) - m_c(t)]$, if x is classified incorrectly
- $m_i(t+1) = m_i(t)$, for $i \neq c$

OLVQ1 extends LVQ1 by modifying the latter so that an individual learning rate $a_i(t)$ is assigned to each m_i. Again, the discrete time learning process is given from above equations. For their fastest convergence, $a_i(t)$ is optimally determined by:

$$a_c(t) = \frac{a_c(t-1)}{1 + s(t)a_c(t-1)}$$

3 TreeQ Classifier: a short overview

For the purposes of the undertaken experiments, the TreeQ software package [3], [4], [5] has been considered, implementing the TreeQ machine learning algorithm. TreeQ is data-driven and therefore it can be used for any kind of data to find similarities by learning their differences. Especially for the case of audio data, the algorithm may be applied for speaker identification, speech and music classification, music and audio retrieval by similarity, audio segmentation.

Given labeled training data, the algorithm constructs templates that characterize these data, utilizing three main steps. First, it calculates spectral parameters for the audio data. Second, it grows a quantization tree from labeled parameterized data. This step learns those features that best characterize a class, i.e. given adequate training data, it learns the salient differences amongst classes and learns to ignore other insignificant differences. Third, the produced tree is used to construct the

templates. As soon as the templates are constructed (class models), similarities can be measured by calculating distances between them and test data.

The basic operation of the system is illustrated in Figure 2. A suitable corpus of audio examples must be accumulated and parameterized into feature vectors. The corpus must contain examples of the classes of audio to be discriminated. Next, a tree-based quantizer is constructed. This is a "supervised" operation and requires the training data to be labeled with a class. The tree automatically partitions the feature space into regions, called 'cells', which have maximally different class populations.

To generate an audio template, represented by a histogram, for subsequent retrieval, parameterized data are quantized using the tree. An audio file can be characterized by finding into which cells the input data vectors are most likely to fall. A template is an estimate of the vector counts for each cell, which captures the salient characteristics of the input audio, since sounds from different classes will have very different counts in the various histogram bins, while similar audio data should have similar counts. To retrieve audio by similarity, a histogram is further constructed for the query audio. The query histogram is compared to the corpus histograms, a similarity measure is calculated for each audio file in the corpus, and finally the query template is associated with a corpus template.

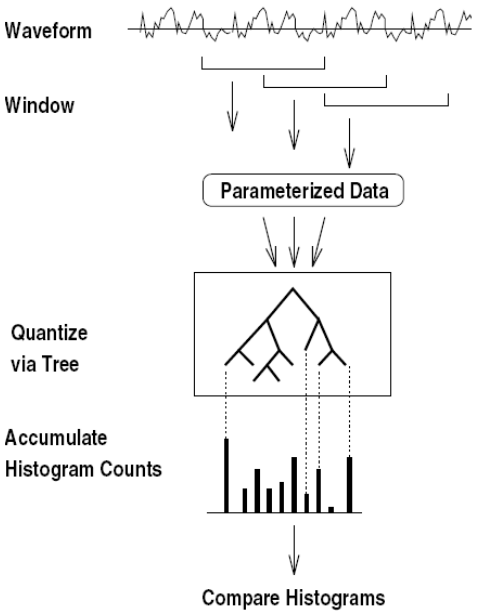

Fig. 2. An overview of the basic operations of the TreeQ algorithm [1].

4 Music Information Retrieval based on Genre Queries

It is a fact that the rapid development of technology continuously realizes scenarios that previously seemed science fiction. Web-based music stations like pandora.com give each user the opportunity to specify the kind of music he wants to listen to, and

in the context of ambient intelligence, pervasive systems will dress the surroundings with music based on the human's mood. Given the above and a number of other emerging applications of music, classification and information retrieval based on music genres becomes not only important, but essential and fundamental.

Our work addresses this problem and aims to provide extended experiments in order to push TreeQ and LVQ to their limits and decide which gives the best results under what contexts of use. Before presenting our experimental results, we considered it necessary to briefly describe the music features used to parameterize the audio data, i.e. the input to the previously presented classifiers.

4.1 Music Feature Extraction

Few classifiers directly operate on raw data such as pixels of an image or samples of speech waveforms. Most pattern recognition tasks are preceded by a pre-processing transformation that extracts invariant features from raw data, such as spectral components of acoustical signals. Thus, in our case, decisions need to be made on the types of representative features to be used by classifiers to achieve optimal music classification based on genres. Such feature extraction task parameterizes the raw music data into sequences of representative feature vectors.

