Classification of music genres using sparse representations in overcomplete dictionaries ⋆

Cristian Rusu *

* *Faculty of Automatic Control and Computers, "Politehnica" University of Bucharest, Romania, e-mail: rusu.cristian.tgm@gmail.com*

Abstract: This paper presents a simple, but efficient and robust, method for music genre classification that utilizes sparse representations in overcomplete dictionaries. The training step involves creating dictionaries, using the K-SVD algorithm, in which data corresponding to a particular music genre has a sparse representation. In the classification step, the Orthogonal Matching Pursuit (OMP) algorithm is used to separate feature vectors that consist only of Linear Predictive Coding (LPC) coefficients. The paper analyses in detail a popular case study from the literature, the ISMIR 2004 database. Using the presented method, the correct classification percentage of the 6 music genres is 85.59%, result that is comparable with the best results published so far.

Keywords: music genre classification, linear predictive coding, sparse representations, orthogonal matching pursuit, K-SVD.

1. INTRODUCTION

As digital music becomes more and more popular, demand for new and old songs is increasing, so providers seek to create large music databases that would facilitate the user's access to this content. But, the organization of large music databases proves to be a real challenge. After overcoming the first task of actually storing the data, a natural question arises: is there a method to classify the songs in order to improve the user's experience when searching a large music repository?

The first idea that comes to mind, and the one that is the most natural, is to classify the songs by music genre. There are two main issues with this approach. First, there are no clear definitions for each music genre. Furthermore, there is still debate about the actual number of music genres. It is sufficient to say that the genres are not clearly delimited and that they differ from user to user. Second, the classification should use only the musical signal itself. Although most songs have some tag that describes the music genre, that tag could be wrong or empty. A robust classifier must take this into consideration in order for the end results to be relevant. Because of the need to correctly, quickly and automatically manage large sets of songs by using only the information from the audio signal itself the problem of music genre classification is considered a difficult one and it has received much attention in the past years. See Li et al. (2003).

⋆ The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/88/1.5/S/61178.

Since the audio signal contains data revealing the music genre (Scaringella et al. (2006) and Sukittanon et al. (2004)), like in every classification problem, the first step is to identify the features that will be extracted from the audio signal and used in the separation process. Because the audio signal varies a lot with time, the features are computed for short chunks of the audio signal called frames or windows. Some studies use only a single feature and others extract from the signal multiple features (this is the so-called bag-of-features approach). The most popular features used in recent studies are: frequency and spectral envelope information, timbre texture, rhythmic and pitch content. These features can be used separately or together in any combination. Other, simpler, features like the length and the energy of the audio signal can also be used to improve the feature set. Usually, all the selected features are concatenated in a feature vector. This study considers the Linear Predictive Coding (LPC) coefficients and, separately, the Mel-frequency cepstral (MFC) coefficients. See Rabiner et al. (1993) for further details.

After the features are selected, a classification method that clearly separates the features belonging to different music genres is required. In this paper, in order to find a good classification method, tools from the field of sparse representations are introduced. Recent advances in this field lead to the development of new classification methods. The field deals with two basic problems. The first: given an overcomplete dictionary $A \in \mathbb{R}^{m \times n}$ with $m \leq n$ and a vector $b \in \mathbb{R}^m$, find the minimum (or a small fixed) number of columns from A such that, by linear combination, they give x or a good approximation of it; and the second problem is the reverse: given a set of training vectors $\{y_i\}_{i=1}^N \in \mathbb{R}^m$ find an overcomplete dictionary $A \in \mathbb{R}^{m \times n}$

with $m \leq n$ (usually $m \ll n$) such that each vector in the training set has a sparse representation in A. Overcomplete dictionary means in this setting just a matrix with more (or much more) columns than rows. There are many solutions given for the first problem, algorithms like: Matching Pursuit (MP) (Mallat et al. (1993)), Orthogonal Matching Pursuit (OMP) (Temlyakov (1999)) and Basis Pursuit (BP) (Chen et al. (1998)). Of course, the interest is mainly in the solutions to the second problem. The algorithm that is used in this study and that gives a good approximation of the solution in the classification procedure is called K-SVD (Aharon et al. (2006)).

This paper describes an efficient method for genre classification using sparse representations. After conducting extensive simulations, the best results were obtained using the LPC coefficients in the feature extraction step and the K-SVD algorithm in the overcomplete dictionary learning phase. The results obtained are very high (85.59% correct classification), demonstrating the efficiency and the robustness of the proposed method. In Section 2, some basic theoretical notions and algorithms regarding sparse representation are presented, followed by feature extraction algorithms. In Section 3 details are presented regarding the way algorithms from the sparse representation field are used in a classification problem context. In the end, the results obtained by applying the method on the ISMIR 2004 database are described in detail and a comparison with other studies on the same database is provided.

