Neural Networks: Project Report.

Irvin Aloise, Mirco Colosi
Sapienza University of Rome

November 29, 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 [ref. https://en.wikipedia.org/wiki/MIDI]. 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** [ref: LI, Ming, et al. The similarity metric. IEEE transactions on Information Theory, 2004, 50.12: 3250-3264.]. 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 remaining of the document is organized 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** [ref: Basili, Roberto, Alfredo Serafini, and Armando Stellato. "Classification of musical genre: a machine learning approach." ISMIR. 2004.] 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.

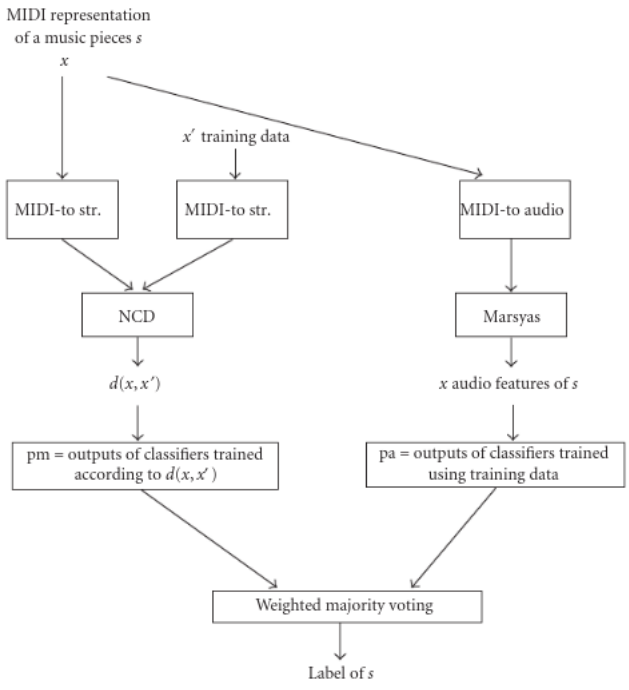

Figure 1: A block scheme representing Cataltepe et al.'s approach

Chet in his work [ref: Gnegy, Chet N. "Classification of Musical Playing Styles using MIDI Information."] 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.** [ref: Cataltepe, Zehra, Yusuf Yaslan, and Abdullah Sonmez. "Music genre classification using MIDI and audio features." EURASIP Journal on Advances in Signal Processing 2007.1 (2007): 1-8.]: 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.

Now that it has been completed an overview of the other approaches used by scientific community, it is possible to better analyse the techniques used 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 [ref: Li, Ming, & Paul Vitànyi. An introduction to Kolmogorov complexity and its applications. Springer Science and Business Media, 2009.]: 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.