Automatic Musical Instrument and Note Recognition

Frederico Malheiro and Sofia Cavaco

CITI, Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
frederico.malheiro@gmail.com, scavaco@fct.unl.pt

Abstract. We propose a sound recognizer that uses a reduced feature set to identify musical instruments from single notes in sound recordings as well as the note that was played. The recognizer learns a set of spectral features from the data using non-negative matrix factorization.
The accuracy of the recognizer is very high for both instrument and note classification: the recognition rate for instruments ranged from 94% to 100%, while for note identification ranged from 86% to 100%.
Keywords: sound classification, musical instruments recognition, monophonic, non-negative matrix factorization

1 Introduction

The principle of multimedia indexing is to provide information about the data, such as its title, author or, more importantly, its contents. Much of this information is still manually inputted. For it to be possible to transition to a completely automated process, a great number of problems still need to be solved. Due to their variety, these problems are studied in numerous areas. Music, for instance, is studied by a cross discipline known as Music Information Retrieval, which embodies many research topics [3]. Its main goal is to develop methods that automatically extract and organize information from large collections of music (a daunting task to perform manually [7]). Automatic recognition of musical instruments belongs to this area and, unsurprisingly, is in itself an immense topic, having many different concepts and approaches. Two classes can be clearly distinguished in terms of recordings: monophonic, when only single notes are played, and polyphonic, when various notes are played simultaneously.

Monophony has been widely researched by exploring more efficient and precise combinations of features (for instance, [1]). Polyphony, on the other hand, tends to be more realistic due to the notion that, in the majority of cases, recordings will contain more complex sounds as opposed to single note compositions.

There are two distinct approaches to handle polyphony. The first focuses on developing methods of retrieving, directly from the mixed audio signal, information used to identify the instruments present in the mix [4, 9]. The second uses statistical sound source separation algorithms, such as independent component

analysis and non-negative matrix factorization (NMF), to learn sound features that can characterize the instruments or notes in the signal [2, 5, 6].

We propose an instrument recognizer that uses NMF to learn a reduced set of spectral features from sounds and a k-NN classifier to classify the data. The recognizer achieves very high recognition rates, not only determining the instrument of the sample, but also the note being played. While the tests presented here use monophonic sounds, the system can also deal with polyphonic sounds: we have performed some preliminary tests on polyphonic sounds, which confirm that the system is able to recognize the instruments and notes of these sounds.

2 The Recognizer

The proposed recognizer does not use a set of pre-defined features, instead, it learns them from the data using NMF[1]. In the training phase, the recognizer starts by normalizing the amplitude of the sounds, computing their spectrograms (namely, $\mathbf{S}_1$ to $\mathbf{S}_\mathrm{N}$), and concatenating all the spectrograms to produce a single matrix $(\mathbf{S}_1, \ldots, \mathbf{S}_\mathrm{N})$ [2], where N is the number of sound samples.

The NMF of the concatenated spectrograms produces two matrices: $\boldsymbol{\Theta}$, the mixing matrix, whose columns consist of the spectra that characterize the sounds in the training data set, and $\mathbf{P}$, the source matrix, whose lines contain the temporal envelopes of the sounds. Using these matrices, the training data set $(\mathbf{S}_1, \ldots, \mathbf{S}_\mathrm{N})$ can be expressed as $(\mathbf{S}_1, \ldots, \mathbf{S}_\mathrm{N}) = \boldsymbol{\Theta}\mathbf{P}$.

The spectra in $\boldsymbol{\Theta}$ are the sound features that later will be used in the classification stage of the recognizer. The temporal envelopes in $\mathbf{P}$ contain the values of these features. In other words, the columns of $\boldsymbol{\Theta}$ are spectral basis functions that define a new space where the data is now represented. The envelopes in $\mathbf{P}$ are vectors of coefficients, which are the coordinates of the data in this space, that is, each frame in $(\mathbf{S}_1, \ldots, \mathbf{S}_\mathrm{N})$ is now represented in this new space of spectral basis functions by a new set of coefficients. If we consider all the coefficients related to the frames of one spectrogram and one basis function, we have a vector of coefficients, which is a temporal envelope. Now we can define matrix $\mathbf{P}_n$ which contains all the vectors of coefficients (or temporal envelopes) that are associated to spectrogram $\mathbf{S}_n$. The data set can then be expressed as:

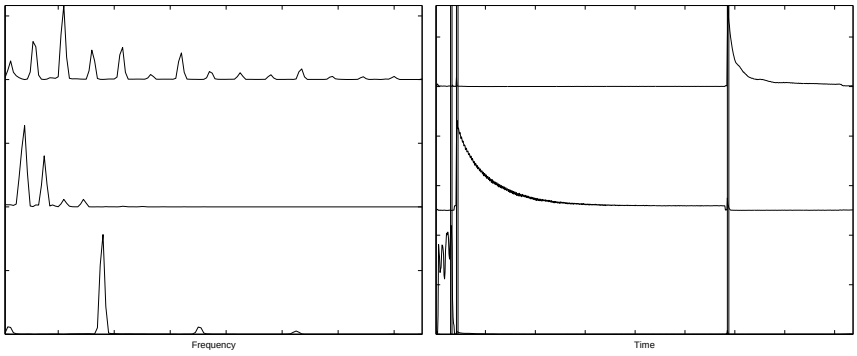
$$(\mathbf{S}_1, \ldots, \mathbf{S}_\mathrm{N}) = \boldsymbol{\Theta}\,(\mathbf{P}_1, \ldots, \mathbf{P}_\mathrm{N})\ , \tag{1}$$

where the ith row of each matrix $\mathbf{P}_n$ is associated to the ith basis function (the ith column) of $\boldsymbol{\Theta}$.

In terms of the classifier, two other concepts are relevant: training and test feature vectors. A training feature vector is an M dimensional vector of coefficients from M temporal envelopes from one sound, where M is the number of basis functions in $\boldsymbol{\Theta}$. More specifically, the training feature vector $\boldsymbol{t}_{i,n}$ =

[1] Our code was developed in MATLAB and we used an NMF software package by Virtanen [8].

[2] ($\mathbf{A}$, $\mathbf{B}$) represents two concatenated matrices.



(a) Frequency (b) Time

Fig. 1: (a) The spectral basis functions in $\boldsymbol{\Theta}$ and (b) the temporal envelopes in $\mathbf{P}$ learned by NMF of the spectrograms of three different sounds ($A5_{flute}$, $F3_{guitar}$ and $C4_{piano}$). (a) We call these spectra $\boldsymbol{\theta}_{C4_{piano}}$, $\boldsymbol{\theta}_{F3_{guitar}}$, and $\boldsymbol{\theta}_{A5_{flute}}$ (from top to bottom). (b) The three feature vectors found are indicated by the rectangles.

$(t_{1,i,n}, \ldots, t_{M,i,n})$ is a vector that contains coefficients associated to a small neighborhood of the ith frame of spectrogram $\mathbf{S}_n$: $t_{m,i,n} \in \{p_{m,i-\Delta i,n}, \ldots, p_{m,i+\Delta i,n}\}$, where $p_{r,c,n}$ is the coefficient in the rth row and cth column of $\mathbf{P}_n$, and Δi determines the size of the neighborhood that is analyzed, that is, how many frames the algorithm analyses in order to build the training feature vectors. In more detail, the algorithm evaluates $\mathbf{P}$ to determine the largest peak in each of the submatrices $(\mathbf{P}_1, \ldots, \mathbf{P}_N)$ to create N training feature vectors (one for each sound in the training set). When a peak is found in $\mathbf{P}_x$, the algorithm looks for a peak in the other lines of $\mathbf{P}_x$, such that the peaks are in close proximity, that is, the algorithm analyses only a limited number of columns (determined by Δi) such that the peaks correspond to the same event (such as the attack of a note).

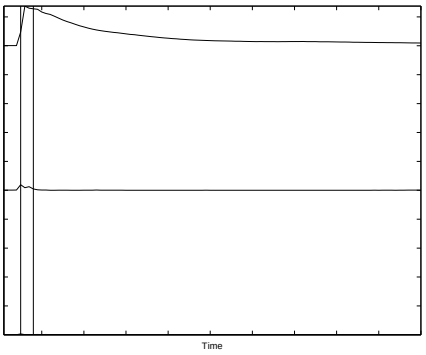
A test feature vector consists also of an M dimensional vector of coefficients extracted from M temporal envelopes, but while the envelopes in the training set are learned by NMF of the spectrograms, the envelopes in the test set are determined by the following equation:

$$\mathbf{P}_{\text{test sound}} = \boldsymbol{\Theta}^{-1}\mathbf{S}_{\text{test sound}} \ , \tag{2}$$

where $\boldsymbol{\Theta}^{-1}$ is the pseudoinverse of $\boldsymbol{\Theta}$.

The training feature vectors are used to train a k-NN classifier that uses the Euclidean distance metric. While we could extract several training feature vectors from each sound (for example, some from the attack, some from the sustain and/or the decay), here we use only one such vector per sound.

To illustrate these concepts, let us consider a simple scenario in which the training set consists of only three sounds ($A5_{flute}$, $F3_{guitar}$ and $C4_{piano}$) and the test set of a single sound ($C4'_{piano}$, from a different recording). NMF of the matrix of concatenated spectrograms $(\mathbf{S}_{A5_{flute}}, \mathbf{S}_{F3_{guitar}}, \mathbf{S}_{C4_{piano}})$ produces matrices $\boldsymbol{\Theta}$ and $\mathbf{P}$ (Figure 1). Each line in Figure 1a consists of one of the columns from $\boldsymbol{\Theta}$ and defines one spectra (or feature). Since matrix $\mathbf{P}$ contains the temporal envelopes associated with the three sounds, it can be represented as $\mathbf{P} = \left(\mathbf{P}_{A5_{flute}}, \mathbf{P}_{F3_{guitar}}, \mathbf{P}_{C4_{piano}}\right)$.



Time

Fig. 2: The temporal envelopes in $\mathbf{P}_{\text{test sound}}$ for the test sound described in section 2 ($C4'_{piano}$). The feature vector found is indicated by the grey rectangle.

Data		Features		
Instr.	Note	$\boldsymbol{\Theta}_{A5_{flute}}$	$\boldsymbol{\Theta}_{F3_{guitar}}$	$\boldsymbol{\Theta}_{C4_{piano}}$
Piano	$C4$	0,0184	0,0000	4,3936
Guitar	$F3$	0,3636	3,6361	0,1087
Flute	$A5$	3,2725	0,4984	0,01562

Table 1: Training feature vectors for the simple test case described in section 2. Each feature vector consists of three coefficients, one for each of the features: $\boldsymbol{\theta}_{A5_{flute}}$, $\boldsymbol{\theta}_{F3_{guitar}}$, and $\boldsymbol{\theta}_{C4_{piano}}$ from Figure 1a.

The next step consists of evaluating $\mathbf{P}$ to determine the largest peak in each of the submatrices ($\mathbf{P}_{A5_{flute}}$, $\mathbf{P}_{F3_{guitar}}$, and $\mathbf{P}_{C4_{piano}}$) to create three training feature vectors (one for each sound in the training set) as was described above. These training feature vectors are marked by the grey rectangles in Figure 1b (and their values are presented in Table 1). Each of these vectors represents a single note and will later be used to train the k-NN classifier.

Finally, to classify a test sound, the algorithm represents it in the same space as the training feature vectors, that is, the space defined by $\boldsymbol{\Theta}$: it normalizes the amplitude, computes the spectrogram, and through equation 2 obtains a matrix $\mathbf{P}_{\text{test sound}}$, of coefficients. Figure 2 shows the matrix obtained for the test sound used with the simple test case described above. Since the highest peak is in the top row, means that this sound has a spectra very similar to that defined by the top spectra in Figure 1a, that is $\boldsymbol{\theta}_{C4_{piano}}$. The algorithm then obtains a feature vector by computing the maximum peak and applying the method described above: $(0.0232, 0.0121, 4.8865)$. With this test feature vector, the k-NN classifier, trained with the values in Table 1 and with $k = 1$, can correctly determine the note and instrument of the test sound as $C4$ and piano.

3 Result Analysis

The data set used to train and test the classifier is composed of single notes from 3 different instruments using various recording conditions and playing techniques. The instruments used were a Fender Stratocaster Electric Guitar, a Yamaha Grand Piano and a Hohner Recorder/English Flute. The recordings were made with a microphone (AKG C1000S) connected to a USB Audio Interface (Edirol UA-25). All recordings were digitized using a sampling frequency of 44100 Hz, 16 bit depth and 1 channel (mono).

In all, 48 notes (spanning 4 octaves) were recorded, of which 12 are guitar (C_3 to B_3)[3], 24 are piano ($F\sharp_3$ to F_5) and 12 are flute (C_5 to B_5). Different recordings were made for each note, giving a total of 288 sound samples: six recordings of each guitar note using different pickups (72 samples), six recordings of each flute note with two different playing techniques, *sustain* and *staccato* (72 samples) and six recordings of each piano note with three different playing techniques, *strong sustain*, *soft sustain* and *staccato* (144 samples).

To test the recognizer, we performed an n-fold cross validation experiment: the training sets were created with all the sounds except one from each note and instrument (that is, all training sets had 240 sounds). The remaining 48 sounds were used as test data (all the sounds from the same instrument in the test data used the same technique, such as *staccato* or *sustain*). Therefore all sounds were used for training at five of the experiments and for testing in one of the experiments. From these six different experiments we gathered results using two different methods: (1) the samples from the test data were classified individually; (2) the system detected and classified the samples from randomly generated sound files containing twenty samples in sequence (distanced by two seconds of silence). Since the maximum number of identical notes for the same instrument in the training sets is 5, we used $k = 5$ for the k-NN classifier.

	Instrument		Note		Total number
	#	%	#	%	of samples
Flute	70	97%	62	86%	72
Guitar	72	100%	72	100%	72
Piano	137	95%	143	99%	144

Table 2: Results for the tests with individual sounds.

	Instrument		Note		Total number
	#	%	#	%	of samples
Flute	27	100%	27	100%	27
Guitar	27	100%	27	100%	27
Piano	62	94%	65	98%	66

Table 3: Results for the tests with random sequences of sounds.

The recognition results were very positive in both scenarios. The values for the classification of individual samples are presented in Table 2[4], where we can verify that the accuracy of the system is consistently high (while obscured by the compounded values, the lowest result was 75% on one of the flute note identification tests). All guitar samples were correctly classified as *guitar sounds* and all the guitar notes were correctly identified. The instrument classification was also very high for flute and piano (97% and 95%, respectively), with only one piano sample incorrectly identified. The lowest results were the note classification of flute samples, with a very acceptable recognition rate of 86%.

For the randomized sequences of samples we obtained even better results, see Table 3, with an accuracy percentage of almost 100% for all the instruments and notes tested. Only 4 piano samples were misclassified as being from another instrument and one piano sample was misclassified as the wrong note.

[3] This representation is called "scientific pitch notation" and identifies western notes by combining the letter-name (A to F) accidentals and a number identifying the pitch's octave.

[4] The columns *Instrument* and *Note* show the number and percentage of correctly classified samples in terms of instrument and note, respectively.

4 Conclusions

We proposed a musical instrument recognizer that, instead of using a set of predefined features, uses NMF to learn a set of spectral features from the data. When tested with samples from flute, guitar and piano, spanning four octaves, this approach proved extremely successful. The recognition rates were very high for the classification of the instruments (ranging from 94% to 100%) as well as for identification of the notes from the samples (between 86% and 100%).

While all presented tests used monophonic sounds, we have also performed some preliminary tests on polyphonic sounds, which have a highly promising recognition rate. As future work we intend to test other musical instruments as well as further explore the application of this technique on polyphonic sounds.

Acknowledgements

This work was part of the Videoflow project and partially funded by *Quadro de Referência Estratégica Nacional* (QREN) and *Fundo Europeu para o Desenvolvimento Regional* and *Programa POR Lisboa*.

References

1. G. Agostini, M. Longari, and E. Pollastri. Musical instrument timbres classification with spectral features. *EURASIP J. Appl. Signal Process.*, 2003:5–14, 2003.
2. E. Benetos, M. Kotti, and C. Kotropoulos. Musical Instrument Classification Using Non-Negative Matrix Factorization Algorithms and Subset Feature Selection. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 221–224, Toulouse, France, May 2006.
3. M. Fingerhut. Music information retrieval, or how to search for (and maybe find) music and do away with incipits. In *IAML-IASA 2004 Congress*, Oslo, Norway, August 2004.
4. W. Jiang, A. Wieczorkowska, and Z. W. Ras. Music instrument estimation in polyphonic sound based on short-term spectrum match. In *A.-E. Hassanien, A. Abraham, F. Herrera (Eds.), "Foundations of Computational Intelligence. Volume 2. Approximate Reasoning". Studies in Computational Intelligence Vol. 202*, pages 259–273. Springer Berlin Heidelberg, 2009.
5. P. S. Lampropoulou, A. S. Lampropoulos, and G. A. Tsihrintzis. Musical instrument category discrimination using wavelet-based source separation. In *New Directions in Intelligent Interactive Multimedia*, pages 127–136. Springer, 2008.
6. L. G. Martins, J. J. Burred, G. Tzanetakis, and M. Lagrange. Polyphonic instrument recognition using spectral clustering. *Proc. International Conference on Music Information Retrieval (ISMIR)*, 2007.
7. G. Tzanetakis, A. Kapur, W. A. Schloss, and M. Wright. Computational ethnomusicology. *Journal of Interdisciplinary Music Studies*, 1(2):1–24, 2007.
8. T. Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing*, 15, 2007.
9. Xin Zhang and Zbigniew W. Raś. Sound isolation by harmonic peak partition for music instrument recognition. *Fundam. Inf.*, 78:613–628, December 2007.