Neural Networks: Project Report.

Irvin Aloise, Mirco Colosi
Sapienza University of Rome

November 30, 2016

1 Introduction

MIDI databases are still very popular nowadays and the scientific community is always improving the way in which it is possible to categorize them. MIDI - which stands for *Musical Instrument Digital Interface* - is a technical standard that allows to connect and make them communicate properly several musical tools, e.g. instruments, digital equipment and computers [1]. It is composed by messages containing informations about *notation*, *pitch*, *velocity*, *control signals* - that describe parameters like vibrato or the volume - and *clock* signals in order to synchronize the tempo of all the instruments.

With the diffusion of machine learning approaches, those classification methods are always more powerful and precise, taking advantage of MIDI informations together with other audio features, in order to achieve categorization of the tracks by *author*, *genre*, *style* and so on.

In this paper, instead, it has been used a more general approach: it has been designed and developed a mechanism that measures the similarity between tracks and, given a new instance returns the author which mostly suits that instance - chosen between the authors available in the training set. To do that, we used a *universal* similarity metric based on **Kolmogorov complexity** [2]. The peculiarity of this approach is that can be employed potentially on every kind of file with no modification and without the need of time expensive calculations for extracting audio features. Then, once that a similarity measure is retrieved, a simple *k-NN* has been employed to classify new instances.

The document is organised as follows: in Section 2 a brief overview of the related approaches is given; Section 3 describes the methodology used in this project together with some results; finally in Section 4 are reported conclusions and possible future improvements.

2 Related works

Most of the works in the literature try to classify MIDI songs by *genre*. This because generally datasets are stored by author but recognize the style could be useful in automated platforms, for example to suggest a new song related to the one that we are listening.

For example, **Basili et al** [4] use some *coarse-grain features* in order to classify MIDI files by genre. Those features are basically provided directly by the MIDI file, in order to evaluate how good MIDI describe symbolic music; they are the following ones:

- Melodic intervals
- Instruments
- Instrument classes and drumkits
- Meter and time changes
- Note extension

Those features are given as input to several machine-learning algorithm - *Naive Bayes*, *VFI*, *J48*, *PART*, *NNge* and *JRip* - to extract a genre prediction for new MIDI instances.

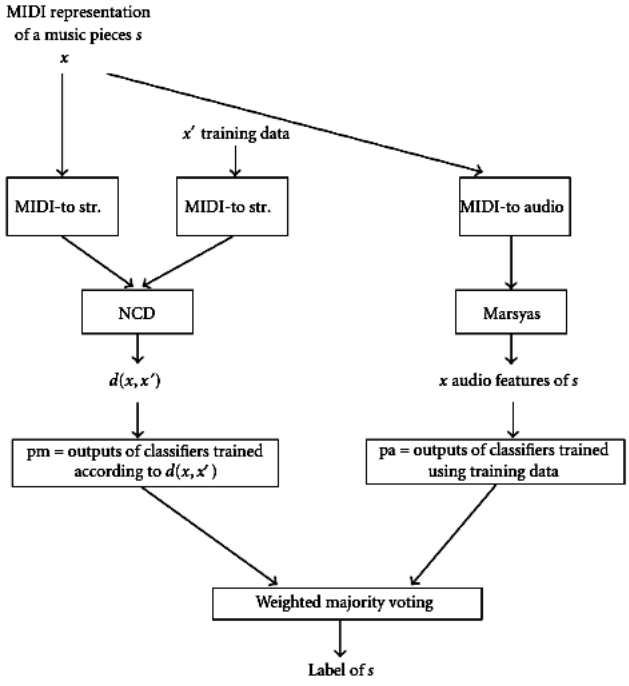

Figure 1: A block scheme representing Cataltepe et al.'s approach

Gnegy in his work [5] instead tries to return a *style* classification of the instances, intended as a functional description of a specific instruments in a song. More precisely, the styles used are the following:

- **Bass**: the song has a predominant single line of deep tones and the other harmonic structure are just supporters.
- **Lead**: a melodic line, e.g. vocal performance or guitar solo.

