A Clustering Analysis of Renaissance Polyphony Using State-Space Models

BRAM GEELEN, DAVID BURN, AND BART DE MOOR

1. Introduction

Symbolic music analysis is the study of digital music represented as a sequence of events. These events are primarily the onset of notes, but also include rests, accidentals, time signatures, and clefs. Some important analytical goals within this area are missing-note prediction and the matching of musical scores to audio. This paper focuses on composer identification in early music based on a symbolic representation.

Within the area of symbolic music analysis, Dean Keith Simonton studied the application of note transitions in 1984, and specifically the use of two- and three-note transitions, in order to distinguish between the musical writing style of composers in a corpus of 15,618 works.¹ In some sense, this encoding is very similar to taking an n-gram approach, in which the occurrence of (very) short snippets of melody are used to identify the author of a musical work. This is the approach McKay et al. proposed in a 2020 paper: using n-grams of melodies to identify composers.²

The authors of this paper have implemented their methods in jSymbolic, a software suite, which extracts features from symbolic music.³ The latest release—jSymbolic 2.2—derives up to 246 such features, which describe concrete counts and statistics that can be extracted from midi files. These include various pitch statistics, the prevalence of musical intervals, instrumentation, but also rhythmic information, like statistics on note and rest duration, meter, time between attacks, etc. In Section 3.1, we shall compare the results of classifying the dataset through our method, with the representation built by the statistics output by jSymbolic.

Peter Van Kranenburg also proposed a feature-extraction approach for the composer attribution problem. Specifically, in a 2006 paper, he applied his methods to a corpus of Baroque organ fugues.⁴ By using the counts of twenty intervals as a representation of a work, together with (supervised) dimensionality reduction, he visualized the more than one thousand works in the *Bach-Werke-Verzeichnis* onto a 2D plane. Focusing on sixteen fugues with uncertain authorship, he argues that some of them should be attributed to Johann Ludwig Krebs, instead of Johann Sebastian Bach.

¹ Dean Keith Simonton, 'Melodic Structure and Note Transition Probabilities: A Content Analysis of 15,618 Classical Themes', in *Psychology of Music* 12 (1984), 3-16.

² Cory McKay et al, 'Exploring Renaissance Music Using n-gram Aggregates to Summarize Local Musical Content', presented at the *Medieval and Renaissance Music Conference*, Edinburgh, 1-4 July 2020, presentation accessible at <<http://www.music.mcgill.ca/~cmckay/papers/musictech/mckay20exploring.pdf>> (accessed 28 September 2020).

³ Cory McKay and Ichiro Fujinaga, 'jSymbolic: A Feature Extractor for MIDI Files', in *Proceedings of the International Computer Music Conference*, New Orleans, 6-11 November 2006 (New Orleans, 2006), 302-5.

⁴ Peter van Kranenburg, 'Composer Attribution by Quantifying Compositional Strategies', in *Proceedings of the 7th International Conference on Music Information Retrieval*, Victoria, Canada, 2006 (Victoria, 2006), 375-76.

The goal of this paper is to identify novel, meaningful patterns in Renaissance polyphony, by using autoregressive state-space models. The corpus we use—the Josquin Research Project⁵ dataset—contains 1248 transcribed musical Renaissance works, which are freely available in different formats (PDF, MIDI, etc.). The works are attributed to twenty different composers, of which Pierre de la Rue (176 works) and Josquin des Prez (136 works) are the most prevalent. However, 416 works are insecurely attributed, or attributed to an anonymous composer. Still, the database has enough labelled items, such that valid, meaningful conclusions can be made about these labels through induction. In Section 3.1 of this paper, we shall suggest new attributions for the insecurely attributed works, based on classification algorithms using our method.

By using a state-space model approach, we shall try to model the harmonic progressions of a complete song. This is in contrast to previous approaches, which rely more on the counting of the occurrence of certain fragments, or the incidence of (harmonic and melodic) intervals by size. We shall capture the melodic *and* harmonic movements of an entire work in a state-space model matrix, by modelling how the chromatic activity moves from tactus to tactus, through the whole piece. This means that the features we produce will be slightly less specific in their meaning. But because they will be exclusively built on the melodic progression of the work, they should encapsulate the musical tone and feeling as part of a composer’s writing style.