2. THEORETICAL FOUNDATIONS

2.1 Sparse representations

Consider a matrix $A \in \mathbb{R}^{m \times n}$ with $m < n$ and full rank m. In this setting A is called overcomplete and, usually, n is much greater than m. Define the underdetermined system of linear equations $Ax = b$ with $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. In general, this system has, because of the structure of the matrix A, an infinite number of solutions. In order to obtain a single, well defined, solution, an extra minimization/maximization criterion is added. So, an objective function is introduced that minimizes a criterion among all the solutions of the system. The following optimization problem is defined

$$(P_j): \quad \underset{x}{\text{minimize}}\ J(x) \text{ subject to } Ax = b \tag{1}$$

Choosing an objective function $J(x)$ strictly convex guarantees a unique solution to the underdetermined system, solution that can be found using efficient methods. Most of the time, the objective function is represented by the square of the Euclidian norm $||x||_2^2$. This problem is labeled as (P_2). The explicit solution to this problem is

$$x = A^+ b = (A^T A)^{-1} A^T b \tag{2}$$

This paper considers a different objective function, the sparsity of the solution x. An intuitive measure of the sparsity of x is the number of non-zero elements it has. A vector is called sparse if it has only a few (relative to its size) non-zero elements in its composition. Thus, the l_0 pseudo-norm of a vector x is defined to be

$$||x||_0 = card(x) = \#\{i \mid x_i \neq 0\} \tag{3}$$

Notice that $||x||_0$ is a pseudo-norm and not a norm because it is not positive homogeneous ($||\alpha x||_0 \neq |\alpha| ||x||_0\ \forall \alpha \in \mathbb{R}$). Analogously, the (P_0) problem is considered by setting $J(x) = ||x||_0$. The optimization problem for underdetermined systems is

$$(P_0): \quad \underset{x}{\text{minimize}}\ ||x||_0 \text{ subject to } Ax = b \tag{4}$$

Although problems (P_2) and (P_0) look very similar they are fundamentally different. The solution to problem (P_2) is unique and is given by (2). The (P_0) problem is much more difficult and it raises a lot of issues when trying to solve it. This is because of the discrete, discontinuous and non-convex nature of the l_0 pseudo-norm.

In the general case, it has been shown that (P_0) is a combinatorial search problem: for a given A all the subsystems $A_S x_S = b$ need to be generate, where A_S represents the matrix obtained by picking the columns of A that are in the set S and check if they can be solved. Evidently, this approach is not practical in real world applications since there are too many such subsystems to check. Thus, the combinatorial optimization problem (P_0) is relaxed to a convex optimization problem that can be solved in a reasonable amount of time. As an alternative, heuristic search methods are used or greedy algorithms are applied.

Most of the time, the (P_0) problem is relaxed to the (P_1) optimization problem that is

$$(P_1): \quad \underset{x}{\text{minimize}}\ ||x||_1 \text{ subject to } Ax = b \tag{5}$$

where $||x||_1 = \sum_i |x_i|$ is the l_1 norm of the vector x. This is a convex optimization problem and it can be solved efficiently with linear programming (LP) solvers. In this context, of further interest are the conditions that need to be imposed so that the solution of the original problem (P_0) and the solution of the relaxed problem (P_1) are the same.

The search techniques deployed in solving problem (P_0) are usually greedy methods. These methods try to directly solve the original (P_0) problem, unlike the (P_1) approach. Since these types of problems were the first studied, the theoretical and experimental support is vastly superior.

This paper considers the greedy algorithm approach.

In a greedy strategy, the exhaustive search of the global solution is abandoned in favor of an iterative process that improves the current solution. The algorithm starts from a solution $x^0 = 0$ and it constructs a solution at step k by maintaining a set of active column indices from the dictionary (set which is initially empty) that grows with one atom at each iteration (an atom represents one column from the dictionary). Each step selects an atom index, that is not in the active set, such that it minimizes the l_2 norm of the difference between the current residual r^k and the atom itself. After an atom index is selected, the l_2 residual is recalculated and the algorithm continues by selecting a new atom index. Next, a standard approach to solving the (P_0) problem by a greedy strategy, the Orthogonal Matching Pursuit (OMP) algorithm is described.

OMP Algorithm (Given: the dictionary $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$. Return: $x^k \in \mathbb{R}^n$ the approximate solution of $Ax = b$ with k_0 non-zero elements.)

(1) **Initialization**
   (a) initial solution $x^0 = 0$

(b) initial residual $r^0 = b - Ax^0 = b$
(c) set S of active atom indices $S^0 = \emptyset$

(2) **Iteration** : $k = 1, \ldots, k_0$
(a) compute $\epsilon(j) = \min_{z_j} ||a_j z_j - r^{k-1}||_2^2$ for $j = 1, \ldots, n,\ j \notin S^{k-1}$; a_j is the jth column of matrix A
(b) find $j_0 = \operatorname{argmin}_j \epsilon(j)$ and update $S^k = S^{k-1} \cup \{j_0\}$
(c) compute $x^k = \operatorname{argmin}_x ||Ax - b||_2^2$ on S^k
(d) update residual $r^k = b - Ax^k$

The solution has k_0 non-zero elements and the algorithm has complexity $O(k_0 mn)$. Evidently, the complexity of this algorithm is much less than the complexity of a full exhaustive search which is $O(mn^{k_0} k_0^2)$. The problem is that the greedy strategy does not guarantee that the solution found is the global minimum, whereas the exhaustive search guarantees to find the optimum solution.

2.2 Dictionary training

A fundamental question that arises in the study of such problems is the way in which the dictionary A is chosen. One way of obtaining the dictionary is to create it as a result of a training process applied to a set of input data. Consider a set of available vectors $\{y_i\}_{i=1}^N \in \mathbb{R}^m$ generated by an unknown linear system. We note that $m \ll N$ in order to have a representative collection of vectors. The goal is to estimate the matrix A, the upper limit of the $||x||_0$ pseudo-norm (k_0) and the error (ϵ) such that the training vectors have a sparse representation in A.

First, consider ϵ known and try to estimate A. This leads to the following optimization problem

$$\underset{A}{\text{minimize}} \sum_{i=1}^{N} ||x_i||_0 \text{ subject to } ||y_i - Ax_i||_2 < \epsilon,\ 1 \le i \le N \tag{6}$$

If this problem could be solved directly, the resulting matrix A would have the sparsest representation possible of all the N training vectors.

Now, consider the problem where A and k_0 (the maximum value of the $||x||_0$ pseudo-norm) are known and try to estimate the vector x such that the reconstruction error is minimized. The resulting optimization problem is

$$\underset{x_i}{\text{minimize}} \sum_{i=1}^{N} ||y_i - Ax_i||_2^2 \text{ subject to } ||x_i||_0 < k_0,\ 1 \le i \le N \tag{7}$$

This problem can be formulated in the context of matrix factorization algorithms. Since the matrix $Y \in \mathbb{R}^{m \times N}$ is available (created by joining, columnwise, the training vectors $\{y_i\}_{i=1}^N$) the problem is to write $Y = AX$ where $X \in \mathbb{R}^{n \times N}$ is a sparse matrix of representations and A is the dictionary.

Note that the dictionary must have a fixed size $A \in \mathbb{R}^{m \times n}$. This restriction is imposed because the goal is to create dictionaries A that represent the data as well as possible with far fewer atoms in the dictionary than available training vectors ($n \ll N$) in order to keep the computational and memory loads to a relatively moderate level. Otherwise, if we accept for example $n = N$, a trivial solution to the problem is to set $A = Y$ and X to be the identity matrix I_N. This solution is unacceptable since it leads to very slow manipulations with the dictionary A.

Coming back to our original approach, consider (6) and (7) as a single optimization problem with an alternating objective. Starting with an initial dictionary A_0 compute the sparse representations matrix X_0 for our training data using OMP and then compute A_1 such that the representations are even sparser. Then, iteratively repeat this process K times.

For the computation of the sparse representations in X the OMP algorithm is used and for the computation of A there are two choices: use the least squares method (MOD algorithm) or use the K-SVD algorithm.

The update formulae for the MOD method is

$$A_k = \operatorname{argmin}_A ||Y - AX_k||_F^2 = YX_k^T (X_k X_k^T)^{-1} \tag{8}$$

The update formulae for the K-SVD algorithm is

$$\begin{aligned} \operatorname{argmin}_A ||Y - AX_k||_F^2 &= \operatorname{argmin}_A ||Y - \sum_{j=1}^{n} a_j x_j^T||_F^2 \\ &= \operatorname{argmin}_{a_{j_0},\, x_{j_0}^T} ||(Y - \sum_{j \neq j_0} a_j x_j^T) - a_{j_0} x_{j_0}^T)||_F^2,\ \forall\ j_0 \end{aligned} \tag{9}$$

In both cases $||\cdot||_F$ is the Frobenius norm of the matrix $Y - AX_k$. As it can be seen from (8), MOD updates the whole dictionary A directly and does not change in any way X, while in (9) K-SVD performs an optimization step for each column of A separately modifying in the process also the sparse representations from X. Denote $E_{j_0} = (Y - \sum_{j \neq j_0} a_j x_j^T)$. In order to minimize expression (9) the optimum values of a_{j_0} and $x_{j_0}^T$ are needed. Notice that this is a rank 1 update (row-column vector product). So, the problem becomes one of finding a_{j_0} and $x_{j_0}^T$ such that their outer product is the best rank 1 approximation of E_{j_0}. The two vectors are obtained by applying the Singular Value Decomposition (SVD) on the matrix $E_{j_0} = U \Delta V^T$ and keeping the vectors associated with the largest singular value. At every step update both a_{j_0} and $x_{j_0}^T$. Since the SVD does not compute sparse vectors, the decomposition is applied only on the set of columns of E_{j_0} that utilize the atom a_{j0}. The rest of the elements are considered 0.

The general structure of the MOD/K-SVD algorithm is presented next.

MOD/K-SVD Algorithm (Given: Training vectors $\{y_i\}_{i=1}^N \in \mathbb{R}^m$, upper bound of the $||x||_0$ pseudo-norm k_0 and the error ϵ. Return: the trained dictionary $A \in \mathbb{R}^{m \times n}$ and $X \in \mathbb{R}^{n \times N}$ containing the sparse representations of the training vectors.)

(1) **Initialization**
(a) initialize the dictionary A_0 randomly or by using n training vectors
(b) normalize A_0's columns

(2) **Iteration**
(a) find a sparse representation for each training vector

$$x_i = \operatorname{argmin}_x ||y_i - A_{k-1} x||_2^2 \text{ subject to } ||x||_0 \le k_0\ \forall i \tag{10}$$

Join columnwise the vectors x_i to form the matrix X_k.

(b) update the dictionary A using MOD or K-SVD

(i) MOD:

$$A_k = \underset{A}{\operatorname{argmin}} ||Y - AX_k||_F^2 = YX_k^T(X_kX_k^T)^{-1} \tag{11}$$

(ii) K-SVD: for $j_0 = 1, 2, \ldots, n$

(A) select the training vectors which use the atom a_{j_0}

$$\Omega_{j_0} = \{i |\ 1 \leq i \leq N, X_k(j_0, i) \neq 0\} \tag{12}$$

(B) compute E_{j_0}

$$E_{j_0} = \left(Y - \sum_{j \neq j_0} a_j x_j^T\right) \tag{13}$$

(C) extract from E_{j_0} the columns from set în Ω_{j_0} obtaining $E_{j_0}^R$

(D) apply one step of the SVD algorithm for $E_{j_0}^R = U\Delta V^T$ and update: $a_{j_0} = u_1$ and $x_{j_0}^R = \Delta[1,1]v_1$

(c) check stop condition $||Y - A_kX_k||_F^2 \leq \epsilon$. Because of the size of the dictionary and the optimization procedures, this condition may never be satisfied. So the algorithm is also restricted to run a maximum number of iterations.

Evidently, because the SVD provides the best rank 1 approximation, the speed of convergence of K-SVD is higher than that of MOD but the price paid is a larger running time for K-SVD. Improvements in the running time of K-SVD can be made, for example, by approximating the SVD step using the power method (Journee et al. (2010)). All results presented in this paper were obtained using only the K-SVD method since the training step in this case is not time constrained. Experiments were also conducted using the MOD algorithm but the results obtained are far below those obtained with K-SVD and therefore will not be described here. The reader should notice that the original problem of training a dictionary for sparse representations is not a convex optimization problem. Both MOD and K-SVD provide an approximation of the solution and neither can gurantee to reach the global minimum.

For details and further references regarding the sparse representations topics the reader is advised to consult Bruckstein et al. (2009).

2.3 Audio features extraction

This paper considers two of the most popular feature extraction methods used for audio signals: Linear Predictive Coding (LPC) and Mel-frequency cepstral (MFC) coefficients.

First, the features of the audio signal are extracted using the Linear Predictive Coding tool. This is a very popular and powerful tool in the field of audio signal processing and it is used mostly for representing the spectral envelope of a digital audio signal. The result is comprised of the set of LPC coefficients for each defined frame of the signal. LPC uses an AR model and it assumes that an element of the discrete audio signal $s(t)$ can be estimated by a linear combination of the previous M elements. N is the length of the signal $s(t)$. For details regarding the types of identification models that can be deployed in such applications the reader should consult L. Ljung (1999). Therefore, the approximation is

$$\tilde{s}(t) = d_1s(t-1) + d_2s(t-2) + \cdots + d_Ms(t-M) = \sum_{i=1}^{M} d_is(t-i) \tag{14}$$

where the coefficients $d_i \in \mathbb{R}$ are called the coefficients of the linear predictor of order M.
The model error (14) is expressed as

$$\epsilon(t) = s(t) - \tilde{s}(t) = s(t) - \sum_{i=1}^{M} d_is(t-i) \tag{15}$$

Coefficients d_i minimize the square of the error defined in (15) and they are the solution of the linear system

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(M-2) & r(M-1) \\ r(1) & r(0) & \cdots & r(M-3) & r(M-2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r(M-2) & r(M-3) & \cdots & r(0) & r(1) \\ r(M-1) & r(M-2) & \cdots & r(1) & r(0) \end{bmatrix} \times \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{M-1} \\ d_M \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(M-1) \\ r(M) \end{bmatrix} \tag{16}$$

written compactly $Rd = r$ where

$$r(k) = \frac{1}{N-k} \sum_{t=0}^{N-1-k} s(t)s(t+k) \tag{17}$$

Solving (16) as a general linear system takes $O(M^3)$ operations but, in this case, taking into account the symmetric Toeplitz structure of the matrix R it takes only $O(M^2)$ operations by using the Levinson-Durbin algorithm.

