Automatic Recognition of Piano Music Compositional Styles

GIOVANNI COSTANTINI, DANIELE CASALI, MASSIMO CAROTA
Department of Electronic Engineering
University of Rome “Tor Vergata”
Via del Politecnico, 1. I-00133 Rome
ITALY

Abstract: - In this paper, we present a system for automatic recognition of piano music compositional styles. It is based on a classifier that takes as input a set of MIDI files containing music pieces composed by Bach, Mozart, Beethoven and Debussy. Several features are extracted from MIDI data: the number of times each degree of the tempered scale is repeated in the whole piece, the interval between a note and the preceding one, and finally a feature based on the Forti’s representation of chords. Each piece of a training set is associated with the corresponding author. After a learning phase, the classifier is able to recognize the author of an unknown piece, by analyzing and classifying the feature set extracted from the unknown piece.

Key-Words: - Author recognition, music style classification, neural networks.

1 Introduction
The work of every great musical composer is characterized by a well defined set of features, which can be found in all of his pieces and that makes them recognizable: like a fingerprint, the style of a composer remains impressed into his works. This fact allows the listener to guess the composer of an unknown piece, if some of the work of that composer has been heard before. Such kind of recognition is more or less consciously accomplished by the human brain, but the same task can be a great challenge for a machine. Many works regarding style recognition in contemporary pop music have been done, and they are eased by the use of a great variety of parameters. In many cases, for example, the knowledge about the musical instruments employed in a score – coded according to the “General MIDI” standard – can be of great help in the recognition of the style.

Works that recognize single composers of a particular kind of music, merely analyzing his style and his choices in the organization of melodies and chords, are more rare. Bach chorales have been deeply investigated, and many regularities have been found, so that an expert system [1] capable to harmonize any given soprano part in imitation of Bach style could be implemented. Anyway, finding explicit rules behind the style of a composer is not only a hard task for a machine, but it could be even inadequate for what concerns generalization capabilities.

Our aim is to design a learning system that could analyze, learn, and then recognize/classify virtually any composer. Buzzanca [2] designed a neural network based style recognizer that works on a set of 13 composers born from year 1440 to 1660. Our aim is to develop a system that goes beyond these limits, being able to work with up to the modern styles of the XX century.

We selected four composers: Bach, Mozart, Beethoven, and Debussy, representative of four different styles.

Starting from the available scores, we have extracted different sets of features and classified them with four different kinds of classifiers.

The paper is organized as follows: in the second Section we describe the three sets of extracted and classified features; in the third Section we describe the employed classifiers; in the fourth Section we show results; finally, the last Section is dedicated to conclusions.

2 Extracted Features
In this work, we considered three sets of features: played notes, intervals, Allen Forti numbers.

2.1 Notes
The most important feature that distinguishes one composer from another and that we considered in our work is the distribution of played notes. If we make a statistics of most of used degrees of the tempered scale, independently of the octave, we obtain a set of 12 values, corresponding to the percentage of usage of every note. Let us first suppose that there are no modulations in the piece. When we order this set from the most used note to
the less used note, the first one is supposed to be the keynote, while the following six positions will have different percentages that depend on the particular composer, particularly on his stronger or weaker adherence to the tonal system. Positions after the seventh will most likely represent notes out of key, characterized by very low rates. Even this information can be very useful to distinguish a composer from another.

In musical compositions, there can be modulations (tonality changes), too. This kind of representation can still be useful, because this statistics also changes according to the number and kind of the tonality changes.

2.1 Intervals
Another feature that we take into account is the interval between subsequent notes, measured in half tones. We build a vector of 25 elements, each related to the number of occurrences of a corresponding interval, from –12 to 12, zero included. Intervals greater than 12 half tones are ignored. Even this feature is essential in recognizing the composer, because it is strictly connected with the tonality system: some kind of intervals are considered more “consonant” or “dissonant” than others, but the way they are treated varies form composer to composer.

