Classification and generation of composer-specific music using global feature models and variable neighborhood search

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

[a] ANT/OR – UA Operations Research Group

[b] Applied Data Mining Research Group

University of Antwerp

Prinsstraat 13, 2000 Antwerp, Belgium

dorien.herremans@uantwerpen.be

kenneth.sorensen@uantwerpen.be

david.martens@uantwerpen.be

Abstract

In this paper a number of musical features are extracted from a large music database, which are consequently used to build four 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 other two models, a logistic regression model and a support vector machine classifier, are more accurate. The probability of a piece being composed by a certain composer given by the logistic regression 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 contrapuntal 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.

Introduction

The task of recognizing a composer by listening to a musical fragment used to be reserved for those who are well versed 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 four 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 more accurate classification models that can help an existing music composition algorithm generate composer-specific music, i.e., music that contains characteristics of a specific composer. In previous papers, the authors developed a variable neighborhood search algorithm (VNS) that can compose contrapuntal music (Herremans and Sörensen 2012, 2013). 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 contrapuntal music with composer-specific traits.

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; Conklin 2013), cultural origin (Whitman and Smaragdis 2002), mood (Laurier et al. 2008), hit ranking (Herremans et al. 2014a) etc. The general task of automatically classifying music per genre has received a lot of attention, see Conklin (2013) for a more complete overview. The more specific task of composer classification has remained largely unexplored in the past (Geertzen and van Zaanen 2008), yet it has gained more attention in the last decade.

The studies below usually list the accuracy rates as a performance measure. It should, however, be noted that accuracy is not always the best performance measure, for instance in the case of an unbalanced dataset. The receiver operating characteristic (ROC) offers a more correct measure and is therefore used to evaluate the performance of the models in this research. This metric takes into account the true positives versus the false positives, which makes it more suited when the dataset is slightly skewed (see Tab. 1) (Fawcett 2004). When evaluating the models listed below, one should take into account that accuracy is not always comparable, depending on the characteristics of the dataset.

In 1958, Youngblood (1958) was one of the first to tackle the composer classification problem with measures from information theory. He manually studied features such as entropy, tonal frequencies and transition probabilities for pieces from Schubert, Mendelssohn and Schumann. These days computers offer researchers the power to create accurate and complex classification models. 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. The accuracies given in this section should not be treated as be-all and end-all, since they depend greatly on the chosen dataset (i.e., balanced; how many and which composers, etc).

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. Manaris et al. (2005) use artificial neural networks to distinguish five composers from various genres with 93.6 to 95%. Their model is based on 20 simple global metrics based on Zipf's law. 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. The system developed by Dor and Reich (2011) uses a number of global characteristics to build C4.5, naive Bayes, Random Forest, SimpleLogistic, support vector machines and RIPPER classifiers. The system reaches an accuracy of 75% when classifying keyboard scores between Mozart and Haydn. For composer-pairs that are more easy to differentiate, such as Mozart and Joplin, the algorithm reaches accuracies as high as 99%.

In the next sections, a technique is described to extract useful musical features from a database. These features are then used to build four 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 2013), which results in a system that is able to generate music that has characteristics of a specified composer.

Feature extraction

Traditionally, a distinction 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 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)

It is not an simple task to extract note information from audio recordings of polyphonic music (Gómez Gutiérrez 2006). Since the high-level features used in this research require detailed note information, we chose to work with MIDI files. Symbolic files such as MIDI files are more comparable to 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.

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 Stanford University (CCARH). 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 Johann Sebastian Bach, Ludwig van Beethoven and Franz Joseph 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.

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 occurrence frequencies normalized to range from 0 to 1.

Table 2. Analyzed 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 melodic 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.

The examined featureset is deliberately kept small to avoid overfitting the model (Gheyas and Smith 2010). 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 (McKay and Fujinaga 2006). In this research, a selection of features was made from the 111 features available in jSymbolic. During this selection process, one dimensional features that output frequency information related to intervals or pitches were preferred because of their normalized nature and ease to handle. All features dependent upon the key of the piece or nominal features were omitted. Features related to instruments, such as 'Electric guitar fraction', were omitted since they are not relevant for the chosen corpus. Rhythmic features were not used because the music generation algorithm currently does not make changes in the rhythm. Since evaluating how much of a particular composer's influence a generated piece has is

one of the goals of the developed models, these features were not included. The resulting 12 features are displayed in Table 2. Some preliminary experiments were performed with automatic feature selection on this limited dataset, yet they did not yield any improvements to the results displayed in this paper.

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, four classification models are developed based on the extracted data.

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 the research described in this article, models were built for two different purposes, reflecting this distinction.

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 the "Repeated Incremental Pruning to Produce Error Reduction algorithm" (RIPPER) and a C4.5 decision tree. The second objective is to build predictive models (logistic regression and support vector machines) that can accurately determine the probability that a musical piece belongs to a certain composer. One of these models is then incorporated into the existing objective function of the music generation algorithm, leading to a new metric 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 was chosen for this purpose. Its

implementation is described in the section "Generating composer-specific music".

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, four 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, four classifier models are developed with RIPPER, C4.5, logistic regression and support vector machines. The first two models are of a more linguistic nature and therefore more comprehensible (Martens et al. 2011). The other two models are not as comprehensible, but have a better performance. One of these latter models is integrated in the objective function of the music generation algorithm in the next section. The performance results based on accuracy and area under the curve (AUC) of all four models are displayed in Table 3. Although the distribution is not heavily skewed (see Table 1), it is not completely balanced either. Because of this the use of the accuracy measure to evaluate our results is not suited and the weighted AUC per class size (wAUC) was used instead (Fawcett 2004), yet both are displayed in Table 3 to be complete.

Each algorithm was evaluated with stratified 10-fold cross validation (10CV). During the cross validation procedure, the dataset is divided into 10 folds. 9 of them are used for model building and 1 for testing. This procedure is repeated 10 times. The displayed AUC and accuracy are the average results over the 10 test sets. The resulting model is built on the entire dataset and can be expected to have a performance which is at least as good as the 10CV performance. A Wilcoxon signed-rank test is conducted to compare the performance of the models with the best performing model. The null hypothesis of this test states: “"There is no difference between the performance of a model and that of the best model”.

Table 3. Evaluation of the models with 10-fold cross-validation

Method	Accuracy	wAUC
RIPPER ruleset	*77%* (3.82)	*82%* (3.51)
C4.5 Decision tree	*79%* (2.97)	*88%* (2.55)
Logistic regression	*83%* (3.27)	94% (2.12)
Support vector machines	**86%** (4.15)	**96%** (2.12)

$p < 0.01$: italic, $p > 0.05$: bold, best: bold.
Standard deviations are shown between parentheses.

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 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).

Five rules were extracted by JRip, one for each composer, with 10-fold cross-validation. The rules are displayed in Figure 1. This figure shows that seven different features are used to decide if a piece is composed by Haydn, Bach or Beethoven. The "Most common melodic interval prevalence", or the occurrence frequency of the interval that is most used, is present in most of the rules. This indicates that, for instance, Beethoven typically does not focus on using one particular interval, in contrast to Haydn or Bach, for whom the prevalence of the most common melodic interval is not as restrictive.

The AUC value weighted by class size and accuracy for each technique are discussed, as well as true positive (TP) rate, false negative (FN) rate, recall, precision and AUC per composer. Precision measures the accuracy provided that a specified class has been predicted (positive predictive value). Recall is a measure of the ability of the model to correctly classify instances of a certain class (sensitivity) (Tan et al. 2007).

A good learning algorithm should be able to accurately predict new samples that

if (Most Common Melodic Interval Prevalence) $\leq$ 0.2688 and (Melodic Octaves Frequency $\geq$ 0.06399) **then**
    Composer = BE
else if (Most Common Melodic Interval Prevalence $\leq$ 0.2823) and (Most Common Pitch Prevalence $\leq$ 0.07051) and (Melodic Octaves Frequency $\geq$ 0.02489) and (Relative Strength of Top Pitches $\leq$ 0.9754) **then**
    Composer = BE
else if (Most Common Melodic Interval Prevalence $\leq$ 0.328) and (Repeated Notes Frequency $\geq$ 0.07592) and (Most Common Pitch Prevalence $\leq$ 0.1076) **then**
    Composer = HA
else if (Stepwise Motion Frequency $\leq$ 0.5732) and (Chromatic Motion Frequency $\geq$ 0.1166) and (Repeated Notes Frequency $\geq$ 0.3007) **then**
    Composer = HA
else
    Composer = BA
end if

Figure 1. Ruleset

are not in the training set. The accuracy of classification and AUC with 10-fold cross-validation are displayed in Table 3. The confusion matrix is displayed in Table 4. The latter table shows that least confusion occurs between Bach and Beethoven. The relatively higher misclassification rate between Haydn and Beethoven; and Haydn and Bach could be due to the fact that the dataset was larger for Bach and Haydn. A second reason could be that Haydn and Beethoven's styles are indeed more similar, as suggested by the greater amount of chronological and geographical overlap between their lives, and by the fact that Haydn was once Beethoven's teacher (DeNora 1997). The timeline in Figure 2 shows they lived more in the same time period, just like Haydn and Bach. As for geographical proximity, Bach spent most of his life in North-East Germany (Leipzig, Köthen, Weimar) compared to Beethoven who moved from West-Germany (Bohn, Cologne) to Austria (Vienna), Haydn's home country (Greene 1985). While running the algorithm, the minimum number of instances in a rule was deliberately set high in order to get a smaller and thus more comprehensible tree, albeit slightly less accurate. Table 5 displays more detailed results per composer. It is noticeable to the recall and precision

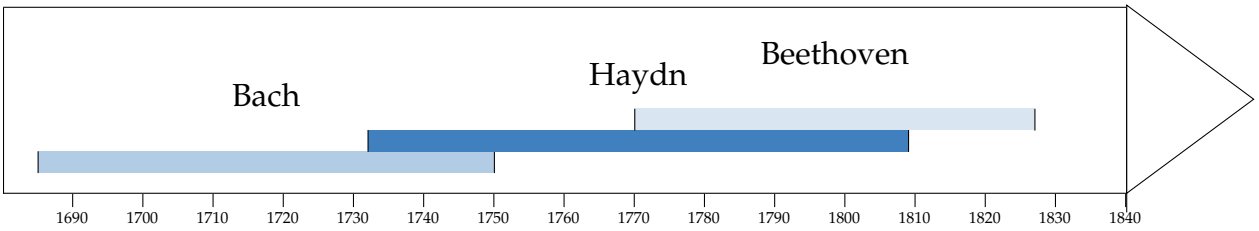


Figure 2. Timeline of the composers Bach, Beethoven and Haydn (Greene 1985)

rates are much higher for Bach, this is possibly due to the higher amount of data available for Bach in the dataset. Still overall, with 77% correctly classified and a weighted average AUC of 82%, the model developed with RIPPER is reasonably good.

Table 4. Confusion matrix for RIPPER

a	b	c	classified as
145	42	67	a = HA
41	**110**	45	b = BE
25	17	**553**	c = BA

Table 5. Detailed results for RIPPER per composer

Composer	TP Rate	FP Rate	Precision	Recall	AUC
HA	0.57	0.08	0.687	0.57	79%
BE	0.56	0.07	0.65	0.56	82%
BA	0.93	0.25	0.83	0.93	86%

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 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 behavior 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 3. All four features from this tree model also occur in the ruleset (Figure 3). It is noticeable that the feature evaluated at the root of the tree is the same feature that occurs in many of the rules from the if-then ruleset (see Figure 1). The importance of the "Most common melodic interval prevalence" feature for composer recognition is again confirmed, as it is the root node of the tree model. The "melodic octaves frequency" feature indicates that Bach uses more octaves then haydn. Bach also seems to use less repeated notes.

Table 3 shows that the accuracy (79%) and weighted average AUC (88%) values of the tree are very comparable to those of the if-then rules extracted in the previous

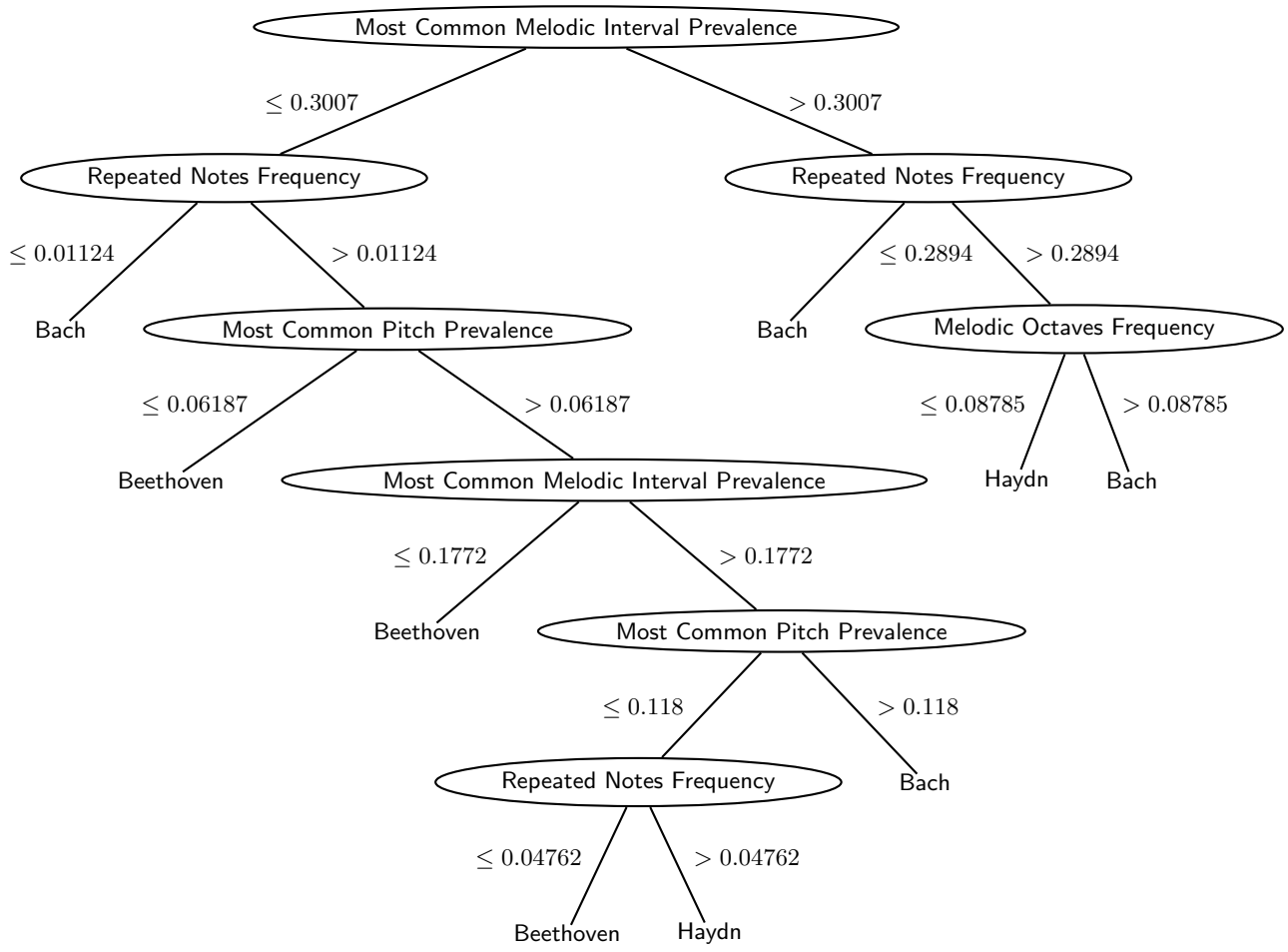


Figure 3. C4.5 decision tree

section. Again, the comprehensibility of the model was favored above accuracy. Therefore, the minimum number of instances per leaf was kept high. The confusion matrix (see Table 6) is also comparable, with most classification errors occurring between Haydn and Beethoven. The least confusion can be seen between Beethoven and Bach, as in the previous model. Similar to previously discussed model the precision and recall rates in Table 7 are much higher for Bach.

Table 6. Confusion matrix for C4.5

a	b	c	classified as
174	31	49	a = HA
63	**106**	27	b = BE
40	14	**541**	c = BA

Table 7. Detailed results for C4.5 per composer

Composer	TP Rate	FP Rate	Precision	Recall	AUC
HA	0.68	0.13	0.63	0.69	83%
BE	0.54	0.05	0.70	0.54	88%
BA	0.91	0.17	0.88	0.91	91%

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 (LR), 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 this 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 4. 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

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

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

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

Whereby x_i refers to the corresponding feature value from Table 2.

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. A coefficient with a high absolute value indicates a features that is important for distinguishing a particular composer. For example, x_5 (most common melodic interval frequency) has a high coefficient value, especially for BA. This feature is also at the top of the decision tree (Figure 3) and occurs in almost all of the rules from the ruleset (Figure 1). In Table 9 an improvement can be seen compared to the two previous models. The results for the composers which are less represented in the dataset (Beethoven and Haydn) are much improved. All of the individual AUC values are now over 91%. Logistic regression is a stronger classification model than both the previous models, which is reflected in its ability to get more accurate results for classes with fewer data.

With 83% correctly classified instances from the test set and an AUC value of 94%,

the logistic regression model outperforms the previous models (see Table 3). This higher prediction accuracy is reflected in the confusion matrix (see Table 8). The average probability of the misclassified pieces is 64%. Examples of misclassified pieces include the Brandenburg Concerto No. 5 in D major, BWV 1050, Mvmt. 1 from Bach, which is classified as Haydn with a probability of 37% and String Quartet No. 9 in C major, Op. 59, No. 3, Allegro molto from Beethoven, which is classified as Haydn with a probability of 4%.

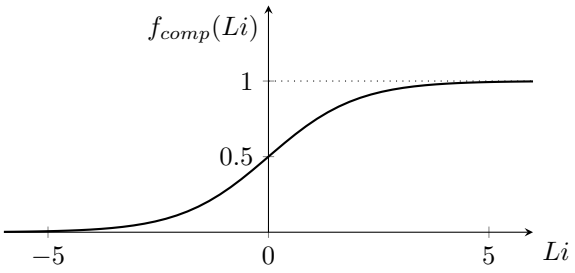


Figure 4. Probability that a piece is composed by composer i.

Table 8. Confusion matrix for logistic regression

a	b	c	classified as
190	30	34	a = HA
57	**119**	20	b = BE
25	15	**555**	c = BA

Table 9. Detailed results for logistic regression per composer

Composer	TP Rate	FP Rate	Precision	Recall	AUC
HA	0.75	0.10	0.70	0.75	92%
BE	0.61	0.05	0.73	0.61	91%
BA	0.93	0.12	0.91	0.93	96%

Support Vector Machines

In this section, LibSVM was used to build a support vector machine (SVM) classifier (Chang and Lin 2011). The support vector machine is a learning procedure

based on the statistical learning theory (Vapnik 1995). This method has been applied successfully in many areas including stock market prediction (Huang et al. 2005), text classification (Tong and Koller 2002), gene selection (Guyon et al. 2002) and others. Given a training set of N data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1, +1\}$, the SVM classifier should fulfil the following conditions (Cristianini and Shawe-Taylor 2000; Vapnik 1995):

$$\begin{cases} \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b \geq +1, & \text{if} \quad y_i = +1 \\ \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b \leq -1, & \text{if} \quad y_i = -1 \end{cases} \tag{5}$$

which is equivalent to

$$y_i[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b] \geq 1, \quad i = 1, ..., N. \tag{6}$$

The non-linear function $\boldsymbol{\varphi}(\cdot)$ maps the input space to a high (possibly infinite) dimensional feature space. In this feature space, the above inequalities basically construct a hyperplane $\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) + b = 0$ discriminating between the two classes (see Figure 5). By minimizing $\mathbf{w}^T \mathbf{w}$, the margin between both classes is maximized.

In primal weight space the classifier then takes the form

$$y(\mathbf{x}) = \text{sign}[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) + b], \tag{7}$$

but, on the other hand, is never evaluated in this form. One defines the convex optimization problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{J}(\mathbf{w}, b, \boldsymbol{\xi}) = \tfrac{1}{2}\mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{N} \xi_i \tag{8}$$

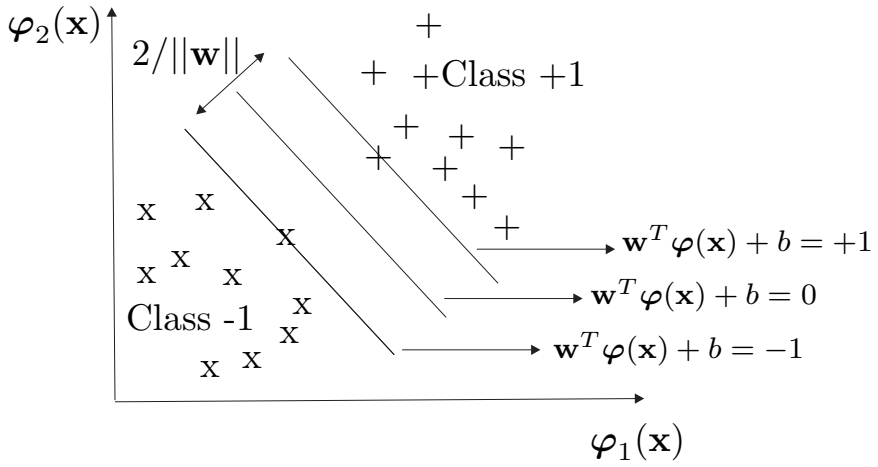


Figure 5. Illustration of SVM optimization of the margin in the feature space.

subject to

$$\begin{cases} y_i[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b] \geq 1 - \xi_i, & i = 1, ..., N \\ \xi_i \geq 0, & i = 1, ..., N. \end{cases} \tag{9}$$

The variables ξ_i are slack variables which are needed to allow misclassifications in the set of inequalities (e.g., due to overlapping distributions). The first part of the objective function tries to maximize the margin between both classes in the feature space and is a regularisation mechanism that penalizes for large weights, whereas the second part minimizes the misclassification error. The regularisation coefficient C, a positive real constant, should be considered as a tuning parameter in the algorithm. This leads to the following classifier (Cristianini and Shawe-Taylor 2000):

$$y(\mathbf{x}) = \text{sign}[\textstyle\sum_{i=1}^{N} \alpha_i \, y_i \, K(\mathbf{x}_i, \mathbf{x}) + b], \tag{10}$$

whereby $K(\mathbf{x}_i, \mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x}_i)^T\boldsymbol{\varphi}(\mathbf{x})$ is taken with a positive definite kernel satisfying the Mercer theorem. The Lagrange multipliers α_i are then determined by optimizing the dual dual problem. In this research, the Radial Basis Function (RBF) kernel was used to

map the feature space to a hyperplane:

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\{-\|\mathbf{x} - \mathbf{x}_i\|^2/\sigma^2\}, \quad (\text{RBF kernel})$$

where d, c and σ are constants.

A hyperparameter optimization procedure was conducted with GridSearch in Weka to determine the optimal setting for the regularization parameter C (0.0001, 0.001,...10,000) and the σ for the RBF kernel (σ = 0.1, 1,...100,000). The choice of hyperparameters to test was inspired by settings suggesting by Weka (2013a). The Weka implementation of GridSearch performs 2-fold cross validation on the initial grid. This grid is determined by the two input parameters (C and σ for the RBF kernel). 10-fold cross validation is then performed on the best point of the grid based on the weighted AUC by class size and its adjacent points. If a better pair is found, the procedure is repeated on its neighbors until no better pair is found or the border of the grid is reached (Weka 2013b).

The SVM classifier with non-linear kernel is a complex, non-linear function. Trying to comprehend the logic of the classifications made is quite difficult, if not impossible (Martens et al. 2009; Martens and Provost 2014). The resulting accuracy is 86% and the weighted AUC-value is 96% for the SVM with RBF kernel (see Table 3). The confusion matrix (Table 10) confirms that SVM is the best model for classifying between Haydn, Beethoven and Bach. Most misclassification occurs between Haydn and Beethoven, which can be explained by the geographical and temporal overlap between the lives of these composers as mentioned in the section "Ripper if-then ruleset". Table 11 reveals that Beethoven has a high AUC, however, the recall and true positive rate are low and the precision rate is higher. This means that pieces which are predicted as being composed by Beethoven have a high accuracy, however, the model will be

conservative in assigning this label. The same pattern can be observed in the previous models and seems to be typical when classifying pieces of Beethoven.

Table 10. Confusion matrix for support vector machines

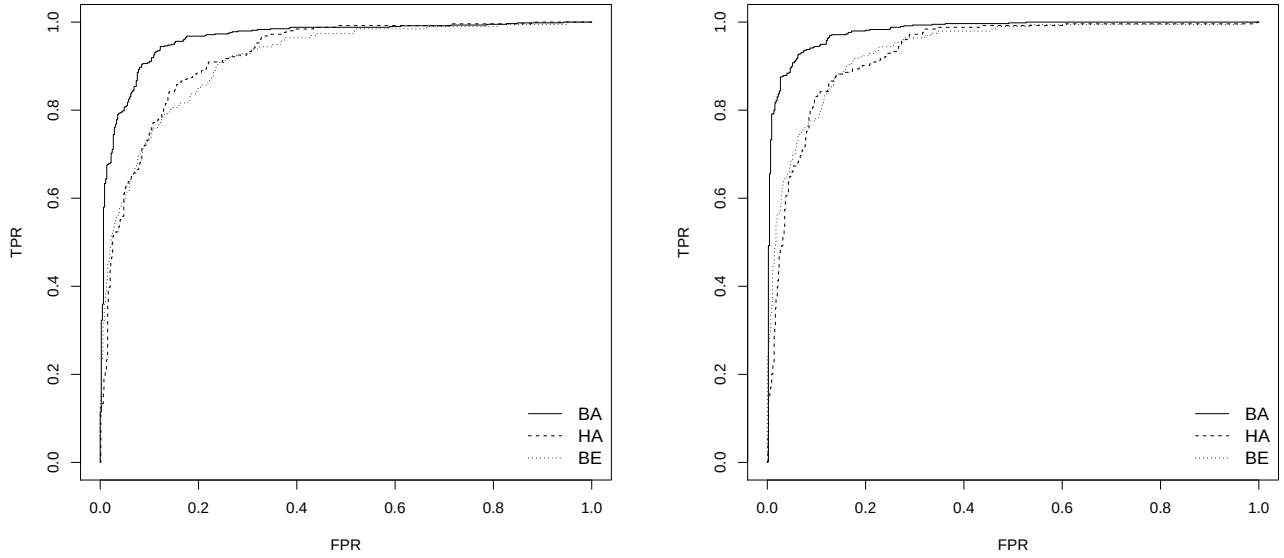
a	b	c	classified as
204	26	24	a = HA
49	**127**	20	b = BE
22	10	**563**	c = BA

Table 11. Detailed results for support vector machines per composer

Composer	TP Rate	FP Rate	Precision	Recall	AUC
HA	0.80	0.09	0.74	0.80	93%
BE	0.65	0.04	0.77	0.65	94%
BA	0.95	0.10	0.93	0.95	98%

The ROC curves of the two best models according to Table 3 are displayed in Figure 6. The ROC curve displays the trade-off between true positive rate (TPR) and false negative rate (FNR). Both models clearly score better than a random classification, which is represented by the diagonal through the origin. Although both models have a high AUC value, the ROC curves for the SVM score slightly better. When examining the misclassified pieces, they all seem to have a very low probability, with an average of 39%. Examples of misclassified pieces are String Quartet in C major, Op. 74, No. 1, Allegro moderato from Haydn, which is classified as Bach with 38% probability and Six Variations on a Swiss Song, WO 64 (Theme) from Beethoven, which is classified as Bach with a probability of 38%.

A second experiment was conducted in a similar way with 10-fold cross validation. Three balanced datasets were randomly extracted from the dataset described in the previous section. Each of the datasets (R1, R2 and R3) contains 196 pieces per composer. The results of this experiment are displayed in Table 12. As is to be expected with a



(a) ROC Logistic Regression (b) ROC Support Vector Machines

Figure 6. ROC curves of the best performing models

smaller dataset, the accuracy and AUC values are lower than those from the previous experiment with the full dataset (see Table3). They are, however, still reasonably high, reaching AUC values of 94% and accuracy (Acc.) of 81% when classifying between the three composers. The best performing model is again support vector machines (SVM), closely followed by logistic regression (LR).

Table 12. Evaluation of the models with 10-fold cross-validation on reduced balanced datasets

	R1		R2		R3	
Method	Acc.	wAUC	Acc.	wAUC	Acc.	wAUC
RIPPER	*70%* (5.90)	*81%* (4.63)	*73%* (5.99)	*83%* (4.44)	*73%* (6.15)	*82%* (4.70)
C4.5	*75%* (3.06)	*85%* (4.11)	*72%* (6.00)	*84%* (4.11)	*71%* (3.74)	*84%* (4.05)
LR	78% (3.75)	91% (2.32)	76% (4.31)	91% (2.32)	75% (5.32)	*92%* (2.62)
SVM	**81%** (4.80)	**93%** (2.44)	**79%** (4.39)	**92%** (1.85)	**80%** (3.64)	**94%** (2.07)

$p < 0.01$: italic, $p > 0.05$: bold, best: bold.
Standard deviations are shown between parentheses.

Table 13 shows the results of a third experiment whereby pairwise classification models were built. One composer was removed from the dataset for each run, so that

only two remain. This process was performed using both the full dataset and a randomly balanced dataset. Classifying between Bach and Beethoven seems to be the easiest task, as was previously mentioned. Support vector machines reach an accuracy of 95% on the full dataset and 94% on the balanced dataset. The AUC values of the classifyier reaches 99% and 98% respectively. These results come close to those reported by Dor and Reich (2011), who reach accuracy values of 96–98% when classifying between these two composers. Table 13 shows that when classifying between Beethoven and Haydn an accuracy of 80% and 82% can be reached (AUC 88% and 90% respectively). These results are comparable to the successrate of Kaliakatsos-Papakostas et al. (2011), who reported 59%–88% success performance when classifying Beethoven versus Haydn. In order to be able to compare results of different studies without bias, however, the exact same database should be used. The support vector machines classifier again outperforms the others. The quality of logistic regression is comparable as the statistical tests almost all have a p-value of > 0.01 (except one).

Table 13. Evaluation of the models with 10-fold cross-validation with pairwise classification

		RIPPER	C4.5	LR	SVM
BA–BE	Acc.	92% (4.09)	92% (3.10)	92% (2.49)	**95%** (2.09)
	wAUC	*89%* (5.31)	*91%* (4.83)	97% (1.14)	**99%** (1.26)
BA–BE (bal)	Acc.	*88%* (3.97)	*86%* (4.63)	**91%** (5.28)	**94%** (3.66)
	wAUC	*88%* (4.05)	*87%* (4.38)	96% (3.45)	**98%** (2.46)
BA–HA	Acc.	89% (3.15)	*87%* (2.85)	91% (2.94)	**94%** (2.72)
	wAUC	*87%* (4.34)	*88%* (3.98)	96% (2.39)	**97%** (2.14)
BA–HA (bal)	Acc.	**87%** (5.92)	*83%* (2.94)	**91%** (3.09)	**92%** (4.20)
	wAUC	*88%* (6.25)	*88%* (3.17)	96% (1.30)	**98%** (1.54)
BE–HA	Acc.	76% (6.96)	75% (5.28)	**79%** (4.33)	**80%** (4.44)
	wAUC	*76%* (7.44)	*77%* (7.68)	86% (4.25)	**88%** (3.99)
BE–HA (bal)	Acc.	76% (5.91)	74% (5.55)	79% (4.57)	**82%** (6.11)
	wAUC	*77%* (5.09)	*76%* (5.09)	88% (5.99)	**90%** (4.62)

$p < 0.01$: italic, $p > 0.05$: bold, best: bold.
Standard deviations are shown between parentheses.

Generating composer-specific music

The idea of algorithmic composition has been present ever since Mozart invented the musical dice game (Musikalisches Würfelspiel) (Boenn et al. 2009). More recently, composers such as John Cage have made use of chance in their compositional process. His piece "Reunion" (1968) is performed differently each time, because it is controlled by moves on a chess board. Each move on the board is registered by photo-receptors and triggers electronic sounds (Fetterman 1996). Charles Dodge used a natural phenomena to inspire his piece "The Earth's Magnetic Field" (1970). This piece is a musical translation of the fluctuations of the earth's magnetic field (Alpern 1995). Lejaren Hiller and Leonard Isaacson used the Illiac computer in 1957 to generate the score of a string quartet resulting in a piece called the "Illiac Suite" (Alpern 1995). More recent compositional systems include, among others, The Continuator (Pachet 2003), OMax (Assayag et al. 2006) and MIMI (Schankler et al. 2014). An approach related to this research, yet using different classes and features, is described by Pachet (2009). Global features are used to develop a support vector machine classification model (SVM) that can classify between tonal, brown, serial, long and short melodies. Existing melodies were then transformed into another type by improving the SVM's score for this particular melody. For a more complete overview of algorithmic composition systems the reader is referred to Herremans and Sörensen (2012) and Fernández Rodríguez and Vico Vela (2013).

A limited number of researchers have investigated automatic composition in the style of a specific composer. David Cope's Experiments in Musical Intelligence (EMI) extract signatures of musical pieces using pattern matching in order to understanding a composer-specific style. He uses a grammar based system to generate into the style of a chosen composer (Cope 1991). The generation of Bach chorales has received some attention in the field of automatic composition. Ebcioğlu (1988) created CHORAL, an

expert system which uses 350 rules for harmonizing four-part Bach Chorales. A Bach harmonization system was also developed by Hild et al. (1992). They created a hybrid system with artificial neural networks called HARMONET. Phon-Amnuaisuk (2002) implemented a control language to harmonize Bach chorales. Spangler (1999) built a system that extracts rules from Bach chorales. These rules are implemented in a system that can harmonize in Bach's style in response to an input in real time.

While there has been some research about automatic composition in the style of one particular composer, combining multiple composers has not received a lot of attention in existing research. The composition system developed by Cope (2000) called SARA combines the influence of selected composers in the generated music. Most of the existing systems, however, focus on learning or defining one particular style, which might be extracted from a collection of works by one composer (e.g. Farbood and Schoner (2001); Herremans et al. (2014b)). In this research, we take a novel approach by allowing to user to dynamically alter and combine the influence of multiple composers in a stream of newly generated 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 2013).

In order to generate music with composer-specific characteristics, the existing objective function for evaluating counterpoint (f_{cp}) was extended with the probabilities

of the logistic regression model. This model was preferred over the slightly more accurate SVM model since it is easy to comprehend (Martens et al. 2007) and returns a clearly defined probability per composer. The resulting objective function for composer i is displayed in Equation 11. When composing with characteristics of a certain 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 contrapuntal music with more influences of composer i.

$$f_i = f_{cp} + \sum_{i \in BE,BA,HA} a_i \cdot (1 - f_{comp}(L_i)) \tag{11}$$

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 11, 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 contrapuntal music with characteristics of a certain composer.

Implementation - FuX

The music generation algorithm with the objective function discussed in the previous section (based on counterpoint rules and the logistic regression model) 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 contrapuntal music with composer-specific characteristics. This music is continuously generated while it is being played.

Android is an operating system for mobile devices based on Linux (Kernel 2.6) (Mongia and Madisetti 2010). 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 2013), 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 Sorensen (2013).

The graphical user interface of FuX 2.0 is displayed in Figure 7. The three sliders (or SeekBars) give the user control over the weights a_i of the objective function (see Equation 11). 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.

[1] http://play.google.com/store/apps/details?id=com.dh.fux2

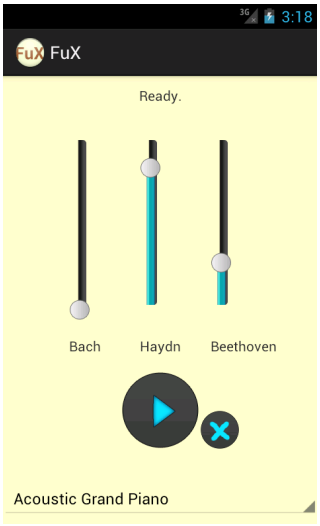


Figure 7. User interface of FuX

Results

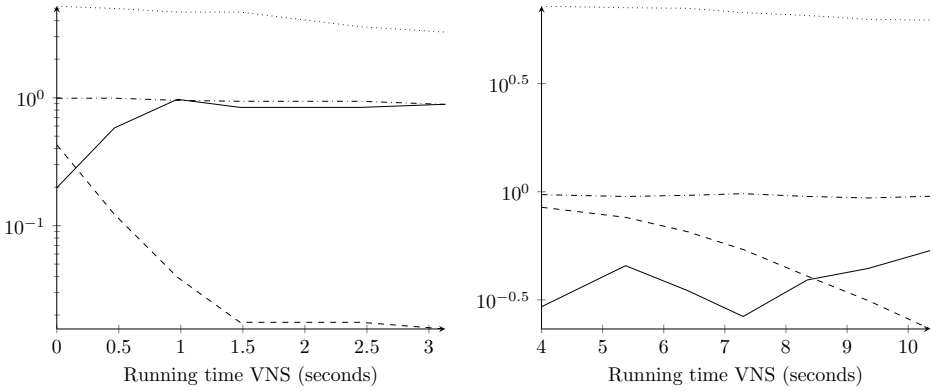
The resulting composer-specific music generation algorithm was tested on an Eclipse Android Virtual Device with Android 4.0.3, ARM processor and 512MB RAM. The emulator was installed on an OpenSuse system with Intel®Core™2 Duo CPU@ 2.20GHz and 3.8GB RAM. Figure 8 displays the evolution of the solution quality over time. The left plots describe the generation of the cantus firmus (CF) or bass line. The plots on the right hand side describe the evolution of the score of the counterpoint (CP) line or top line. In the experiment 16 measures are generated with a cutoff time of 12 seconds. FuX first generates the bass 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), this ensures that the probability of a composer is preferred by the algorithm over the counterpoint rules. Figure 8 shows a drastic improvement of the selected composer's score for each of the three composers. For example, in Figure 8(a) $f_{comp}(L_{HA}$ goes down rapidly over time, while $f_{comp}(L_{BE}$ and $f_{comp}(L_{BA}$ remain relatively stable. 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. Of course, the thought processes of the great composers are far more complex than can be captured by the melodic features used in this research. The limited set of composer-specific characteristics (see Table 2) that FuX overlays on the counterpoint fragment are not enough by themselves to generate a piece of music that would be recognized as being composed by one of the selected composers. The generated music does however contain characteristics of the selected composer. All three composers used in this research composed music that differs in much more than the limited set of characteristics that FuX controls. Bach, e.g., worked in the Baroque period while Beethoven's work was composed during the transition from the Classical to the Romantic Era. The differences in style between the musical styles common during these different periods are vastly more encompassing than simple variations in the melodic characteristics recognized by FuX. This research can be seen as a first step towards creating a system that is able to generate more complete musical pieces in the style of a certain composer.

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.

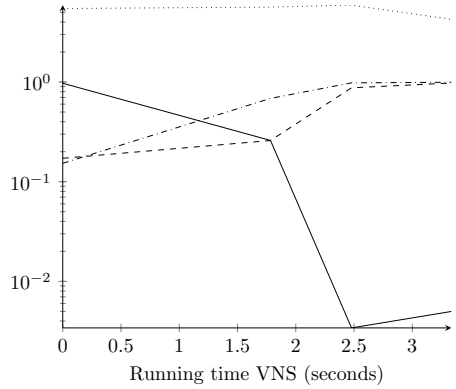
Conclusions

A number of musical features were extracted from a large database of music. Based on these features four classification models were built. The first two models, an if-then

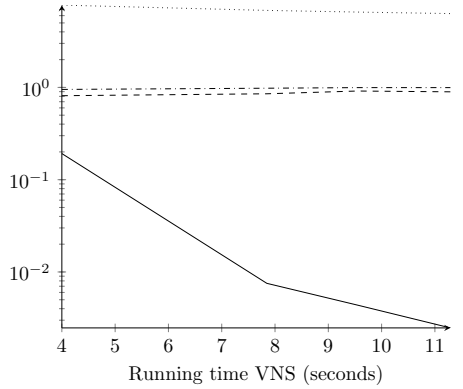


(a) Haydn (CF)

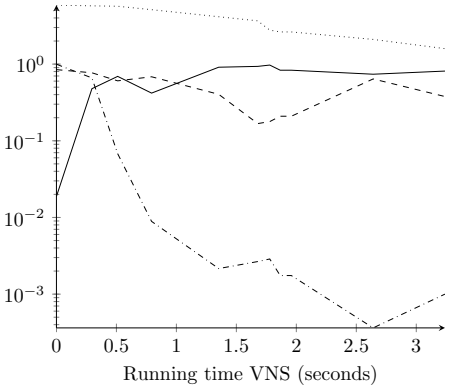
(b) Haydn (CP)



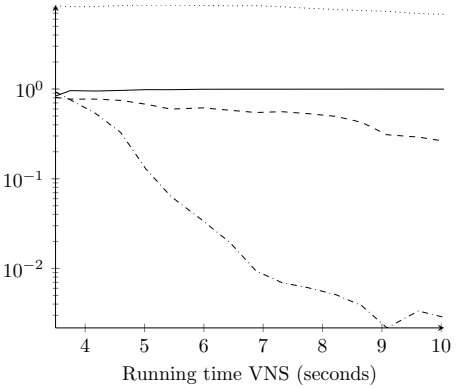
(c) Bach (CF)



(d) Bach (CP)

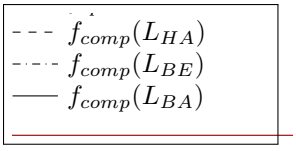


(e) Beethoven (CF)



(f) Beethoven (CP)

Figure 8. Evolution of solution quality over time



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 other two models, a logistic regression and a support vector machine classifier, can more accurately classify musical pieces from Haydn, Beethoven and Bach. The first of these models is integrated in the objective function of a variable neighborhood search algorithm that can efficiently generate contrapuntal music. The resulting algorithm was implemented as a user friendly Android app called FuX, which is able to play a stream of contrapuntal music with composer-specific characteristics that sounds pleasing, at least to the subjective ear of the authors.

Combining a certain composer's characteristics with the counterpoint style creates a peculiar fusion of styles, yet this approach is merely an initial step towards a more complete system. In the future it would be interesting to work with composer classification models built on a dataset of one particular style (e.g., string quartets). Enforcing basic musical properties while generating pieces with characteristics specific to a composer might then be done by integrating rules specific to the chosen style instead of the currently used counterpoint rules. Other future extensions of this research include working with other musical styles, more voices and adding a recurring theme to the music. FuX might also be ported to other platforms, such as iOS from Apple.

References

Alpern, A. 1995. "Techniques for algorithmic composition of music." *On the web: http://hamp. hampshire. edu/~ adaF92/algocomp/algocomp* 95.

Assayag, G., G. Bloch, M. Chemillier, A. Cont, and S. Dubnov. 2006. "Omax brothers: a dynamic yopology of agents for improvization learning." In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*. ACM, pp. 125–132.

Baesens, B., R. Setiono, C. Mues, and J. Vanthienen. 2003. "Using neural network rule extraction and decision tables for credit-risk evaluation." *Management Science* 49(3):312–329.

Berenzweig, A., B. Logan, D. Ellis, and B. Whitman. 2004. "A large-scale evaluation of acoustic and subjective music-similarity measures." *Computer Music Journal* 28(2):63–76.

Boenn, G., M. Brain, M. De Vos, and J. Ffitch. 2009. "Automatic composition of melodic and harmonic music by answer set programming." *Logic Programming* 5366:160–174.

Bornstein, D. 2008. "Dalvik vm internals." In *Google I/O Developer Conference*, volume 23. 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*, volume 2002. p. 167.

Byrd, D., and T. Crawford. 2002. "Problems of music information retrieval in the real world." *Information Processing & Management* 38(2):249–272.

Casey, M., R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. 2008. "Content-based music information retrieval: Current directions and future challenges." *Proceedings of the IEEE* 96(4):668–696.

CCARH. 2012. *KernScores, http://kern.ccarh.org*. URL http://kern.ccarh.org. Last accessed: November 2012.

Chang, C.-C., and C.-J. Lin. 2011. "LIBSVM: a library for support vector machines." *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):27.

Cohen, W. 1995. "Fast Effective Rule Induction." In A. Prieditis, and S. Russell, (editors) *Proceedings of the 12th International Conference on Machine Learning*. Tahoe City, CA: Morgan Kaufmann Publishers, pp. 115–123.

Conklin, D. 2013. "Multiple viewpoint systems for music classification." *Journal of New Music Research* 42(1):19–26.

Cope, D. 1991. "Computers and musical style." .

Cope, D. 2000. *The algorithmic composer*, volume 16. AR Editions, Inc.

Cramer, G., R. Ford, and R. Hall. 1976. "Estimation of toxic hazard—a decision tree approach." *Food and cosmetics toxicology* 16(3):255–276.

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

Cristianini, N., and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. New York, NY, USA: Cambridge University Press.

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

Dor, O., and Y. Reich. 2011. "An evaluation of musical score characteristics for automatic classification of composers." *Computer Music Journal* 35(3):86–97.

Downie, J. 2003. "Music information retrieval." *Annual review of information science and technology* 37(1):295–340.

Dubnov, S., G. Assayag, O. Lartillot, and G. Bejerano. 2003. "Using machine-learning methods for musical style modeling." *Computer* 36(10):73–80.

Ebcioğlu, K. 1988. "An expert system for harmonizing four-part chorales." *Computer Music Journal* :43–51.

Farbood, M., and B. Schoner. 2001. "Analysis and synthesis of Palestrina-style counterpoint using Markov chains." In *Proceedings of the International Computer Music Conference*. pp. 471–474.

Fawcett, T. 2004. "ROC graphs: Notes and practical considerations for researchers." *Machine Learning* 31:1–38.

Fernández Rodríguez, J. D., and F. J. Vico Vela. 2013. "AI Methods in Algorithmic Composition: A Comprehensive Survey." .

Fetterman, W. 1996. *John Cage's theatre pieces: notations and performances*. Routledge.

Friedl, M., and C. Brodley. 1997. "Decision tree classification of land cover from remotely sensed data." *Remote sensing of environment* 61(3):399–409.

Friedman, J., T. Hastie, and R. Tibshirani. 2000. "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)." *The Annals Of Statistics* 28(2):337–407.

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

Geertzen, J., and M. van Zaanen. 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.

Gheyas, I., and L. Smith. 2010. "Feature subset selection in large dimensionality domains." *Pattern Recognition* 43(1):5–13.

Ghias, A., J. Logan, D. Chamberlin, and B. Smith. 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.

Gómez Gutiérrez, E. 2006. "Tonal description of music audio signals." Ph.D. thesis, Universitat Pompeu Fabra.

Google. 2013. *Google Android SDK, http://developer.android.com/sdk/index.html*. URL http://developer.android.com/sdk/index.html. Last accessed: November 2013.

Greene, D. 1985. *Greene's biographical encyclopedia of composers*. Reproducing Piano Roll Fnd.

Guyon, I., J. Weston, S. Barnhill, and V. Vapnik. 2002. "Gene selection for cancer classification using support vector machines." *Machine learning* 46(1-3):389–422.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. 2009. "The Weka data mining software: an update." *ACM SIGKDD Explorations Newsletter* 11(1):10–18.

Hand, D., and W. Henley. 1997. "Statistical classification methods in consumer credit scoring: a review." *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 160(3):523–541.

Herremans, D., D. Martens, and K. Sörensen. 2014a. "Dance hit song prediction." *Journal of New Music Research* 43(3):291–302.

Herremans, D., and K. Sörensen. 2012. "Composing first species counterpoint with a variable neighbourhood search algorithm." *Journal of Mathematics and the Arts* 6(4):169–189.

Herremans, D., and K. Sörensen. 2013. "Composing Fifth Species Counterpoint Music With A Variable Neighborhood Search Algorithm." *Expert Systems with Applications* 40(16):6427–6437.

Herremans, D., and K. Sorensen. 2013. "FuX, an android app that generates counterpoint." In *Proceedings of IEEE Symposium on Computational Intelligence for Creativity and Affective Computing (CICAC), April 2013* . pp. 48–55.

Herremans, D., K. Sörensen, and D. Conklin. 2014b. "Sampling the extrema from statistical models of music with variable neighbourhood search." *Proceedings of ICMC–SMC* :1096–1103.

Hild, H., J. Feulner, and W. Menzel. 1992. "HARMONET: A neural net for harmonizing chorales in the style of JS Bach." In *Advances in Neural Information Processing Systems*. pp. 267–274.

Hillewaere, R., B. Manderick, and D. Conklin. 2010. "String quartet classification with monophonic models." *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR), Utrecht, the Netherlands* .

Huang, W., Y. Nakamori, and S.-Y. Wang. 2005. "Forecasting stock market movement direction with support vector machine." *Computers & Operations Research* 32(10):2513–2522.

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

Kaliakatsos-Papakostas, M., M. Epitropakis, and M. Vrahatis. 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(2):59–67.

Kim, J., H. Song, T. Kim, and H. Kim. 2005. "Detecting the change of customer behavior based on decision tree analysis." *Expert Systems* 22(4):193–205.

Kononenko, I. 2001. "Machine learning for medical diagnosis: history, state of the art and perspective." *Artificial Intelligence in medicine* 23(1):89–109.

Kurt, I., M. Ture, and A. Kurum. 2008. "Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease." *Expert Systems with Applications* 34(1):366–374.

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

Lin, C., J. Lin, C. Dow, and C. Wen. 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(2):137–142.

Manaris, B., J. Romero, P. Machado, D. Krehbiel, T. Hirzel, W. Pharr, and R. Davis. 2005. "Zipf's law, music classification, and aesthetics." *Computer Music Journal* 29(1):55–69.

Martens, D. 2008. "Building Acceptable Classification Models for Financial Engineering Applications." *SIGKDD Explorations* 10(2):30–31.

Martens, D., B. Baesens, T. Van Gestel, and J. Vanthienen. 2007. "Comprehensible credit scoring models using rule extraction from support vector machines." *European Journal of Operational Research* 183(3):1466–1476.

Martens, D., and F. Provost. 2014. "Explaining Data-Driven Document Classifications." *MIS Quarterly* 38(1):73–99.

Martens, D., T. Van Gestel, and B. Baesens. 2009. "Decompositional Rule Extraction from Support Vector Machines by Active Learning." *IEEE Transactions on Knowledge and Data Engineering* 21(2):178–191.

Martens, D., J. Vanthienen, W. Verbeke, and B. Baesens. 2011. "Performance of classification models from a user perspective." *Decision Support Systems* 51(4):782–793.

McKay, C., and I. Fujinaga. 2006. "jSymbolic: A feature extractor for MIDI files." In *Proceedings of the International Computer Music Conference*. pp. 302–5.

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

McKay, C., and I. Fujinaga. 2009. "jMIR: Tools for automatic music classification." In *Proceedings of the International Computer Music Conference*. pp. 65–8.

Mearns, L., D. Tidhar, and S. Dixon. 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., and V. Madisetti. 2010. "Reliable Real-Time Applications on Android OS." *IEEE Electrical and Computer Engineering Electrical and Computer Engineering* .

Ngai, E., L. Xiu, and D. Chau. 2009. "Application of data mining techniques in customer relationship management: A literature review and classification." *Expert Systems with Applications* 36(2):2592–2602.

Pachet, F. 2003. "The continuator: Musical interaction with style." *Journal of New Music Research* 32(3):333–341.

Pachet, F. 2009. "Description-based design of melodies." *Computer Music Journal* 33(4):56–68.

Pearce, J., and S. Ferrier. 2000. "Evaluating the predictive performance of habitat models developed using logistic regression." *Ecological modelling* 133(3):225–245.

Phon-Amnuaisuk, S. 2002. "Control language for harmonisation process." In *Music and Artificial Intelligence*. Springer, pp. 155–167.

Pollastri, E., and G. Simoncelli. 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. San Mateo, CA: Morgan Kaufmann.

Ruggieri, S. 2002. "Efficient C4. 5 [classification algorithm]." *Knowledge and Data Engineering, IEEE Transactions on* 14(2):438–444.

Sapp, C. 2005. "Online database of scores in the Humdrum file format." In *Intl. Symp. on Music Information Retrievel*. pp. 664–665.

Schankler, I., E. Chew, and A. R. François. 2014. "Improvising with Digital Auto-Scaffolding: How Mimi Changes and Enhances the Creative Process." In *Digital Da Vinci*. Springer, pp. 99–125.

Searls, D., et al. 2002. "The language of genes." *Nature* 420(6912):211–217.

Shmueli, G., and O. Koppius. 2011. "Predictive Analytics in Information Systems Research." *MIS Quarterly* 35(3):553–572.

Son, K., and J. Lee. 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.

Spangler, R. R. 1999. "Rule-based analysis and generation of music." Ph.D. thesis, California Institute of Technology.

Tan, P., et al. 2007. *Introduction to data mining*. Pearson Education India.

Tong, S., and D. Koller. 2002. "Support vector machine active learning with applications to text classification." *The Journal of Machine Learning Research* 2:45–66.

Tu, J. 1996. "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes." *Journal of Clinical Epidemiology* 49(11):1225–1231.

Tzanetakis, G., and P. Cook. 2002. "Musical genre classification of audio signals." *IEEE transactions on Speech and Audio Processing* 10(5):293–302.

Tzanetakis, G., A. Ermolinskyi, and P. Cook. 2003. "Pitch histograms in audio and symbolic music information retrieval." *Journal of New Music Research* 32(2):143–152.

van Kranenburg, P., and E. Backer. 2004. "Musical style recognition-a quantitative approach." In *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*. pp. 106–107.

Vapnik, V. 1995. *The nature of statistical learning theory*. New York, NY, USA: Springer New York, Inc.

Weka. 2013a. "Optimizing Parameters." URL http://weka.wikispaces.com/Optimizing+parameters. Last accessed: October 2014.

Weka. 2013b. "Weka documentation, class GridSearch." URL http://weka.sourceforge.net/doc.stable/weka/classifiers/meta/GridSearch.html. Last accessed: October 2014.

Whitman, B., and P. Smaragdis. 2002. "Combining musical and cultural features for intelligent style detection." In *Proceedings of the International Society for Music Information Retrieval Conference (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(03):757–770.

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

Wolberg, W., and O. Mangasarian. 1990. "Multisurface method of pattern separation for medical diagnosis applied to breast cytology." *Proceedings of the national academy of sciences* 87(23):9193–9196.

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

Wu, X., V. Kumar, J. Ross Quinlan, J. Ghosh, et al. 2008. "Top 10 algorithms in data mining." *Knowledge and Information Systems* 14(1):1–37.

Youngblood, J. 1958. "Style as information." *Journal of Music Theory* :24–35.