It is evident that the selection of the adequate pre-processing method is equally vital as to the selection of the proper classifier for optimal performance, thus, it requires careful consideration. For this task, we have used the HTK Toolkit [8], which supports Hidden Markov Models (HMMs) using both Fast Fourier Transformation (FFT) and Linear Predictive Coding (LPC). The feature extraction process is controlled by a customizable configuration file that specifies all the conversion parameters towards extracting the desirable feature vectors.

In the current investigation, we have considered widely known and used features, namely the mel-frequency cepstral coefficients (MFCCs) and the linear prediction coding coefficients (LPCs). Both are, in general, the parameterisation of choice for many speech recognition applications, since they attain good discrimination capabilities and are flexible towards a number of manipulations.

In linear prediction analysis [9], the following transfer function is considered:

$$H(z) = \frac{1}{\sum_{i=0}^{p} a_i z^{-i}}$$

where the filter coefficients $\{a_i\}$ are chosen so as to minimize the mean square filter prediction error summed over the analysis window.

On the other hand, cepstral parameters are calculated from the log filter-bank amplitudes $\{m_j\}$ using the Discrete Cosine Transform (DCT), where N is the number of filter-bank channels [9]:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j-0.5)\right)$$

5 Experimental Setup and Performance Evaluation

In order to evaluate the performance of the two classifiers, namely, TreeQ and LVQ, in music genre classification and retrieval, we performed a set of experiments that are described in the sequel.

Initially, the experimental corpus has been created carefully. Five music genres have been considered, namely, jazz, reggae, pop, post rock and electro techno. For each genre, the corpus contains twenty music pieces, summing up to a total of one hundred pieces for all genres. Each music clip is about ten seconds long and has been selected as the most representative part of the entire music file. We have used the holdout sampling method in order to split the corpus into training and test data. Thus, seventy five of the music clips were used for training (the first fifteen of each genre) while the remaining twenty five were used for testing. All music pieces were later parameterized using the HTK toolkit, in MFCC and LPC feature vectors.

Experimentation on the classifiers (TreeQ, OLVQ1) performance followed using the extracted feature vectors in different combinations of features-classifiers. For the TreeQ experiments, we first obtained the optimal window size and the target rate of the algorithm. This was achieved by measuring the performance with varying values and combinations for these two parameters. An iterative procedure was used during which a new parameter was added, tested and decided to be maintained only if the acquired performance was no worse than the best performance achieved that far. This was done for both MFCC and LPC features with or without using quantization into histograms. Due to lack of time, the OLVQ1 experiments were performed using only MFCCs, without histogram quantization. LVQ involved only two parameters, n (number of codebook vectors) and k (kNN parameter), whose optimal values were obtained in a similar manner as the optimal window size and target rate for TreeQ. In all the experiments, the classifier performance was measured using Precision, Recall, Accuracy, F1 on precision and recall, and Confidence measures. Results are summarized in Table 1. We observe that TreeQ outperforms OLVQ1, when LPC features are used, with histogram quantization, for all considered measures.

Table 1. Optimal performance achieved by TreeQ and OLVQ1.

Performance Metric	*Classifier*		
	TreeQ		**LVQ**
	MFCC	LPC - Histograms	MFCC
Overall Accuracy(%)	**68.00**	**68.00**	52.00
Average Accuracy (%)	86.00	**87.00**	79.00
Average Precision (%)	71.13	**75.00**	58.75
Average Recall (%)	65.00	**70.00**	55.00
Average F-1 (%)	66.10	**69.99**	52.36

Having obtained the optimal setup for each classifier, we created learning curves as shown in Figures 3, 4 and 5. From these curves, we cannot extract valid conclusions about the learning capability of the two algorithms. For instance, for TreeQ with MFCCs, the curve is still increasing at the final steps which might indicate that a larger training set could help us achieve better performance. On the

other hand, at these final steps, the curve also seems to smoothen, so the improvement of the performance using more training data might be insignificant. Finally, comparing the two TreeQ curves with the OLVQ1 curve, we may say that TreeQ seems to better learn that OLVQ1. However, this is only an indication. More extensive experiments with a much larger dataset (and thus bigger test and training data sets) need to be undertaken to draw validated conclusions.