2.1 Forte numbers
In order to have more features at our disposal, we also take into account chords, that is to say more notes playing at the same time. There are several ways of labeling and classifying chords, other than simply listing the notes they are composed of. A notation like “Cmaj7” and similar is very popular and of practical use, but it is not very suitable for our work. For the purpose of classification, we tried to exploit this kind of notation, but the results achieved were absolutely disappointing. One of the drawbacks of this kind of notation is that the same set of notes can be sometimes associated with more than one chord, depending on the tonality that the composer is using at that moment. Hence, the association is ambiguous.

A notation that better matches our purposes has been proposed by Allen Forte, who associates every possible combination of the 12 notes to a “prime form”. Then he lists all prime forms, associating to each of them two numbers: the former represents the number of notes inside the chord, while the latter is the index of the prime form the chord is associated to. For example, all major and minor triads have the same prime form, namely (0,3,7); consequently, their Forte number is 3-11: 3 is the number of notes in the chord, and the prime form associated is the 11th prime form with three notes. The major or minor scale, which has as prime form (0,1,3,5,6,8,10), has as Forte number 7-35. Hence, Forte numbers can give to every combination of any number of notes a set of only two values. The total number of combinations for Forte numbers is 208.

3 Classifiers
We made a comparative study of various classification systems, in order to find the one that’s as satisfactory as possible for our applications. We tested four classifiers: J48, Naïve Bayesian Classifier, Support Vector Machines, and Multilayer Perceptron. For all of them we leaned on a tool written in Java, called Weka [4].

3.1 Decision Tree
Decision trees represent a supervised approach to classification. A decision tree is a simple structure where non-terminal nodes represent tests on one or more attributes and terminal nodes reflect decision outcomes. J.R. Quinlan [5] has popularized the decision tree approach with his research spanning more than 15 years. The latest public domain implementation of Quinlan's model is C4.5. We used the Weka version of this classifier, known as J48.

We will summarize here the general approach followed by the classifier:
1. Choose an attribute that best differentiates the output attribute values.
2. Create a separate tree branch for each value of the chosen attribute.
3. Divide the instances into subgroups to reflect the attribute values of the chosen node.
4. For each subgroup, terminate the attribute selection process if:
   a) All members of a subgroup have the same value for the output attribute, terminate the attribute selection process for the current path and label the branch on the current path with the specified value.
   b) The subgroup contains a single node or no further distinguishing attributes can be determined. As in (a), label the branch with the output value seen by the majority of remaining instances.
5. For each subgroup created in (3) that has not been labeled as terminal, repeat the above process.
3.2 Naïve Bayes Rule Generator
Naïve Bayes is a rule generator based on Bayes’s rule of conditional probability [6]. It uses all attributes and let them to contribute to decision, as if they were all equally important and independent of each other. The expression of conditional probability is the following:

\[
Pr[H \mid E] = \frac{Pr[E_1 \mid H] \cdot Pr[E_2 \mid H] \cdots Pr[E_n \mid H]}{Pr[E]}
\]

Pr[A] denotes the probability of event A, Pr[A|B] denotes the conditional probability of event A with respect to event B. \(E_n\) is the \(n\)-th attribute of the instance, \(H\) is the outcome in question, and \(E\) is the combination of all the attribute values.

3.3 Support Vector Machines
The Support Vector Machine (SVM) algorithm [7,8] was invented by Vladimir Vapnik. It identifies a hyperplane that splits the data into two classes with the maximum-margin. Given some training examples labeled either "yes" or "no", a maximum-margin hyperplane divides the "yes" from the "no" examples, such that the distance between the hyperplane and the closest examples (the margin) is maximized.

The use of the maximum-margin hyperplane is motivated by Vapnik Chervonenkis theory, which puts a probabilistic test error bound at our disposal that is minimized when the margin is maximized. However, the utility of this theoretical analysis is sometimes dubious, given the large slack associated with these bounds: the bounds often predict more than 100% error rates.

The parameters of the maximum-margin hyperplane are derived by solving a quadratic programming (QP) optimization problem. There exist several specialized algorithms for quickly solving the QP problem that arises from SVMs. The most common method for solving the QP problem is Platt’s SMO algorithm.

3.4 Multilayer Perceptron
Multilayer Perceptron (MLP), also known as Backpropagation Neural Network (BPNN), due to its learning algorithm, is one of the most common neural network structures [9]. This kind of structures are simple and effective, and have found home in a wide assortment of machine learning applications, such as character recognition.