- **Rhythm**: in general represented by repeated chords that give the rhythmic structure to the melody.
- **Acoustic**: this style is characterized by multiple individual notes - or melody lines - often performed on the same instruments.

A large variety of audio features is used in this work, spanning from simple statistics - *mean pitch*, *pitch standard derivation* and many others - to very complex ones - *coverage*, *liricality* and several others. Many different machine-learning algorithm are then used in order to assign a *style* to new instances, e.g. *decision tree*, *logistic regression*, *k-NN*, *QDA* and *SVM*.

An approach very similar to what it has been developed in this project is used by **Cataltepe et al.** [6]: they used a combined approach that takes the advantage both of MIDI *and* audio features - those ones evaluated after a conversion into simple audio files of the MIDI pieces. As it is possible to appreciate in Figure 1, in order to classify the instances, they used the *NCD* (Normalized Compression Distance) - which is based on the same concept of the *similarity measure* employed in this project - and then a k-NN classifier together with another classifier that acts on audio features. The final vote takes is represented by the weighted majority vote resulting from the two classifiers.

Concluded this preliminary phase, the next Session will propose a deeper analysis on what it has been developed in this project.

3 Classification using Similarity Metric

This Section contains a brief overview of what it has been done in our project. It has been developed a k-NN classifier that uses a *Similarity Metric* based on Kolmogorov complexity in order to classify MIDI files by author.

Firstly there will be an introduction on the mathematical concepts behind the Similarity Metric and then it will be provided details about the actual implementation and results.

3.1 Similarity Metric based on Kolmogorov Complexity

The core idea behind this *Similarity metric* is the Kolmogorov complexity [3]: this can be intended as the length of the file's ultimate compression. More precisely, any object can be coded into strings, denoted by x and $K(x) = |x|$ represents the number of bits needed to computationally retrieve x.

Hence, it is possible to define the **similarity metric** between two files x and y as follows:

$$d_1(x, y) = \frac{K(x|y) + K(y|x)}{K(x, y)} \tag{1}$$

$$d_2(x, y) = \frac{\max\left(K(x|y), K(y|x)\right)}{\max\left(K(x), K(y)\right)} \tag{2}$$

Both definitions require the *conditional complexity* $K(x|y)$ but this quantity cannot be evaluated in close form. Thus, the conditional complexity is approximated as $K(x|y) \approx K(x, y) - K(y)$, where $K(x, y)$ is computed very intuitively

as the length of x and y concatenated together. In this project, it has been chosen to use the formula (1) to evaluate the distance between files.

3.2 Experimental results

In this project we implemented a k-NN classifier to assign the author to MIDI files using the distance described by (1).

It has been used a dataset of 600 MIDI files divided into 6 authors, each one with 100 entries; the chosen composers are the following: *Bach*, *Beethoven*, *Haendel*, *Mozart*, *Schubert* and *Vivaldi*. The whole dataset has been split into **training-set** - 60 files for each author - and **test-set** - 40 songs for each composer.

The project has been developed entirely in MATLAB. As first thing we cloned the dataset - $\mathcal{D}_1$ and $\mathcal{D}_2$ will indicate respectively original and cloned dataset. Then, $\mathcal{D}_2$ has been pre-processed according to the work of *Li et al.* [7]:

- *Timing* and *expression* informations of each song has been removed;
- *Multi-tracks* songs has been converted to single track, keeping only the highest pitch note between simultaneous notes.

After this preprocessing, it has been evaluated the similarity metric - according to (1) - for both training sets $\mathcal{D}_1^{train}$ and $\mathcal{D}_2^{train}$, obtaining a (360×360) matrix in which all the distances are stored, applying Algorithm 1.

Data: $\mathcal{D}_j$
Result: SM_j a (360×360) matrix with all the distances.
foreach $x_k \in \mathcal{D}_j$ **do**
  Convert x_k into a text file $\bar{x}_k$
  Zip $\bar{x}_k$ obtaining $\bar{x}_k^{zip}$
  Evaluate the length $K\left(\bar{x}_k^{zip}\right)$
  foreach $x_h \in \mathcal{D}_j$ **do**
    Convert x_h into a text file $\bar{x}_h$
    Zip $\bar{x}_h$ obtaining $\bar{x}_h^{zip}$
    Evaluate length $K\left(\bar{x}_h^{zip}\right)$
    Create concatenated files $\bar{x}_{kh}$ and $\bar{x}_{hk}$
    Zip $\bar{x}_{kh}$ and $\bar{x}_{hk}$ obtaining $\bar{x}_{hk}^{zip}$ and $\bar{x}_{kh}^{zip}$
    Evaluate the lengths $K\left(\bar{x}_{kh}^{zip}\right)$ and $K\left(\bar{x}_{hk}^{zip}\right)$
    Evalaute d_{kh} and d_{hk} according to (1)
  end
end

Algorithm 1: Evaluate distances on MIDI instances using the Similarity Metric

Therefore, it has been evaluated the similarity metric between each new instance of the test sets and the training files, in order to generate for each new MIDI file a *prediction* of its author label using k-NN. Finally, we collected some statistics for both the datasets, i.e. *precision* and *recall* of the method -

shown in Table 1 - and *confusion matrices* provided in Figure 2. It is possible to appreciate how the preprocessing phase enhance the classification results with respect to the raw MIDI dataset.

	Precision	Recall
$\mathcal{D}_1$	0.1514	0.1136
$\mathcal{D}_2$	0.2083	0.1907

Table 1: Precision and recall on the two datasets. It is possible to appreciate the boost in performances given by the preprocessing phase.

However, this method has some poor results compared to other state-of-the-art systems and this because it is designed to co-work with other feature-based classification or to preform a pre-clustering of the raw dataset.

4 Conclusions and future work

In this project it has been developed a classification system for MIDI file, based on the usage of k-NN together with the *Similarity Metric*. This last one is based on the concept of **Kolmogorov complexity** of a file: basically this can be approximated evaluating the length of the file in it's ultimate compression. This can be approximated using standard industrial compressors - e.g. *zip* - calculating the *Normalized Compressed Distance* (NCD).

The peculiarity of NCD is that it is not file specific - like a audio feature - but *universal*. As a matter of fact, in [2] the *Kolmogorov complexity* is used to create mitochondrial phylogeny tree and the language tree of 52 different languages.

It has been tested the classifier on a dataset of 600 MIDI classical songs, composed by 100 files for each author and divided in *training set* and *test set*. The classifier had to predict the author of new instances - taken from the test set - performing *k-NN* using the Similarity Metric defined by (1).

It has been performed the same experiment also on a dataset cloned from the standard one, but preprocessed according to [7]. It has been noticed that the preprocessing is useful to obtain better results in the classification phase.

What has been noticed is that a classifier based *only* on the Similarity Measure is too weak to perform state-of-the-art classification of MIDI file. Thus, it must be enforced exploiting other informations available, coming either from MIDI parameters or from audio features. Hence, one possible future development could be an extension of this classifier using one of the above-mentioned approaches.

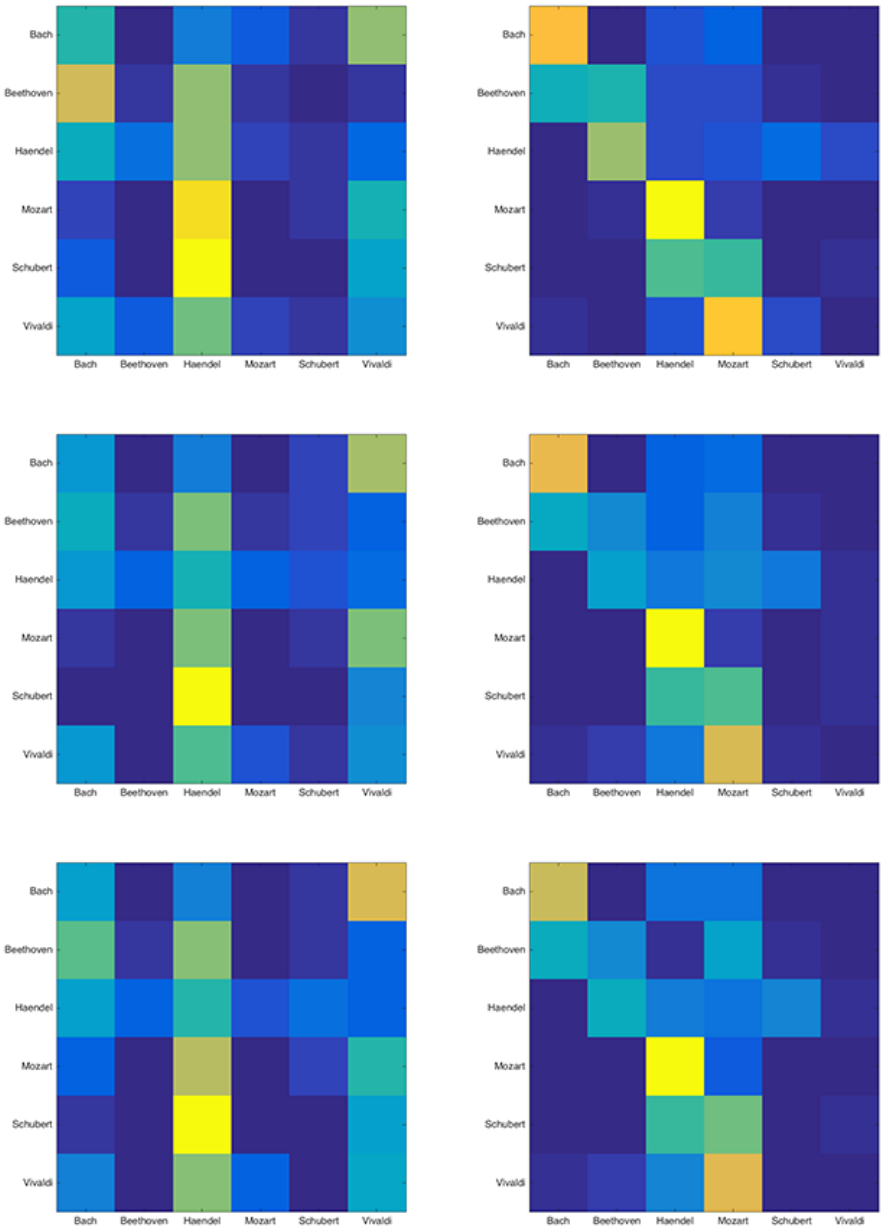

Figure 2: Confusion matrices on both the datasets using k-NN with $k = [3, 5, 7]$ - shown in rows 1,2 and 3 respectively. Left column represents the confusion matrices obtained from the raw dataset $\mathcal{D}_1$, instead the right column embeds the confusion matrices of the preprocessed dataset $\mathcal{D}_2$. Warmer colours indicate higher values.

References

[1] MIDI page on Wikipedia:
https://en.wikipedia.org/wiki/MIDI

[2] Li, Ming, et al. *The similarity metric* in *IEEE transactions on Information Theory*, 2004, 50.12: 3250-3264.

[3] Li, Ming, & Paul Vitànyi. *An introduction to Kolmogorov complexity and its applications*. Springer Science and Business Media, 2009.

[4] Basili, Roberto, Alfredo Serafini, and Armando Stellato. *Classification of musical genre: a machine learning approach* in *ISMIR*, 2004.

[5] Gnegy, Chet N. *Classification of Musical Playing Styles using MIDI Information*.

[6] Cataltepe, Zehra, Yusuf Yaslan, and Abdullah Sonmez. *Music genre classification using MIDI and audio features*, in *EURASIP Journal on Advances in Signal Processing* 2007.1 (2007): 1-8.

[7] Li, Ming, and Ronan Sleep. *Melody classification using a similarity metric based on Kolmogorov complexity*, in *Sound and Music Computing*, 2012 (2004).