This article is structured as follows: in Section 2, we offer a brief introduction to state-space models, explaining how a musical work fits within this framework, using an example composition. We then use the state-space models as vectors in a classification problem, in order to associate previously unattributed, or insecurely attributed works with a composer. Section 3.1 describes how the classification algorithms work, as well as their results. In Section 3.2, we use dimensionality reduction techniques to visualize the dataset. Finally, the conclusion in Section 4 recapitulates the proposed method and results.

2. Method

2.1 State-space Models for Discrete Time Signals

A state-space model is a mathematical technique for modelling the behaviour of a discrete time signal using vectors and matrices. The theory behind state-space models considers a perceived discrete time signal as the output of a *dynamical system* that has an *internal state*, which changes at discrete *time instants* (or *time steps*), as can be seen in Figure 1.⁶

To model a signal with a state-space representation implies the determination of the transition and output matrix from given data. As can be seen in Figure 1, an underlying process modifies a state from step to step. In turn, the output is the result of a transformation of the state at that time instant. If we identify the process by estimating

⁵ Jesse Rodin, ‘The Josquin Research Project’, Stanford University, 2010, <<https://josquin.stanford.edu/>>, (accessed 28 September 2020).

⁶ Each arrow in Figure 1 signifies a step to be modelled, usually by a linear transformation (i.e., a matrix multiplication) that can be estimated using example data.

Figure 1: Illustration of a state transition matrix used to model an output signal

the transition matrix (horizontal arrows), as well as how the output is produced based on the state (vertical arrows), then we can predict how the signal will behave in the future, based on the last (few) states. After all, by estimating the next state and the next signal output, and applying the model on its own estimations, we can create a prediction for even more future states, which should at least be accurate for the next couple of states.

In our case, we consider that the output signal is equal to the state of the process, and that this state of the process is the *chromatic activity* at a certain tactus. This way, the model estimates the temporal correlations in a song, which correspond with harmonic progressions in the modelled music. This unification is crucial. Otherwise, the internal state of the signal could have different meanings between different musical works. When we compare different musical works this way—which we do in Section 2.4 onwards—we explicitly want the internal state to have a consistent interpretation. Limiting the output signal to be equal to the internal state results in the fact that we only have to estimate a single matrix when modelling the signal, and it is exactly this *state transition matrix* we shall compare between works.

2.2 Representing Symbolic Music as a Discrete Signal

In order to represent a musical signal using state-space signals, we first need to define what constitutes a time instant in music, and how we represent the current state at a given time instant. We shall consider each *tactus* to correspond to a time step; it is the most logical fit. Another possibility is to have each different note onset to be a new time step; this approach is used by McKay et al. in the jSymbolic toolbox. However, in this setup, the actual time between every time step is variable. This is very unconventional in the signal processing literature, whereas having a constant sampling rate is more common. Moreover, Ruth DeFord has described the *tactus*—specifically the ‘compositional tactus’—as ‘a standard of reference for various aspects of rhythm, such as the rate of contrapuntal motion, dissonance treatment, and syncopation, in a composition.’⁷ This implies that the harmonic progression of the work will be expressed exactly to the rhythm of the *tactus*. As a result, we also measure—or sample—the composition on a per-tactus basis.

Next, we have to define what the state at every *tactus* consists of. In state-space models, the state is a *vector*. This means that we should specify a list of variables which together represent the state at a specific time instance. The simplest definition would be

⁷ Ruth I. DeFord, *Tactus, Mensuration and Rhythm in Renaissance Music* (Cambridge, 2015), 51–52.

to have a number in the state for every (chromatic) note height. For example—following the MIDI specification—we could create a vector of size 128, with one value for every note from C-1 to G9. This approach was taken by Nicolas Boulanger-Lewandowski et al., in their 2012 paper.⁸ Although this creates a direct, reversible link between the note height and the value in the vector, this method is rather superfluous; the JRP dataset only uses notes A ♯, to a', so the eighty state values corresponding to MIDI note-heights outside of this interval will never be active. Modelling the state transitions for these notes would be useless, and result in computational *overhead* without any added value. Moreover, some of the note-heights *within* this range are used so rarely that conclusions based on their occurrence would be less dependable. For this reason, we go one step further, aggregating the notes by *chroma*. As a result of this aggregation, the final vector contains twelve elements (one for every pitch class), and contains the number of notes in this tactus of every chroma. Note that regarding *musica ficta*, we consider the music to be canonical in the way it is transcribed in the dataset. As a result, the notes outside the hexachord system are encoded separately, and still represent a different harmonic progression than the *recta* notes.

We define the activity of a specific chroma as the number of voices that have that chroma in the relevant tactus. We thus construct a vector with twelve values, one for every chroma. For notes that do not fill an entire tactus—for example because they are shorter than the tactus length, or because they begin or end halfway through the tactus—we only add the fraction of the tactus that the note fills to the relevant value (usually this is 0.5, when the note is sung for half the tactus).

Example 1: The first five measures of Anon., *Helas mon coeur*. Credits to the JRP

The image shows a musical score for three voices: Soprano, Tenor, and Contra. The score is written in a medieval style with square notes on a four-line staff. The Soprano part starts on a high note and moves down. The Tenor part starts on a middle note and moves down. The Contra part starts on a low note and moves up. The notation is in a medieval style with square notes on a four-line staff.

The following examples will serve as an illustration. Example 1 shows the first five measures of the work *Helas mon coeur*, one of the anonymous unica from the Leuven Chansonnier.⁹ Figure 2 displays how this snippet is discretized into a matrix format. The

⁸ Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent, 'Modeling Temporal d-Dependencies in High-dimensional Sequences: Application to Polyphonic Music Generation and Transcription', in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, 26 June-July 1 2012 (Madison, WI, 2012), 1881-88.

⁹ The transcription provided by the Josquin Research Project and is found in its entirety in the Appendix. Different digital formats of the transcription can be found on the JRP website: <<https://josquin.stanford.edu/work/?id=Ano3225>> (.krn format accessed on 28 September 2020). The technical source for Figure 2 was the accompanying

B	0	0	0	0	0	0	0	0	0	0	...
B $\downarrow$ f	0	0	0	0	1	0	0	0	2	0	...
A	0	0	1	0	0	0	0	1	0	0	...
G $\downarrow$ f	0	0	0	0	0	0	0	0	0	0	...
G	3	3	0	2	0	1	0	0	0	2	...
F $\downarrow$ f	0	0	0	0	0	0	0	0	0	0	...
F	0	0	1	0	1	0	1	0	0	0	...
E	0	0	0	1	0	1	0	1	0	0	...
E $\downarrow$ f	0	0	0	0	0	0	0	0	0	1	...
D	0	0	1	0	1	0	2	0	1	0	...
C $\downarrow$ f	0	0	0	0	0	0	0	0	0	0	...
C	0	0	0	0	0	1	0	1	0	0	...

Figure 2: Matrix representation of the Leuven Chansonnier unicum *Helas mon cuer*¹⁰

first ten columns of this matrix correspond to the first five measures of the work (every measure consists of two tacti). The first measure starts with all three voices singing a G for the complete measure, so the first two columns (tacti) only consist of a value of 3 in the G row.

2.3 Modelling the Harmonic Progression of the Song

In order to encapsulate the musical progression of a song in a numerical format, we construct a simple model to estimate the activity of the chroma in the next tactus. In a *first-order model*, we use a simple matrix multiplication, to go from one state to the next, as follows:

$$\text{tactus}_{t+1} \cong X \cdot \text{tactus}_t$$

In this approximation, tactus_t is the vector state at a specific tactus t , as defined above. This is a vector of size twelve with the chroma activity, for example the first column of the matrix in Figure 2. We matrix-multiply it with a square matrix X of size twelve by twelve, to create an estimate of the (next) vector state, i.e., tactus_{t+1} .

In a matrix multiplication, we take the twelve values of every row of the matrix X , and multiply them with the twelve values in the state vector tactus_t . We then sum the result of these multiplications per row, such that we get a vector with as many values as there are rows in the matrix X . As a result, every row of the matrix X is responsible for predicting the activity of a single chroma in the *next* step, while every *value* in the row is tied to the activity of a different chroma in the *previous* step.