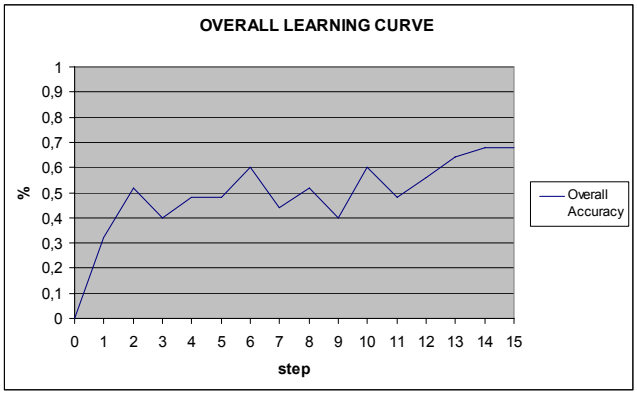

Fig. 3. Overall learning curve using MFCCs with TreeQ

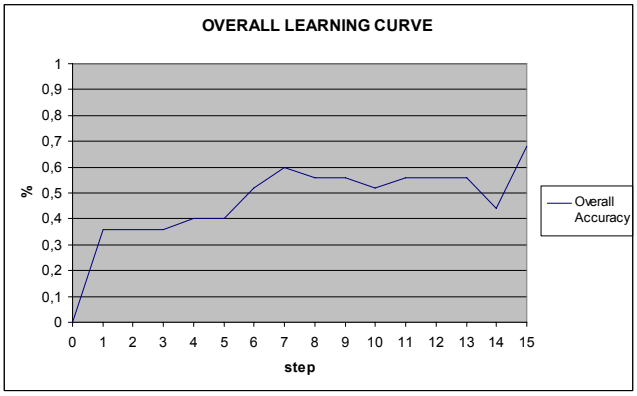

Fig. 4. Overall learning curve using LPCs with TreeQ

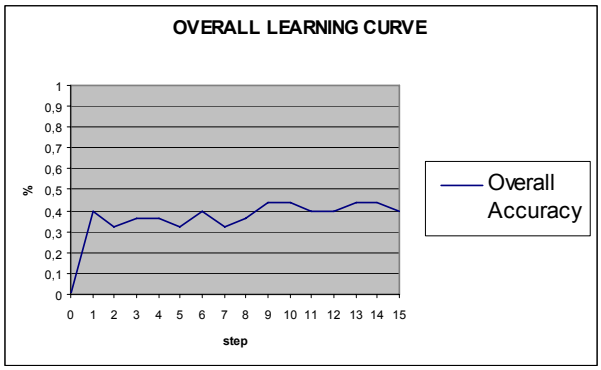

Fig. 5. Overall learning curve using MFCCs with LVQ

6 Conclusions and Future Work

In this paper, we have underlined the importance of content-based retrieval, which we addressed by conducting extensive experiments for music information retrieval, based on music genre queries, in order to compare the performance of two state-of-the-art classifiers, TreeQ and LVQ. From the performance evaluation, we could not reach valid conclusions, however, we have identified performance hints to extend the work further towards a certain direction. The learning capability of both algorithms needs to be further explored and hence we intend to undertake more experiments with a larger dataset as continuation of the currently reported work.

References

1. Jonathan T. Foote, TreeQ Manual V0.8, September, 2003
2. T. Kohonen, H. Hynninen, J. Kangas, H. Laaksonen, and K. Torkkola. LVQ-PAK: The learning vector quantization program package, Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland, 1996
3. Jonathan T. Foote, Content-based retrieval of music and audio, Multimedia Storage and Archiving Systems II, Proceedings of SPIE, 1997
4. Jonathan T. Foote, An overview of audio information retrieval, Multimedia Syst., Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999
5. Music retrieval demo using open-source software package TreeQ by Jonathan T. Foote, http://www.rotorbrain.com/foote/musicr/doc16.html
6. Forecasting with artificial neural networks, http://www.neural-forecasting.com
7. Helsinki University of Technology – Neural Networks Research Centre, http://www.cis.hut.fi/research/som_lvq_pak.shtml
8. The HTK Toolkit, http://htk.eng.cam.ac.uk/
9. Steve Young et al., *The HTK Book* (1995-1999 Microsoft Corporation, 2001-2006 Cambridge University Engineering Department)