DEPARTMENT OF ENGINEERING MANAGEMENT

Looking into the minds of Bach, Haydn and Beethoven: Classification and generation of composer-specific music

Dorien Herremans, David Martens & Kenneth Sörensen

UNIVERSITY OF ANTWERP
Faculty of Applied Economics

City Campus
Prinsstraat 13, B.226
B-2000 Antwerp
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
www.uantwerpen.be

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENGINEERING MANAGEMENT

Looking into the minds of Bach, Haydn and Beethoven: Classification and generation of composer-specific music

Dorien Herremans, David Martens & Kenneth Sörensen

RESEARCH PAPER 2014-001
JANUARY 2014

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@uantwerpen.be

D/2014/1169/001

Looking into the minds of Bach, Haydn and Beethoven: Classification and generation of composer-specific music

Dorien Herremans[a]*, David Martens[b] and Kenneth Sörensen[a]

[a] *ANT/OR, University of Antwerp Operations Research Group*
[b] *Applied Data Mining Research Group, University of Antwerp*
Prinsstraat 13, B-2000 Antwerp

January 8, 2014

In this paper a number of musical features are extracted from a large music database, which are consequently used to build three composer classification models. The first two models, an if-then ruleset and a decision tree, result in an understanding of the style differences between Bach, Haydn and Beethoven. The third model, a logistic regression model, gives the probability that a piece is composed by a certain composer. This model is integrated in the objective function of a previously developed variable neighborhood search algorithm that can generate counterpoint. The result is a system that can generate an endless stream of counterpoint music with *composer-specific characteristics* that sounds pleasing to the ear. This system is implemented as an Android app called FuX that can be installed on any Android phone or tablet.

1 Introduction

The task of recognizing a composer by listening to a musical fragment used to be reserved for experts in music theory. The question that is tackled in this research is "Can a computer accurately recognize who composed a musical piece?". We take a data-driven approach, by scanning a large database of existing music and develop three classification models that can accurately classify a musical piece in groups of three composers. This research builds predictive classification models that can be used both for theory-building and to calculate the probability that a piece is composed by a certain composer.

The first goal of this paper is to build a ruleset and a decision tree that gives the reader an understanding of the differences between styles of composers (Bach, Haydn and Beethoven). These models give the reader more insight into *why* a piece belongs to a certain composer. The second goal is to build a classification model that can help an existing music composition algorithm generate composer-specific music. In previous papers, the authors developed a variable

*Corresponding author. Email: dorien.herremans@uantwerpen.be

neighborhood search algorithm (VNS) that can compose counterpoint music (Herremans and Sörensen, 2012, 2013a). The logistic regression model developed in this paper is incorporated into the objective function of the VNS. The resulting system is able to play a stream of continuously generated counterpoint music with composer-specific traits.

2 Prior work

The digitization of the music industry has attracted growing attention to the field of Music Information Retrieval (MIR). MIR is a multidisciplinary domain, concerned with retrieving and analyzing multifaceted information from large music databases (Downie, 2003). According to Byrd and Crawford (2002) the first publication about MIR originates from the mid-1960s (Kassler, 1966). Kassler (1966) uses the term MIR to name the programming language he developed to extract information from music files. In recent years, numerous MIR systems have been developed and applied to a broad range of topics. An interesting example is the content-based music search engine "Query by Humming" (Ghias et al., 1995). This MIR system allows the user to find a song based on a tune that he or she hums. Another application of MIR is measuring the similarity between two musical pieces (Berenzweig et al., 2004). In this research however, the focus lies on using MIR for composer classification.

When it comes to automatic music classification, machine learning tools are used to classify musical pieces per genre (Tzanetakis and Cook, 2002), cultural origin (Whitman and Smaragdis, 2002), mood (Laurier et al., 2008) etc. While the general task of automatically classifying music per genre has recently received increasing attention, the more specific task of composer classification remains largely unexplored (Geertzen and van Zaanen, 2008).

A system to classify string quartet pieces by composer has been implemented by Kaliakatsos-Papakostas et al. (2011). In this system, the four voices of the quartets are treated as a monophonic melody, so that it can be represented through a discrete *Markov chain*. The weighted Markov chain model reaches a classification success of 59 to 88% in classifying between two composers. The Hidden Markov Models designed by Pollastri and Simoncelli (2001) for the classification of 605 monophonic themes by five composers has a lower accuracy rate. Their best result has an accuracy of 42% of successful classifications on average. However, it must be noted that this accuracy is not measured for classification between two classes like in the previous example, but classification is done with five classes or composers.

Wołkowicz et al. (2007) show that another machine learning technique, i.e. *n-grams*, can be used to classify piano files in groups of five composers. An n-gram model tries to find patterns in properties of the training data. These patterns are called n-grams, in which n is the number of symbols in a pattern. Hillewaere et al. (2010) also use n-grams and global feature models to classify string quartets for two composers (Haydn and Mozart). Their trigram approach to composer recognition of string quartets has a classification accuracy of 61.4%, for violin and viola, and 75.4% for cello.

n-grams belong to the family of grammars, a group of techniques that use a rule-based approach to specify patterns (Searls et al., 2002). Buzzanca (2002) states that the use of grammars such as n-grams for modeling music style is unsatisfying because they are vulnerable with regard to creating ad hoc rules and they can not represent ambiguity in the musical process. Buzzanca (2002) works with Palestrina style recognition, which could be considered a more general problem than composer recognition. Instead of n-grams, he implemented a *neural network* that can classify with 97% accuracy on the test set. Although it should be noted that all the pieces of the music database are heavily preprocessed and classification is only done on short "main themes". One

of the disadvantages of neural networks is that these models are in essence a black-box, as they provide a complex non-linear output score. They do not give any new music theoretical insights in the differences between two composers as they are. To generate a comprehensible model, rules could be extracted from an existing black-box neural network, using pedagogical rule extraction techniques like Trepan and G-REX (Martens et al., 2007).

Van Kranenburg and Backer (2004) apply other types of machine learning algorithms to a database of 320 pieces from the eighteenth and early nineteenth century. 20 *high-level* style markers based on properties of counterpoint are examined. The *K-means clustering* algorithm they developed shows that musical pieces of the chosen five composers do form a cluster in feature space. A *decision tree* (C4.5) and *nearest neighbor* classification algorithm show that it is possible to classify pieces with a fairly low error rate. Although the features are described in the paper, a detailed description of the models is missing.

Mearns et al. (2010) also use high-level features based on counterpoint and intervallic features to classify similar musical pieces. Their developed C4.5 *decision tree* and *naive Bayes models* correctly classified 44 out of 66 pieces with 7 groups of composers. Although the actual decision tree is not displayed in the paper it could give music theorists an insight in the differences between styles of composers.

In the next sections, a technique is described to extract useful musical features from a database. These features are then used to build three accurate classification models. In contrast to many existing studies, the models described in this research are both accurate as well as comprehensible and the full details are described in this paper. The developed models give insights into the styles of Haydn, Beethoven and Bach. In a next phase, one of the models is incorporated in a previously developed VNS algorithm that can compose music (Herremans and Sörensen, 2013a), which results in a system that is able to generate music that has characteristics of a specified composer.

3 Feature extraction

Traditionally, a difference between *symbolic MIR* and *audio MIR* is made. Symbolic music representations, such as MIDI, contain very high-level structured information about music, e.g., which note is played by which instrument. However, most existing work revolves around audio MIR, in which automatic computer audition techniques are used to extract relevant information from audio signals (Tzanetakis et al., 2003). Different features can be examined depending on the type of audio file that is being analyzed. These features can be roughly classified in three groups:

- *low-level features* extracted by automatic computer audition techniques from audio signals such as WAV files, e.g., spectral flux and zero-crossing rate (Tzanetakis et al., 2003)
- *high-level features* extracted from structured files such as MIDI, e.g., interval frequencies, instrument presence and number of voices (McKay and Fujinaga, 2006)
- *metadata* such as factual and cultural information information related to a file which can be both structured or unstructured, e.g., play list co-occurrence (Casey et al., 2008)

In this research we work with high-level features extracted from MIDI files. Symbolic files such as MIDI files are in essence the same as musical scores. They describe the start, duration, volume and instrument of each note in a musical fragment and therefore allow the extraction of

characteristics that might provide meaningful insights to music theorists. It must be noted that MIDI files do not capture the full richness of a musical performance like audio files do (Lippincott, 2002). They are, however, very suitable for the features analyzed in this research.

3.1 KernScores database

The KernScores database is a large collection of virtual music scores made available by the Center for Computer Assisted Research in the Humanities at Standford University. It holds a total of 7,866,496 notes and is available online (CCARH, 2012). This database was specifically created for computational analysis of musical scores (Sapp, 2005). The composers Bach, Beethoven and Haydn were selected for inclusion in our classification models because a large number of musical pieces is available for these three composers in the KernScores database. Having a large amount of instances available per composer allows the creation of more accurate models. 1045 musical pieces from a total of three composers were downloaded from the database. Almost all available musical pieces per composer were selected, except for a few very short fragments. An overview of the selected pieces is given in Table 1.

Table 1: Dataset

Composer	# Instances
Haydn (HA)	254
Beethoven (BE)	196
Bach (BA)	595

The KernScores database contains musical pieces in the **KERN notation, ABC notation and MIDI. For this research, the MIDI files are used as they are compatible with the feature extraction software jSymbolic. jSymbolic is a part of jMIR, a toolbox designed for automatic music classification (Mckay and Fujinaga, 2009). Van Kranenburg and Backer (2004) point out that MIDI files are the representation of a performance and are therefore not always an accurate representation of the score. It is true that MIDI files are often recorded by a human playing the score, which results in inaccurate timing. However, since the KernScore database is encoded by hand from **KERN files, it offers a reliable source of accurate MIDI files.

3.2 Implementation of feature extraction

The software used to extract the features is jSymbolic. jSymbolic is a Java based Open Source software that allows easy extraction of high-level features from MIDI files (McKay and Fujinaga, 2007). Twelve features (see Table 2) are extracted from our dataset. All of these features offer information regarding melodic intervals and pitches. They are measured as frequencies between 0 and 1.

The examined featureset is deliberately kept small to avoid overfitting the model. McKay and Fujinaga (2006) refer to the "curse of dimensionality", whereby the number of labeled training and testing samples needed increases exponentially with the number of features. Not having too many features allows a thorough testing of the model with limited instances and can thus improve the quality of the classification model.

jSymbolic outputs the extracted features of all instances in ACE XML files. These XML files are converted to the Weka ARFF format with jMIRUtilities, another tool from the JMIR toolbox (Mckay and Fujinaga, 2009). In the next sections, three classification models are developed based on the extracted data.

Table 2: Analysed features

Variable	Feature description
x_1	Chromatic Motion Frequency - Fraction of melodic intervals corresponding to a semi-tone.
x_2	Melodic Fifth Frequency
x_3	Melodic Octaves Frequency
x_4	Melodic Thirds Frequency
x_5	Most Common Melodic Interval Prevalence
x_6	Most Common Pitch Prevalence
x_7	Most Common Pitch Class Prevalence
x_8	Relative Strength of Most Common Intervals - fraction of intervals belonging to the second most common / most common intervals
x_9	Relative Strength of Top Pitch Classes
x_{10}	Relative Strength of Top Pitches
x_{11}	Repeated Notes - fraction of notes that are repeated melodically
x_{12}	Stepwise Motion Frequency

Pitch refers to an absolute pitch, e.g., C in the 7th octave.
Pitch class refers to a note without the octave, e.g., C.

4 Composer classification models

Shmueli and Koppius (2011) point out that predictive models can not only be used as practically useful classification models, but can also play a role in theory-building and testing. In this paper, models are built for two different purposes, based on the aforementioned.

The first objective of the authors is to develop a model from which insight can be gained into the characteristics of musical pieces composed by a certain composer. This resulted in a ruleset built with RIPPER and a C4.5 decision tree. The second objective is to build a predictive model that can accurately determine the probability that a musical piece belongs to a certain composer. This model is then incorporated into the existing objective function of the music generation algorithm, leading to a new objective score that allows it to automatically assess how well a generated musical piece fits into a certain composer's style. In order to accurately modify the developed model for inclusion in the objective function, not all classification models are suited. A logistic regression model is built and described in this section.

A number of machine learning methods like neural networks, Markov chains and clustering have already been implemented for musical style modeling (Dubnov et al., 2003). Since most of the research papers do not give an accurate description of the model, nor a full feature list, the existing research could only be used as inspiration for the developed models.

Based on the features extracted in the previous section, three supervised learning algorithms are applied to the dataset. Since our dataset includes labeled instances, supervised learning techniques can be used to learn a classification model based on these labeled training instances. The Open Source software Weka is used to create the classification models (Witten and Frank, 2005). Weka offers a toolbox and framework for machine learning and data mining that is recognized as a landmark system in this field (Hall et al., 2009).

In this section, three classifier models are developed with RIPPER, C4.5 and Logistic Regression. The first two models are of a more linguistic nature and therefore more comprehensible

(Martens et al., 2011). The third model is integrated in the objective function of the music generation algorithm in the next section.

Table 3: Evaluation of the three models on the test set

Method	Accuracy
RIPPER ruleset	70.7%
C4.5 Decision tree	71.8%
Logistic Regression	81.7%

4.1 Ripper if-then ruleset

The advantage of using high level musical features is that they can give useful insights in the characteristics of a composer's style. A number of techniques are available to obtain a comprehensible model from these features. Rulesets and trees can be considered as the most easy to understand classification models due to their linguistic nature (Martens, 2008). Such models can be obtained by using *rule induction* and *rule extraction* techniques. The first category simply induces rules directly from the data, whereas rule extraction techniques attempt to extract rules from a trained black-box model (Martens et al., 2007). This research focuses on using rule induction techniques to build a ruleset and a decision tree.

In this section an inductive rule learning algorithm is used to learn "if-then" rules. Rulesets have been used in other research domains to gain insight in credit scoring (Baesens et al., 2003), medical diagnosis (Kononenko, 2001), customer relationship management (Ngai et al., 2009), diagnosis of technical processes (Isermann and Balle, 1997) and more.

In order to build a ruleset for composer classification, the propositional rule learner RIPPER was used (Cohen, 1995). JRip is the Weka implementation of the "Repeated Incremental Pruning to Produce Error Reduction algorithm" (RIPPER). This algorithm uses sequential covering to generate the ruleset. It starts by learning one rule, removes the training instances that are covered by the rules, and then repeats this process (Hall et al., 2009).

Three rules were extracted by JRip, one for each composer, whereby 66% of the instances were used as the training set and the rest as test set. The rules are displayed in Table 4. This table shows that four features are used to decide if a piece is composed by Haydn, Bach or Beethoven. The "Most common melodic interval prevalence", or the frequency of the interval that is most used, occurs in all of the rules. This indicates that, for instance, Beethoven typically does not focus on using one particular interval, in contrast to Haydn or Bach, who have a higher prevalence of the most common melodic interval.

A good learning algorithm should be able to accurately predict new samples that are not in the training set. 34% of the instances were not used for training, but kept for testing. The accuracy of classification on the test set is displayed in Tables 3 and 5. The latter table shows that most of the misclassifications occur between musical pieces of Haydn and Beethoven. This could be due to the fact that the dataset was larger for Bach. A second reason could be that Haydn was once Beethoven's teacher (DeNora, 1997) therefore their styles have more in common and are harder to distinguish. While running the algorithm, the minimum total weight of the instances in a rule was deliberately set high in order to get a smaller and thus more comprehensible tree, albeit slightly less accurate. Still overall, with 70.7% correctly classified, the model developed with RIPPER is quite accurate.

Table 4: Ruleset

if (Most Common Melodic Interval Prevalence) ≤ 0.2708 and (Melodic Octaves Frequency ≥ 0.04629) **then**
    Composer = BE
else if (Most Common Melodic Interval Prevalence ≤ 0.3491) and (Most Common Pitch Prevalence ≤ 0.1102) and (Repeated Notes Frequency ≥ 0.07592) **then**
    Composer = HA
else
    Composer = BA
end if

Table 5: Confusion matrix RIPPER

a	b	c	classified as
57	10	35	a = HA
26	**22**	19	b = BE
11	3	**172**	c = BA

4.2 C4.5 decision tree

A second tree-based model is induced to get a more visual understanding of the classification process. Weka's J48 algorithm (Witten and Frank, 2005) is used to build a decision tree with the popular C4.5 algorithm (Quinlan, 1993).

A decision tree is a tree data structure that consists of decision nodes and leaves. The leaves specify the class value, in this case the composer, and the nodes specify a test of one of the features. A path from the root to a leaf of the tree can be followed based on the feature values of the particular musical piece and corresponds to a predictive rule. The class at the resulting leave indicates the predicted composer (Ruggieri, 2002). Although decision trees do not always offer the most accurate classification results, they are often useful to get an understanding of how the classification is done.

Similar to rulesets, decision trees have been applied to a broad range of topics such as medical diagnosis (Wolberg and Mangasarian, 1990), credit scoring (Hand and Henley, 1997), estimation of toxic hazards (Cramer et al., 1976), land cover mapping (Friedl and Brodley, 1997), predicting customer behaviour changes (Kim et al., 2005) and others. Much like rulesets, one of the main advantages of a decision tree model is its comprehensibility (Craven and Shavlik, 1996).

Unlike the covering algorithm implemented in the previous model, C4.5 builds trees recursively with a "divide and conquer" approach (Quinlan, 1993). This type of approach works from the top down, seeking a feature that best separates the classes, after which the tree is pruned from the leaves to the root (Wu et al., 2008).

The resulting decision tree is displayed in Figure 1. It is noticeable that the feature evaluated at the root of the tree is the same feature that occurs in all off the rules from the if-then ruleset (see Table 4). This again confirms the importance the "Most common melodic interval prevalence" feature for composer recognition. A second feature that occurs in both models is "Most common pitch prevalence", which is the frequency of the most used pitch.

Table 3 shows that the accuracy of the tree is very comparable to the accuracy of the if-then rules extracted in the previous section. Again, the comprehensibility of the model was favored above accuracy. Therefore, the minimum number of instances per leaf was kept high. The

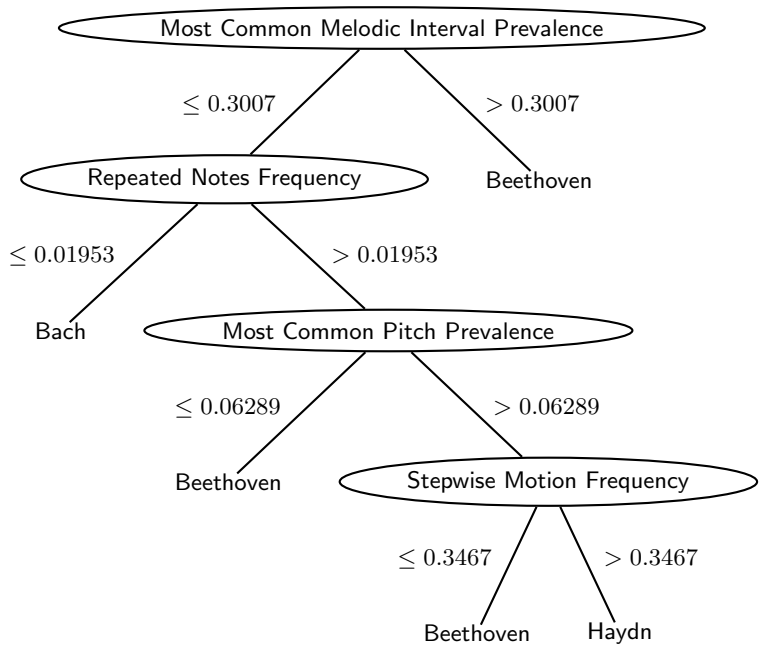


Figure 1: C4.5 decision tree

confusion matrix (see Table 6) is also comparable, with most classification errors occurring with Haydn.

Table 6: Confusion matrix C4.5

a	b	c	classified as
59	14	29	a = HA
30	**27**	10	b = BE
15	2	**169**	c = BA

4.3 Logistic regression

In the previous sections, two comprehensible models are developed. These models provide crisp classification, which means that they determine if a musical piece is either composed by a certain composer or not. They do not offer a continuous measure that indicates "how much" characteristics of a certain composer are in a piece. In this section, a scoring model is developed that can accurately describe *how well* a musical piece belongs to a composer's style.

Weka's SimpleLogistic function was used to build a logistic regression model, which was fitted using LogitBoost. The LogitBoost algorithm performs additive logistic regression (Witten and Frank, 2005). Boosting algorithms like LogitBoost sequentially apply a classification algorithm, a simple regression function in this case, to reweighted versions of training data. For many classifiers the simple boosting strategy results in dramatic performance improvements (Friedman et al., 2000).

A logistic regression model was chosen because it can indicate the statistical probability that a piece is written by a certain composer. This is useful for the inclusion in the objective function of the music generation algorithm, as described in the next section. Logistic regression models are less prone to overfitting than other models such as neural networks and require limited computing power (Tu, 1996). Logistic regression models can again be found in many areas, including the creation of habitat models for animals (Pearce and Ferrier, 2000), medical diagnosis (Kurt et al., 2008), credit scoring (Wiginton, 1980) and others.

The resulting logistic regression model for composer recognition is displayed in Equations 1 to 4. $f_{comp}(i)$ represents the probability that a piece is composed by composer i. This probability follows a logistic curve, as displayed in Figure 2. The advantage of using this model is that it outputs a number in the interval [0,1], which can easily be integrated in the objective function of the music generation algorithm to assess how well a fragment fits into a certain composer's style.

$$f_{comp}(Li) = \frac{1}{1 + e^{-L_i}} \tag{1}$$

whereby

$$\begin{aligned} L_{HA} = -3.39 &+ 21.19 \cdot x_1 + 3.96 \cdot x_2 + 6.22 \cdot x_3 + 6.29 \cdot x_4 - 4.9 \cdot x_5 \\ &- 1.39 \cdot x_6 + 3.29 \cdot x_7 - 0.17 \cdot x_8 + 0 \cdot x_9 \\ &- 0.72 \cdot x_{10} + 8.35 \cdot x_{11} - 4.21 \cdot x_{12} \end{aligned} \tag{2}$$

$$\begin{aligned} L_{BE} = 6.19 &+ 5.44 \cdot x_1 + 14.69 \cdot x_2 + 24.36 \cdot x3 - 0.45 \cdot x_4 - 6.52 \cdot x_5 \\ &- 29.99 \cdot x_6 + 3.84 \cdot x_7 - 0.38 \cdot x_8 - 3.39 \cdot x_9 \\ &- 2.76 \cdot x_{10} + 2.04 \cdot x_{11} - 0.48 \cdot x_{12} \end{aligned} \tag{3}$$

$$\begin{aligned} L_{BA} = -4.88 &- 13.15 \cdot x_1 - 6.16 \cdot x_2 - 5.28 \cdot x_3 - 11.63 \cdot x_4 + 11.92 \cdot x_5 \\ &+ 34 \cdot x_6 - 13.21 \cdot x_7 + 3.1 \cdot x_8 + 2.37 \cdot x_9 \\ &+ 0.66 \cdot x_{10} - 5.05 \cdot x_{11} + 3.03 \cdot x_{12} \end{aligned} \tag{4}$$

A musical piece is classified as being composed by composer i when it has the highest probability for that specific composer according to Equation 1 compared to the probabilities for other composers. With 81.7% correctly classified instances from the test set, the accuracy of the logistic regression model is higher than the previous models (see Table 3). This higher prediction accuracy is reflected in the confusion matrix (see Table 7).

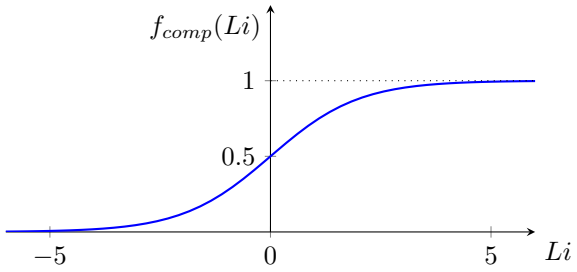


Figure 2: Probability that a piece is composed by composer i.

Table 7: Confusion matrix logistic regression

a	b	c	classified as
77	11	14	a = HA
20	**38**	9	b = BE
5	6	**175**	c = BA

5 Generating composer-specific music

In previous research, we developed a VNS algorithm that could efficiently generate music in the style of fifth species counterpoint, a type of polyphonic baroque music (Fux and Mann, 1971). In order to do this, the process of composing music was modeled as a *combinatorial optimization problem* whereby the objective is to find a musical fragment that fits the counterpoint style as well as possible. In order to evaluate how well a fragment adheres to the counterpoint style, the rules of this style were quantified to form an objective function. A detailed description of the inner workings of the VNS and the objective function is given by (Herremans and Sörensen, 2013a).

In order to generate music with composer-specific characteristics, the existing objective function for evaluating counterpoint (f_{cp}) was extended. The resulting objective function for composer i is displayed in Equation 5. When composing music in the style of composer i, the weights a_i should be set high and the others to 0. This ensures that only the counterpoint characteristics and those of composer i are taken into account. A *low* score corresponds to better counterpoint music with more influences of composer i.

$$f_i = f_{cp} + \sum_{i \in BE,BA,HA} a_i \cdot (1 - f_{comp}(L_i)) \tag{5}$$

The new model was added to the existing objective function for counterpoint in order to ensure that some basic harmonic and melodic rules are still checked. By temporarily removing the first term from Equation 5, it is quickly confirmed by listening that generating music that only adheres to the rules extracted in the previous section, without optimizing f_{cp}, does not result in musically meaningful results. The counterpoint rules are therefore necessary in order to ensure that the generated music also optimizes some basic musical properties such as "only consonant intervals are permitted". With this new objective function, the VNS algorithm is able to generate counterpoint music with characteristics of a certain composer.

5.1 Implementation - FuX

The music generation algorithm was implemented as an Android application called FuX, named after the author of the most influential book on counterpoint called "Gradus ad Parnassum" (Fux and Mann, 1971). Johann Fux was an Austrian composer and theorist that lived in the 17^{th}–18^{th} century. The FuX app is available as Open Source software in the Google Play store[1] and can be run on any Android based device. The developed app can continuously generate counterpoint music with composer-specific characteristics. This music is continuously generated while it is being played.

[1] `http://play.google.com`

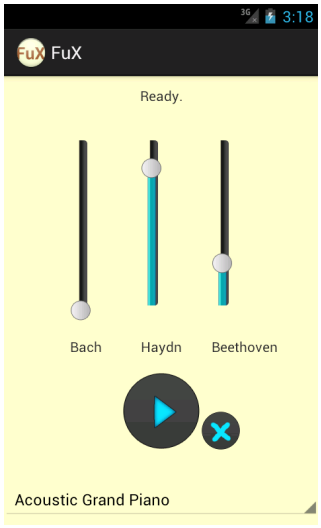


Figure 3: User interface of FuX

Android is an operating system for mobile devices based on Linux (Kernel 2.6) (Mongia and Madisetti). Applications can be run by a runtime engine–the Dalvik Virtual Machine (VM), which runs applications written in Android's variant of java (Bornstein, 2008). Since resources are typically limited on mobile devices, a careful consideration had to be made on how to implement the music generation algorithm. Instead of using the Android Software Development Kit (SDK) to develop java applications (Google, 2012), Son and Lee (2011) recommend using the Android Native Development Kit (**N**DK) for computationally expensive tasks. Benchmark experiments of (Lin et al., 2011) confirm this statement. For developing the music generation algorithm, Android NDK was used to compile C++ code for the Android platform. Java's multithreading capabilities were then used to allow continuous playback while new music is being generated. A detailed description of the implementation details of FuX 1.0 are given by Herremans and Sörensen (2013b).

The graphical user interface of FuX 2.0 is displayed in Figure 3. The three sliders (or SeekBars) give the user control over the weights a_i of the objective function (see Equation 5). This even allows a user to generate music consisting of a mix of multiple composers if he or she wishes to do so. The playback instrument can be chosen and dynamically changed by the user.

5.2 Results

The resulting composer-specific music generation algorithm was tested on an Eclipse Android Virtual Device with Android 4.0.3, ARM processor and 512MB RAM. The emulator was installed on an OpenSuse system with Intel®Core^{TM}2 Duo CPU@ 2.20GHz and 3.8GB RAM. Figure 4 displays the evolution of the solution quality over time. The left plots describe the generation of the cantus firmus or base line. The plots on the right hand side describe the evolution of the score of the counterpoint line or top line. In the experiment 16 measures are generated with a cutoff time of 12 seconds. FuX first generates the base line and continues with the top line. Since the main objective is to produce music with composer-specific characteristics, the weight of the respective composer's score is set very high (100). Figure 4 shows a drastic improvement of the selected composer's score for each of the three composers. This means that the generated music actually contains composer-specific elements according to the logistic regression model that

was built. When optimizing for a specific composer no real change in the scores for the other composers can be noted. The improvements of the counterpoint score are not very high, but are enough to add basic musical properties to the fragment.

Due to large differences in computing power between different Android devices, the quality of the generated music is highly dependent on the architecture of the mobile device on which it is run. Still, the subjective opinion of the authors is that the generated stream of music sounds pleasant to the ear, even on relatively modest hardware. The reader is invited to install the app and listen to the resulting music.

6 Conclusions

A number of musical features were extracted from a large database of music. Based on these features three classification models models were built. The first two models, an if-then ruleset and a decision tree, give the user more insight and understanding in the musical style of a composer, e.g. "Beethoven typically does not focus on using one particular interval, in contrast to Haydn or Bach, who have a higher prevalence of the most common melodic interval". The third model, a logistic regression, can accurately classify musical pieces from Haydn, Beethoven and Bach. This model is integrated in the objective function of a variable neighborhood search algorithm that can efficiently generate counterpoint music. The resulting algorithm was implemented as a user friendly Android app called FuX, which is able to play a stream of counterpoint music with composer-specific characteristics that sounds pleasing to the ear.

Future extensions of this research include working with more musical styles, voices and adding a recurring theme to the music. FuX might also be ported to other platforms, such as iOS from Apple.

References

Baesens, B., Setiono, R., Mues, C., Vanthienen, J., 2003. Using neural network rule extraction and decision tables for credit-risk evaluation. Management Science 49, 312–329.

Berenzweig, A., Logan, B., Ellis, D., Whitman, B., 2004. A large-scale evaluation of acoustic and subjective music-similarity measures. Computer Music Journal 28, 63–76.

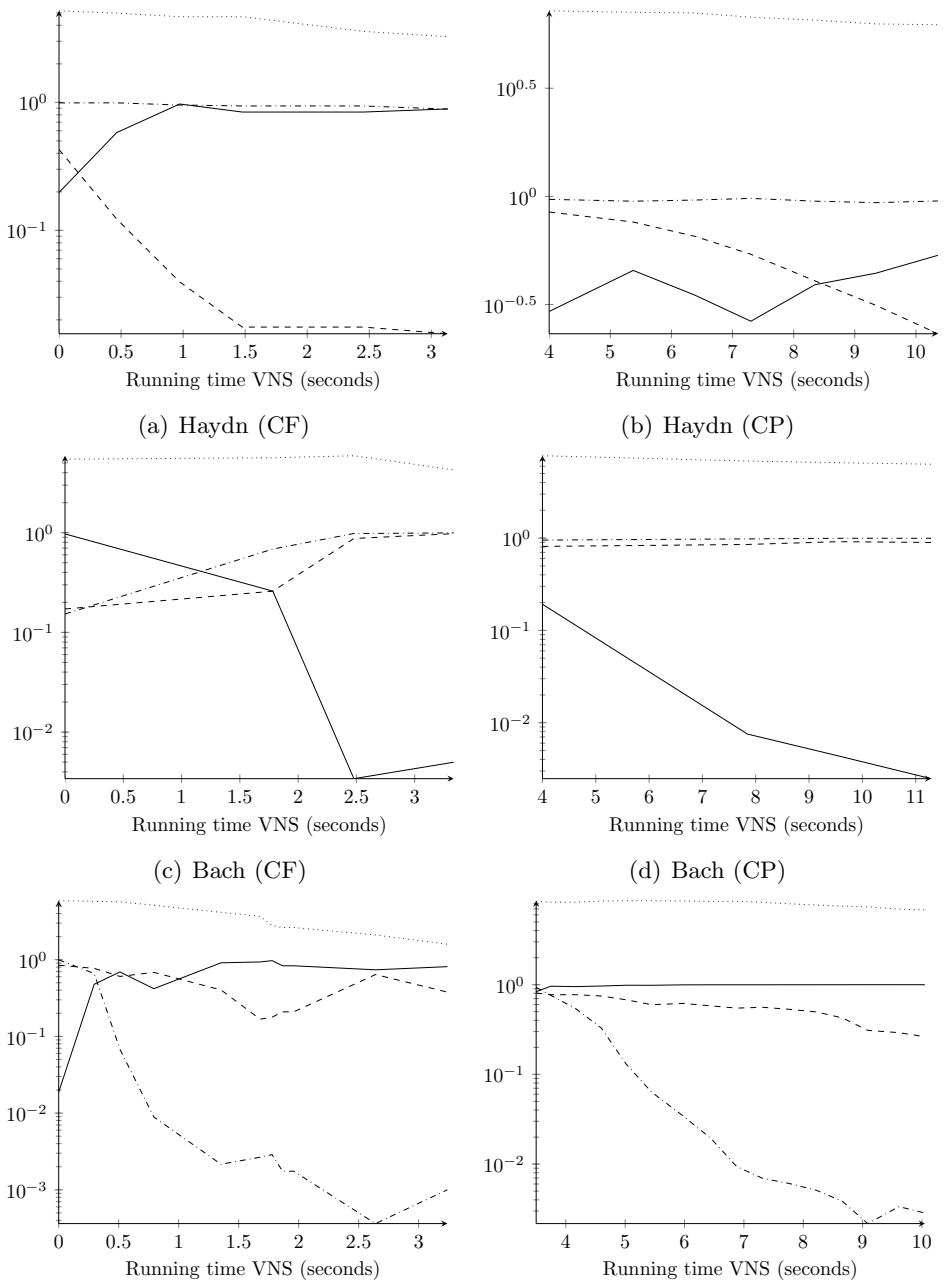
Bornstein, D., 2008. Dalvik vm internals, in: Google I/O Developer Conference, pp. 17–30.

Buzzanca, G., 2002. A supervised learning approach to musical style recognition, in: Music and Artificial Intelligence. Additional Proceedings of the Second International Conference, ICMAI, Citeseer. p. 167.

Byrd, D., Crawford, T., 2002. Problems of music information retrieval in the real world. Information Processing & Management 38, 249–272.

Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., Slaney, M., 2008. Content-based music information retrieval: current directions and future challenges. Proceedings of the IEEE 96, 668–696.

CCARH, 2012. Kernscores, `http://kern.ccarh.org`.

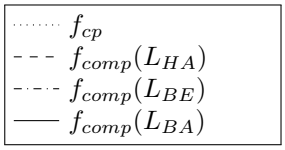


(a) Haydn (CF) (b) Haydn (CP)

(c) Bach (CF) (d) Bach (CP)

(e) Beethoven (CF) (f) Beethoven (CP)

Figure 4: Evolution of solution quality over time



Cohen, W., 1995. Fast effective rule induction, in: Prieditis, A., Russell, S. (Eds.), Proceedings of the 12th International Conference on Machine Learning, Morgan Kaufmann Publishers, Tahoe City, CA. pp. 115–123.

Cramer, G., Ford, R., Hall, R., 1976. Estimation of toxic hazard—a decision tree approach. Food and cosmetics toxicology 16, 255–276.

Craven, M., Shavlik, J., 1996. Extracting tree-structured representations of trained networks. Advances in neural information processing systems , 24–30.

DeNora, T., 1997. Beethoven and the construction of genius: Musical politics in Vienna, 1792-1803. University of California Press.

Downie, J., 2003. Music information retrieval. Annual review of information science and technology 37, 295–340.

Dubnov, S., Assayag, G., Lartillot, O., Bejerano, G., 2003. Using machine-learning methods for musical style modeling. Computer 36, 73–80.

Friedl, M., Brodley, C., 1997. Decision tree classification of land cover from remotely sensed data. Remote sensing of environment 61, 399–409.

Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics 28, 337–407.

Fux, J., Mann, A., 1971. The study of counterpoint from Johann Joseph Fux's Gradus Ad Parnassum - 1725. Norton, New York.

Geertzen, J., van Zaanen, M., 2008. Composer classification using grammatical inference, in: Proceedings of the MML 2008 International Workshop on Machine Learning and Music held in conjunction with ICML/COLT/UAI, pp. 17–18.

Ghias, A., Logan, J., Chamberlin, D., Smith, B., 1995. Query by humming: musical information retrieval in an audio database, in: Proceedings of the third ACM international conference on Multimedia, ACM. pp. 231–236.

Google, 2012. Google Android SDK, `http://developer.android.com/sdk/index.html`.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I., 2009. The weka data mining software: an update. ACM SIGKDD Explorations Newsletter 11, 10–18.

Hand, D., Henley, W., 1997. Statistical classification methods in consumer credit scoring: a review. Journal of the Royal Statistical Society: Series A (Statistics in Society) 160, 523–541.

Herremans, D., Sörensen, K., 2012. Composing first species counterpoint with a variable neighbourhood search algorithm. Journal of Mathematics and the Arts 6, 169–189.

Herremans, D., Sörensen, K., 2013a. Composing fifth species counterpoint music with a variable neighborhood search algorithm. Expert Systems with Applications 40, 6427–6437.

Herremans, D., Sörensen, K., 2013b. Fux, an android app that generates counterpoint, in: Computational Intelligence for Creativity and Affective Computing (CICAC), 2013 IEEE Symposium on, pp. 48–55.

Hillewaere, R., Manderick, B., Conklin, D., 2010. String quartet classification with monophonic models. Proceedings of ISMIR, Utrecht, the Netherlands .

Isermann, R., Balle, P., 1997. Trends in the application of model-based fault detection and diagnosis of technical processes. Control engineering practice 5, 709–719.

Kaliakatsos-Papakostas, M., Epitropakis, M., Vrahatis, M., 2011. Weighted markov chain model for musical composer identification. Applications of Evolutionary Computation , 334–343.

Kassler, M., 1966. Toward musical information retrieval. Perspectives of New Music 4, 59–67.

Kim, J., Song, H., Kim, T., Kim, H., 2005. Detecting the change of customer behavior based on decision tree analysis. Expert Systems 22, 193–205.

Kononenko, I., 2001. Machine learning for medical diagnosis: history, state of the art and perspective. Artificial Intelligence in medicine 23, 89–109.

Kurt, I., Ture, M., Kurum, A., 2008. Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease. Expert Systems with Applications 34, 366–374.

Laurier, C., Grivolla, J., Herrera, P., 2008. Multimodal music mood classification using audio and lyrics, in: Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on, IEEE. pp. 688–693.

Lin, C., Lin, J., Dow, C., Wen, C., 2011. Benchmark dalvik and native code for android system, in: Innovations in Bio-inspired Computing and Applications (IBICA), 2011 Second International Conference on, IEEE. pp. 320–323.

Lippincott, A., 2002. Issues in content-based music information retrieval. Journal of Information Science 28, 137–142.

Martens, D., 2008. Building acceptable classification models for financial engineering applications. SIGKDD Explorations 10, 30–31.

Martens, D., Baesens, B., Van Gestel, T., Vanthienen, J., 2007. Comprehensible credit scoring models using rule extraction from support vector machines. European Journal of Operational Research 183, 1466–1476.

Martens, D., Vanthienen, J., Verbeke, W., Baesens, B., 2011. Performance of classification models from a user perspective. Decision Support Systems 51, 782–793.

McKay, C., Fujinaga, I., 2006. jsymbolic: A feature extractor for midi files, in: Proceedings of the International Computer Music Conference, pp. 302–5.

McKay, C., Fujinaga, I., 2007. Style-independent computer-assisted exploratory analysis of large music collections. Journal of Interdisciplinary Music Studies 1, 63–85.

Mckay, C., Fujinaga, I., 2009. jmir: Tools for automatic music classification, in: Proceedings of the International Computer Music Conference, Citeseer. pp. 65–8.

Mearns, L., Tidhar, D., Dixon, S., 2010. Characterisation of composer style using high-level musical features, in: Proceedings of 3rd international workshop on Machine learning and music, ACM. pp. 37–40.

Mongia, B., Madisetti, V., . Reliable real-time applications on android os. IEEE Electrical and Computer Engineering Electrical and Computer Engineering .

Ngai, E., Xiu, L., Chau, D., 2009. Application of data mining techniques in customer relationship management: A literature review and classification. Expert Systems with Applications 36, 2592–2602.

Pearce, J., Ferrier, S., 2000. Evaluating the predictive performance of habitat models developed using logistic regression. Ecological modelling 133, 225–245.

Pollastri, E., Simoncelli, G., 2001. Classification of melodies by composer with hidden markov models, in: Web Delivering of Music, 2001. Proceedings. First International Conference on, IEEE. pp. 88–95.

Quinlan, J., 1993. C4. 5: programs for machine learning. volume 1. Morgan kaufmann.

Ruggieri, S., 2002. Efficient c4. 5 [classification algorithm]. Knowledge and Data Engineering, IEEE Transactions on 14, 438–444.

Sapp, C., 2005. Online database of scores in the humdrum file format, in: Intl. Symp. on Music Information Retrievel, pp. 664–665.

Searls, D., et al., 2002. The language of genes. Nature 420, 211–217.

Shmueli, G., Koppius, O., 2011. Predictive analytics in information systems research. MIS Quarterly 35, 553–572.

Son, K., Lee, J., 2011. The method of android application speed up by using ndk, in: Awareness Science and Technology (iCAST), 2011 3rd International Conference on, IEEE. pp. 382–385.

Tu, J., 1996. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. Journal of clinical epidemiology 49, 1225–1231.

Tzanetakis, G., Cook, P., 2002. Musical genre classification of audio signals. Speech and Audio Processing, IEEE transactions on 10, 293–302.

Tzanetakis, G., Ermolinskyi, A., Cook, P., 2003. Pitch histograms in audio and symbolic music information retrieval. Journal of New Music Research 32, 143–152.

Van Kranenburg, P., Backer, E., 2004. Musical style recognition-a quantitative approach, in: Proceedings of the Conference on Interdisciplinary Musicology (CIM), pp. 106–107.

Whitman, B., Smaragdis, P., 2002. Combining musical and cultural features for intelligent style detection, in: Proc. Int. Symposium on Music Inform. Retriev.(ISMIR), pp. 47–52.

Wiginton, J., 1980. A note on the comparison of logit and discriminant models of consumer credit behavior. Journal of Financial and Quantitative Analysis 15, 757–770.

Witten, I., Frank, E., 2005. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

Wolberg, W., Mangasarian, O., 1990. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proceedings of the national academy of sciences 87, 9193–9196.

Wołkowicz, J., Kulka, Z., Keselj, V., 2007. N-gram-based approach to composer recognition. Ph.D. thesis. M. Sc. Thesis. Warsaw University of Technology.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., et al., 2008. Top 10 algorithms in data mining. Knowledge and Information Systems 14, 1–37.