BPNNs start as networks of nodes arranged in three layers: the input, hidden, and output layers. The input and output layer nodes simply buffer, respectively, network input and output, while the hidden layer models the non-linear relations existing between inputs and outputs. Before any data has been run through the network, the weights of the synapses are random, that is to say the network is much like a newborn’s brain: developed but without knowledge.

When we present a pattern to the network input, each input node takes the value of the corresponding attribute in the input pattern. These values are then “fired”; each node in the hidden layer performs a weighted sum of all input attributes. If the sum exceeds the node’s threshold value, it fires a value of ‘1’; otherwise it fires a value of ‘0’. The same process is repeated in the output layer on values from the hidden layer, and if the threshold value is exceeded, the input pattern is considered classified. During the training phase of the network, once a classification has been given, it is compared to the desired classification. The classification error, if it occurs, is then “backpropagated” through the network, and consequently the weights of hidden and output layer nodes are adjusted. The weights are updated according to a gradient descent algorithm that tries to find a global minimum of the error curve. Unfortunately, the found minimum is not always the global minimum, and the network settles in a non-optimal configuration. Sometimes, we can avoid this situation, by increasing or decreasing the number of hidden layer nodes or even by rerunning the algorithm (to initialize again the weights to different random values, hoping to be more lucky).

4 Results
We used a set of 274 MIDI files of piano scores: 64 pieces composed by Bach, 58 by Debussy, 60 by Mozart and 64 by Beethoven. For every composer we selected randomly 60% of the files as learning set, and the remaining 40% as test set. Hence, a total of 165 files were used for learning set and 109 files for test set. We compared two feature sets: in the former, we use 12 values for feature “notes”, 25 values for feature “intervals”, and 208 values for Allen Forte numbers, with a total of 245 properties. In the latter, we based the classification on notes and intervals only, yielding to a set of 37 values.

In table 1, we show error rate for both feature set 1 and feature set 2. Graphs on Fig. 1 and Fig. 2 also show error rate respectively for feature set 1 and 2.

As can be argued from the results, best performance is obtained with Bayesian and SVM classifier, which brings to only a 10.1 % of error on the test, when used with feature set 1. Anyway, some interesting consideration can arise by
comparing results for both feature sets: using Allen Forte numbers let SVM to reach 0% of error rate for the training set, but 22.2% for the test set, higher than set 2, which is without Allen Forte numbers.

<table>
<thead>
<tr>
<th>% Error for feature set 1</th>
<th>Decision Tree</th>
<th>Bayesian</th>
<th>SVM</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>1.38</td>
<td>15.9</td>
<td>0</td>
<td>1.69</td>
</tr>
<tr>
<td>Test set</td>
<td>25.3</td>
<td>27.3</td>
<td>22.2</td>
<td>22.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>% Error for feature set 2</th>
<th>Decision Tree</th>
<th>Bayesian</th>
<th>SVM</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>2.07</td>
<td>7.58</td>
<td>7.58</td>
<td>0.69</td>
</tr>
<tr>
<td>Test Set</td>
<td>30.3</td>
<td>10.1</td>
<td>10.1</td>
<td>13.1</td>
</tr>
</tbody>
</table>

Table 1. Error rates for feature sets 1 and 2.

That means that this information doesn’t bring a real help to SVM, because it allows the machine to recognize exactly pieces of the training set, but it takes off generalization capabilities. On the other hand, only the decision tree classifier can really benefit by this additional information; in fact, its error rate comes down from 30.3 to 25.3.

5 Conclusion

We designed a system for automatic recognition of piano music compositional styles. We made a comparative analysis based on many feature sets and many classifiers, concluding that the best result was achieved by classifying notes and interval with Naïve Bayesian classifier or with Support Vector Machines. Error rate reaches 10.1%.

The work is open to future developments that could take into account many other features. As far as the classifier is concerned, we think that Naïve Bayesian and SVM give best generalization capabilities. Moreover, new styles will be taken into account, as the dodecaphony, which, just oppositely to baroque and classical styles, pursues a perfect parity in the utilization rate of the 12 degrees of the tempered scale.

References: