Midi Classification using Similarity Metric based on Kolmogorov Complexity

SAPIENZA
UNIVERSITÀ DI ROMA

Faculty of Information Engineering, Computer Science and Statistics

M.Sc. in Artificial Intelligence and Robotics

AA 2016-2017

PROFESSOR:

Aurelio Uncini

PRESENTERS:

Irvin Aloise, Mirco Colosi

Introduction

The project proposes a method to *classify* MIDI files by author, evaluating a similarity measure based on the concept of *Kolmogorov complexity*.

In general, classification is the task of assigning instances to one of several predefined categories, called labels. This task has main two sub-categories.

Supervised Approach:

Dataset $\mathcal{D}$ has a **labelled training-set** $\mathcal{D}^{Train} = \langle(x_i, y_i)\rangle$ and you want to generalize the labelling rule in order to classify new instances coming from the **test-set** $\mathcal{D}^{Test} = \langle(x_i)\rangle$.

Unsupervised Approach:

Dataset $\mathcal{D} = \langle(x_i)\rangle$ is has **no labelled instances**, thus you want to retrieve *patterns* in the dataset in order to generate clusters of data.

In this case, it has been used a **k-NN** classifier to generate label, exploiting the similarity metric based on the Kolmogorov complexity of the files.

MOTIVATIONS

Kolmogorov complexity's main feature is that it is *universal* and, thus, can be employed potentially for every data-type without loss of generality.

WHY IS MIDI SUITABLE FOR SUCH TASK?

- POPULAR
- LIGHT-WEIGHT
- MUSICAL INFORMATION
- EASY TO MANAGE

RELATED WORKS

Classify MIDI songs is a well-known problem in the literature, especially to infer the *genre* of the song in order to identify related songs.

SEVERAL DIFFERENT APPROACHES HAVE BEEN TRIED:

COARSE-GRAIN FEATURES → This approach exploits MIDI raw musical feature to classy files by genre. Examples of features are: **Melodic intervals, instrument classes, drum-kits, meter, note extension.**

COMPLEX AUDIO FEATURES → Classes represent the *style* of the song e.g. Bass, Lead, Rhythm, Acoustic. A large set of audio feature is exploited, from simple statistics – **mean pitch, pitch SD**, etc. – to more complex ones – **coverage**, **liricality** and so on .

AUDIO FEATURES & SIMILARITY METRIC → Similar to the one used in this project, composed by two parts:

- NCD-based classifier
- Feature-based classifier

Similarity metric based on Kolmogorov Complexity

Kolmogorov complexity can be intended as the length of the file's ultimate compression. Thus, the similarity between two object is defined as follows:

$$d(x, y) = \frac{K(x|y) + K(y|x)}{K(x, y)}$$

- x and y represent the two files to compare coded into strings.
- $K(x) = |x|$ denotes the number of bits needed to computationally retrieve x.
- $K(x, y)$ is simply computed as length of the ordinated concatenation of the two files.
- $K(x|y)$ cannot be evaluated in close form, thus, it is approximated as $K(x|y) \approx K(x, y) - K(y)$.

Dataset description

It has been implemented a **k-NN** classifier in order to assign the author to MIDI files, using the previously specified distance.

MIDI Dataset:

Population → This Dataset is composed by 600 files, divided into 100 files for each authors taken in consideration. The chosen composers are: *Bach, Beethoven, Mozart, Schubert* and *Vivaldi.*

Training - Test sets → The dataset has been divided as follows:

- 60 files for each author as training set
- 40 files for each author as test set

Experiments have been done using the raw MIDI dataset and a cloned version of itself that has been **pre-processed:**

- *Timing* and *expression* information have been removed.
- *Suppression* of multi-tracks.

Algorithm implementation

Firstly, it has been evaluated the similarity between all the files in the training set, obtaining a (360×360) matrix in which all distances are stored.

It has been exploited the following algorithm on both datasets:

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}
  end
end

EXPERIMENTAL RESULTS

Each new instance of the test-set as been classified, predicting the author label using a k-NN classifier built on the similarity metric previously described.

OUTCOMES:

CONFUSION MATRICES

- Experiments employed different values for the classifier parameter $k = \{3,5,7\}$.
- Both dataset taken in consideration.
- Pre-processed dataset obtained better scores.

PRECISION AND RECALL

- Both dataset taken in consideration.
- Mean values of precisions and recalls obtained with different k.

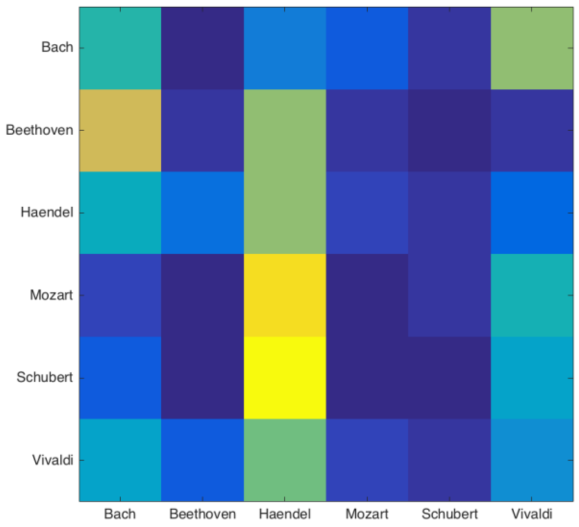
Experimental results

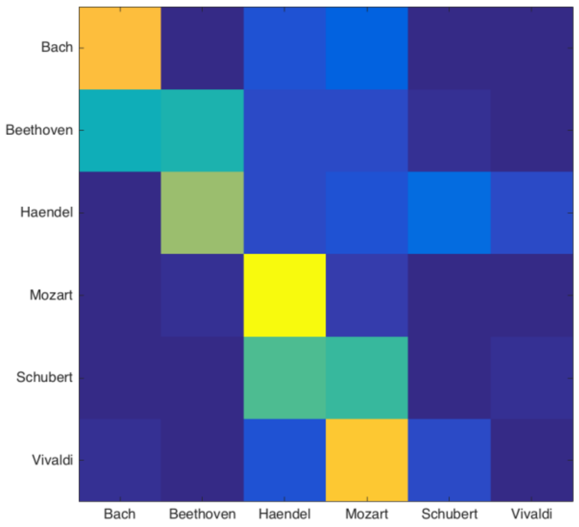
Each new instance of the test-set as been classified, predicting the author label using a **k-NN** classifier built on the similarity metric previously described.

Outcomes:

Confusion Matrices

Precision and recall





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

CONCLUSIONS AND FUTURE WORKS

The project provides a classification system for MIDI files, based on a *k-NN* that uses the Similarity Metric computed through the evaluation of the *Kolmogorov Complexity*.

EXPERIMENTS RAN ON A 600-FILES-DATASET SHOWN TWO MAIN OUTCOMES:

PRE-PROCESSING IS REQUIRED

- Cloned dataset provides better results both in precision and recall.
- Same execution time for both datasets.

WEAK CLASSIFICATION PERFORMANCES

- Classification using only a similarity metric is not accurate.
- Embedding into a bigger classification system – based on audio features – would improve performances.