The MFC coefficients are used to represent the short-term power spectrum of a discrete audio signal $s(t)$ with $0 \leq t \leq N-1$. This tool is a very popular one because it allows for a better representation of sounds by constructing a good approximation of the way in which humans process audio signals. In order to compute the coefficients the following steps are executed:

(1) Define the Discrete Fourier Transform (DFT) of $s(t)$ as $\hat{s}(t) = DFT\{s(t)\}$ and then the real cepstrum of $s(t)$ is defined to be

$$c(m) = DFT^{-1}\{\log|\hat{s}(t)|\} = \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} \log|\hat{s}(t)| e^{2i\pi tm/N} \tag{18}$$

where $i = \sqrt{-1}$ and $l = 0, 1, \ldots, N/2 + 1$. For the above definition to work the Fourier transform of $s(t)$ must be non-zero everywhere.

(2) Apply now a Mel-frequency scaling so that we introduce the energy in frequency bands that are exponentially-spaced. The mapping is defined by

$$f(\phi) = 700(e^{\phi/1125} - 1),\ \phi \geq 0 \tag{19}$$

In this fashion, $\hat{h}_p$ with $p = 1, \ldots, P$ triangular filters with equal area and center frequency the linearly

spaced Mels are constructed. The triangular filters are $P = 50$ in number and span the bandwidth [0 11000] Hz. Each filter is given by

$$\hat{h}_p(t) = \begin{cases} 0, \ 0 \le tF_s/N < f_c(p-1) \\ a_p \dfrac{tF_s/N - f_c(p-1)}{f_c(p) - f_c(p-1)}, \\ \quad f_c(p-1) \le tF_s/N < f_c(p) \\ a_p \dfrac{tF_s/N - f_c(p+1)}{f_c(p) - f_c(p+1)}, \\ \quad f_c(p) \le tF_s/N < f_c(p-1) \\ 0, \ f_c(p+1) \le tF_s/N \le F_s \end{cases} \tag{20}$$

where F_s is the Nyquist frequency, f_c are the centers of each frequency band and the band dependent magnitude factors a_p are given by

$$a_p = \begin{cases} 0.0015, & 1 \le p \le 14 \\ \dfrac{2}{f_c(p+1) - f_c(p-1)}, & 15 \le p \le 50 \end{cases} \tag{21}$$

This warping is used because it closely approximates the response of the human auditory system.

(3) The MFC coefficients are given by

$$cc(m) = \beta_P(m) \sum_{p=1}^{P} \log \left(\sum_{t=0}^{N-1} |\hat{s}(t)\hat{h}_p(t)| \right) \times \cos \left[\frac{m\pi}{P} \left(p - \frac{1}{2}\right) \right] \tag{22}$$

where $0 \le m \le P$ and the scaling factor

$$\beta_P(m) = \begin{cases} \dfrac{1}{\sqrt{P}}, & m = 0 \\ \sqrt{\dfrac{2}{P}}, & m > 0 \end{cases} \tag{23}$$

Additionally, one can compute the short time MFC coefficients by

$$cc(m,j) = \beta_P(m) \sum_{p=1}^{P} \left(\sum_{t=0}^{J-1} \log |\hat{s}(t,j)\hat{h}_p(t)| \right) \times \cos \left(\frac{m\pi}{P} \left(p - \frac{1}{2}\right) \right) \tag{24}$$

where $0 \le m \le P$ and where $\hat{s}(t,j)$ is the DFT of length J of $s(t)$ localized over a small time region.

(4) Finally, compute the first and second derivatives of the MFC coefficients using

$$\Delta(m,j) = cc(m,j) - cc(m,j-1) \tag{25}$$

$$\Delta\Delta(m,j) = \Delta(m,j) - \Delta(m,j-1) \tag{26}$$

These two features describe the rate of change of the coefficients from one window of the audio signal to the next.

Usually in speech processing applications, only the first 13 coefficients are kept and in audio signal processing applications the first 20 coefficients. In both cases, the first coefficient ($m = 0$) is discarded because it contains only information regarding the energy of the signal.

Experiments show that for the classification of music genres LPC coefficients perform much better than MFC coefficients. Because of this, the sections that are presented next will contain only results using the LPC coefficients.

Another important observation is the fact that both LPC and MFC coefficients are local features in the audio signal. In both cases, the original audio signal is split up into windows and the coefficients are computed for each window separately (in some situations the windows could actually be overlapping). Another approach to the problem of feature selection from an audio signal is to extract from the whole signal (or from relatively big chunks of it) some global parameters that best describe the signal. Features that fall into this category are based on the MPEG-7 standard and usually include among others: temporal centroid, spectral centroid information, audio spectrum centroid, spread and envelope values for different frequency bands, spectral flatness measure etc.

All conducted experiments considered only one type of features, either LPC or MFC coefficients. Another possible strategy in the feature selection stage is the so-called bag-of-features approach. In this setting, several types of features are extracted in order to improve the performance of the classifier. For example, one might use both MFC and LPC coefficients in the hope that both will better represent the audio signals for classification. The problem that arises in this case is deciding which combination of features from the total available bag leads to the best results. This approach is detailed in Scaringella et al. (2006).

For a detailed and complete analysis of the most important methods of features extraction from digital audio signals, the reader should consult Rabiner et al. (1993). Also, H. G. Kim et al. (2005) contains a detailed discussion on the MPEG-7 standard.

3. CLASSIFICATION OF MUSIC GENRES USING SPARSE REPRESENTATIONS

Now that the main theoretical notions were presented, a description of how sparse representations are used to solve the classification problem is next. The main idea is the following: after extracting the feature vectors from the available songs, use the K-SVD algorithm to train a dictionary of length L in $D_g \in \mathbb{R}^{M \times L}$ for each music genre $g = 1, \ldots, G$, separately using only the feature vectors belonging to that particular genre. G represents the number of music genres considered. At the end of the training phase there will be G dictionaries each capable of constructing sparse representation of a particular music genre. The G dictionaries make up the classifier. Now, when a test song is presented to the classifier, the following steps occur. First, feature vectors are extracted from the presented song by the same method that was used in the training process and saved columnwise in the matrix $F^{\text{test}} \in \mathbb{R}^{M \times q}$ where M is the length of a feature vector and q is the number of feature vectors extracted. Now that F^{test} was created, sparse representation are computed for each feature vector in each available dictionary by letting the OMP algorithm run just a few iterations (in this paper only 5 iterations were considered). That means that the OMP algorithm is applied qG times and the results are saved in the matrices $X_g^{\text{test}} \in \mathbb{R}^{L \times q}$ (again, separately for each music genre g). Thus, the error matrices for each music genre $E_g = F^{\text{test}} - D_g X_g^{\text{test}}$ are defined, where $E_g \in \mathbb{R}^{M \times q}$. The test song is attributed to the genre having the lowest Frobenius norm of the error matrix. The Frobenius norm of the matrix E_g is $||E_g||_F = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{q} e_{ij}^2}$. Only a few iterations of OMP

are used because experiments show that in the proper dictionary a low residual magnitude is reached very quickly when trying to represent a new vector. A large number of iterations would generate low residuals magnitudes in each dictionary making it much harder to separate the correct dictionary from the rest.

Intuitively, the algorithm works like this: for each classification group a dictionary D_g is created such that members of that group have a sparse representation, which means that the atoms in the dictionary extract the most important characteristics of the group; the assumption that the classifier makes is that the members of a group i with dictionary D_i do not have a good (low residual) sparse approximation in the other dictionaries D_j, $i \neq j$. So, a test input is labeled as belonging to class i if it has the lowest residual magnitude of its sparse representation in the trained dictionary D_i. Next, follows a detailed description of the training and classification phases.

3.1 Training

For the training process all the available mp3 songs in the ISMIR 2004 training database are used. The following training steps are applied

- convert the .mp3 files to .wav for easier access
- for each wav file
  - read the full content of the files and cut the first 10 seconds of the audio signal in order to eliminate the intro sounds.
  - all songs are recorded at 44 kHz so they are decimated with the factor 4 to obtain a rate of 11 kHz. This step is necessary in order to reduce the amount of data to be processed. In any case, for the separation of music genre, high quality audio signals are not really needed since details of the signal are of no interest. In order to speed up the running time, for the rest of the processing steps only o portion of the original signal is kept. Specifically, only about 30s (after the initial 10s intro) of the audio signal is saved.
  - the decimated signal is partitioned in windows of length $T = 3000$ samples (about 250 ms) with overlap $K = 750$ samples (about 60 ms). The values for T and K were chosen taking into account the recommendations presented in other articles and studies. The high value of T was chosen in order to capture a long term characteristic in the signal and the value of K was chosen to be a forth of that of T as studies suggest this to be a good choice. Of course, different values were also tested but it seems that this combination works best for your case.
  - for each separated window apply the Levinson-Durbin algorithm to compute its LPC coefficients of dimension $M = 120$. The computation yields a large number of coefficients because of the large window length T.
  - assuming N vectors with LPC coefficients are computed, they are concatenated columnwise in the matrix $Y \in \mathbb{R}^{M \times N}$. Again, in order to improve the speed of the algorithm some of the N feature vectors from the matrix Y could be dropped. This is especially useful when the processed song is lengthy. In this article the full feature vectors are kept.
- apply the K-SVD algorithm on the obtained matrix Y with the restriction $D \in \mathbb{R}^{M \times 3M}$ to obtain $Y \approx D \times X$ where the matrix X is sparse. The size of the matrix D was chosen large enough such that the dictionary is overcomplete but not large enough to slow down the algorithm too much. Different dimensions were checked and tests show that dictionaries three times overcomplete suffice to obtain good results in a reasonable amount of computing time.

At the end of the training process the dictionaries D_g, $g = 1, \ldots, G$ are available for each music genre separately. This paper considers the number of music genres $G = 6$: classical, electronic, jazz-blues, metal, rock-pop and world. The number of songs in each category varies from 26 for jazz-blues to 320 for classical. The world genre contains songs that do not belong in any of the previous defined genres.
The training algorithm needs an overnight run to finish constructing the dictionaries for all 6 genres.

3.2 Classification

Now that the training phase is over the actual classification phase begins. The set of test songs to be classified is completely different from the one that was used in the training phase. For each music genre, the number of test songs equals the number of training songs. The classification steps are the following

- convert the .mp3 files to .wav for easier access
- for each wav file
  - apply the same processing algorithm as in the training phase: decimate with factor 4, split the signals into windows with $T = 3000$ and $K = 750$, compute the LPC coefficients for each of the N windows and construct $Y \in \mathbb{R}^{M \times N}$ by concatenating columnwise the feature vectors. It is important to stress that the audio signals need to be processed in the same manner in which they were processed in the training phase.
  - compute the sparse representations matrix X_g in the dictionary D_g for $g = 1, \ldots, G$ by applying the OMP algorithm N times, for each column of Y.
  - compute the matrix $E_g = Y - D_g X_g$, for $g = 1, \ldots, G$ of remaining residuals.
  - classify the test song as belonging to the cluster indexed i, where $i = \underset{g}{\operatorname{argmin}} ||E_g||_F$ with $g = 1, \ldots, G$, and $||\cdot||_F$ is the Frobenius norm.

The total number of test songs fed to the classifier is 729, summing up to more than 49 hours of audio playback. The number of test songs, clustered by their correct genre, is: 320 for classical, 115 for electronic, 26 for jazz-blues (j-b), 45 for metal, 101 for rock-pop (r-p) and 122 for world.
The running time of the current classification algorithm is a little over 2 hours for all the 729 songs, on an Intel i3 2.13GHz processor with 4GB of RAM.

3.3 Results

For convenience, the results obtained are centralized in the result matrix R

	classical	electronic	j-b	metal	r-p	world
classical	316	0	0	0	0	4
electronic	11	86	2	0	11	5
j-b	2	0	22	0	0	2
metal	3	0	0	27	15	0
r-p	5	4	0	1	90	1
world	20	9	0	0	10	83

(27)

where element $r = R(i, j)$ reads as: r song of genre indexed i have been clustered as belonging to genre indexed j.

The percentage of correct classification, per genre, is

$$\begin{aligned} \text{correct}_{\text{classical}} &= 98.75\% \\ \text{correct}_{\text{electronic}} &= 74.78\% \\ \text{correct}_{\text{jazz-blues}} &= 84.61\% \\ \text{correct}_{\text{metal}} &= 60.00\% \\ \text{correct}_{\text{rock-pop}} &= 89.10\% \\ \text{correct}_{\text{world}} &= 68.03\% \end{aligned} \tag{28}$$

The genres that have the lowest correct classification percentage are world and metal. These categories are mistaken most often with genres classical and rock-pop. In the case of the world genre, the relatively bad result can be explained by the fact that the group is a mixture of many types of songs with different characteristics. Thus, this classifier cannot obtain a sparse representation of all the characteristics that it encounters in this group.
On the other hand, the genre with the highest correct classification percentage is classical. It is important to mention that this category has by far the most training songs of any genre.
The average correct classification percentage is

$$p = \frac{\text{number of correctly classified songs}}{\text{total number of songs}} = 85.59\% \tag{29}$$

Since the number of test songs to be grouped in each genre is different another metric is also introduced

$$p_{\text{normalized}} = \sum_{g \in \text{genres}} p_g \times \text{correct}_g = 85.30\% \tag{30}$$

where p_g is the probability of appearance of each genre g and correct_g is the percentage of correctly identified songs of genre g. So, this represents a normalized metric.

Both the training and the test databases are available on the ISMIR 2004 web page. The best results for the ISMIR 2004 database obtained in recent studies are depicted in Table 1.

Table 1. Best results obtained on the ISMIR 2004 database

Authors	p
Y. Panagakis et al. (2010)	94.93%
our approach	85.59%
Holzapfel et al. (2008)	83.50%
Pampalk et al. (2005)	82.30%
Bergstra et al. (2006)	82.30%
I. Panagakis et al. (2008)	80.95%
Lidy et al. (2005)	79.70%

Only the results cited in Y. Panagakis et al. (2010) are higher than the ones presented here, the main reason being that this paper describes a whole framework for music genre classification. Although the classification algorithms in both cases rely on sparse representations, the difference in the results comes from the fact that Y. Panagakis et al. (2010) introduces a multilinear sub-space analysis method that reduces the dimensionality of the music signal's representation. On the other hand, this paper describes a method that uses a single, simple feature extracted from the audio signal. An important observation regarding the method presented here is that, even when the classification is not the right one, the correct genre has a small residual, even though it is not the smallest one. To illustrate this fact, Table 2 describes the position of the classification error associated with the true genre of the songs.

Table 2. Order of residual errors magnitudes

	1	2	3	4	5	6
classical	316	3	0	1	0	0
electronic	86	20	6	3	0	0
jazz-blues	22	3	0	1	0	0
metal	27	11	4	0	2	1
rock-pop	90	4	4	2	0	1
world	83	25	11	3	0	0

For example, the last line of Table 2 reads: 83 songs belonging to the world genre had the smallest classification error in the world genre dictionary (i.e. they were correctly classified), 25 world songs that were incorrectly classified had the second smallest error in the world genre dictionary, 11 world songs that were incorrectly classified had the third smallest error in the world genre dictionary, and so on. As Table 2 depicts, the correct genre is in the top 3 smallest residuals in 98.07% of the cases.

4. CONCLUSIONS

This paper describes an efficient and robust solution to the problem of classifying songs into music genres using only information from the digital audio signal itself. The presented approach uses as feature vectors the LPC coefficients computed on the frames of the audio signal. The feature vectors are then clustered using a sparse representation technique, namely the K-SVD algorithm. The presented method is applied on the ISMIR 2004 database and the average correct classification percentage obtained is 85.59% which is comparable to the state-of-the-art solutions that exist. Future work that improves the results presented in this paper includes: feeding the classifier additional features in order to improve the correct classification accuracy, using less of the audio signal from which to extract the features to speed up the method, adding an extra classification step for the top 3 genres with the smallest initial residuals and applying some fast dimensionality reduction algorithm to reduce the size of the original problem.

REFERENCES

T. Li, M. Ogihara and Q. Li, *A comparative study on content-based music genre classification*, Proc. 26th Int. ACM SIGIR Conf. Research and Development in Information Retrieval, Toronto, Canada, pp. 282-289, 2003.

N. Scaringella, G. Zoia and D. Mlynek, *Automatic genre classification of music content: A survey*, IEEE Signal Processing Magazine, vol. 23, no.2, pp. 133141, 2006.

S. Sukittanon, L. E. Atlas and J. W. Pitton, *Modulation-scale analysis for content identification*, IEEE Trans. Signal Processing, vol. 52, no. 10, pp. 30233035, Oct. 2004.

L. R. Rabiner and B. H. Juang, *Fundamentals of speech recognition*, PTR Prentice Hall, 1993.

H. G. Kim, N. Moreau and T. Sikora, *MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval*, John Wiley & Sons, 2005.

S. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, IEEE Trans. on Signal Processing, vol. 41, pp. 33973415, 1993.

V. N. Temlyakov, *Greedy algorithms and m-term approximation*, J. Approx. Theory, pp. 117145, 1999.

S. S. Chen, D. L. Donoho and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., vol. 20,no.1 pp. 3361, 1998.

M. Aharon, M. Elad and A. Bruckstein, *K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation*, IEEE Trans. Signal Processing, vol. 54, no. 11, pp. 43114322, Nov. 2006.

ISMIR 2004 web page, ISMIRWeb http://ismir2004.ismir.net/genre_contest/index.htm .

M. Journee, Yu. Nesterov, P. Richtarik and R. Sepulchre, *Generalized power method for sparse principal component analysis*, The Journal of Machine Learning Research, vol 11., pp. 517-553, 2010.

A. M. Bruckstein, D. L. Donoho and M. Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM REVIEW Society for Industrial and Applied Mathematics, vol. 51,No . 1,pp. 3481, 2009.

L. Ljung, *System Identification - Theory for the User*, Prentice Hall, Upper Saddle River, N.J., 1999.

Y. Panagakis and C. Kotropoulos, *Music genre classification via topology preserving non-negative tensor factorization and sparse representations*, IEEE Trans. Audio, Speech, and Language Processing, pp. 249 - 252, 2010.

A. Holzapfel and Y. Stylianou, *Musical genre classification using nonnegative matrix factorization-based features*, IEEE Trans. Audio, Speech, and Language Processing, vol. 16, no. 2, pp. 424-434, 2008.

E. Pampalk, A. Flexer and G. Widmer, *Improvements of audio-based music similarity and genre classification*, Proc. 6th Int. Symp. Music Information Retrieval, pp. 628-633, London, UK, 2005.

J. Bergstra, N. Casagrande, D. Erhan, D. Eck and B. Kegl, *Aggregate features and AdaBoost for music classification*, Machine Learning, vol. 65, no. 2-3, pp. 473-484, 2006.

I. Panagakis, E. Benetos and C. Kotropoulos, *Music genre classification: A multilinear approach*, Proc. 9th Int. Symp. Music Information Retrieval, pp. 583-588, Philadelphia, USA, 2008.

T. Lidy and A. Rauber, *Evaluation of feature extractors and psycho-acoustic transformations for music genre classification*, Proceedings of the Sixth International Symposium on Music Information Retrieval, pp. 34-41, London, UK, 2005.