The above formula describes an estimation, not an equation (hence the special equals sign). After all, within the confinement of a matrix multiplication, it is impossible to predict exactly the next state vector, for every state of a work. However, by applying

Humdrum file at <https://github.com/josquin-research-project/Ano/blob/master/Ano3225-Helas_mon_cueur_tu_mociras--Leuven_Chansonnier.krn> (accessed on 28 September 2020).

¹⁰ The columns of this matrix represent the different timesteps, in this case corresponding to only the first five measures of the work. The rows of the matrix correspond to the twelve pitch classes.

B	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
B $\downarrow$	0.	0.254	0.236	-0.	0.051	-0.399	0.057	0.31	-0.092	0.064	-0.546	0.392
A	0.	0.165	0.344	-0.	0.163	-0.528	0.022	0.028	0.359	0.052	-0.371	-0.135
G $\nearrow$	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
G	0.	0.144	0.166	0.	0.488	3.567	0.785	-0.538	-0.133	0.158	0.372	0.307
F $\downarrow$	0.	-0.014	0.098	-0.	0.01	-0.175	-0.096	-0.042	-0.018	0.033	-0.056	0.011
F	0.	0.133	0.164	-0.	0.097	-0.188	0.092	0.506	-0.17	-0.023	-0.67	0.105
E	0.	0.004	-0.068	0.	-0.008	-0.03	0.315	0.154	0.	0.09	-0.086	0.005
E $\uparrow$	0.	0.132	0.023	-0.	-0.026	-0.044	-0.056	0.028	0.364	0.01	-0.05	-0.013
D	0.	-0.018	0.136	-0.	0.211	-0.855	-0.166	0.465	0.169	0.506	2.399	0.357
C $\searrow$	0.	0.015	0.044	-0.	-0.013	-0.073	-0.044	0.154	0.09	0.01	-0.198	-0.081
C	0.	0.115	-0.074	-0.	-0.027	-0.078	0.028	0.023	0.684	0.132	0.051	0.

Figure 4: Visualized one-step transition matrix of *Helas mon coeur*¹³

the least-squares method to all the state vectors of a single musical work, we can at least find the values for the matrix X , such that the above estimation is done as accurately as possible within the framework of the formula.¹¹ For every work in the dataset we shall find a different result, but we leverage these differences in the following sections.

2.4 Analyzing the Transition Matrix

We could consider this twelve by twelve transition matrix X to be a representation of the chromatic progression of the song. After all, it models how the chromatic activity changes from one tactus to the next. Every column in the matrix represents a single chroma activity in the previous step, and the values in the column represent by which chromae they are likely followed. As such, we can visually check which transitions are likely, given a certain (previous) active chroma.

For example, Figure 3 shows the matrix for *Helas mon coeur*, and Figure 4 shows this matrix using a heatmap visualization, for easy inspection. From the heatmap in Figure 4 we can immediately see that there are two extremes in the transition matrix;

¹¹ To find X , we solve the equation $T_{2:n} \cong X \cdot T_{1:n-1}$ such that $\|T_{2:n} - X \cdot T_{1:n-1}\|_2^2$ is minimized (T is a matrix of size $12 \times n$, containing the n state vectors as columns, e.g., as in Figure 2). The solution to this problem can be found by using the right-inverse matrix:

$$X = T_{2:n} \cdot T_{1:n-1}^T (T_{1:n-1} \cdot T_{1:n-1}^T)^{-1}$$

¹² The rows correspond to the prediction of a single chroma activity in the next step, while every column corresponds to a chroma activity in the previous step. Note that the rows are in reversed order (B, B♭, ... C), such that the correct order of the chromae is retained through the matrix multiplication.

the value in the *D* row, second to last column, is 2.399. This cell corresponds to the chromatic action from *C♯* to *D*. This means that—within the matrix-vector multiplication described above—the *C♯* activity in the previous step is multiplied by 2.399, which is then added to the results of the multiplications of every other pair in this row (for example, the previous *C* activity times 0.357 is also added), to predict the *D* activity in the next step. Obviously, because the value from *C♯* to *D* is so high compared to the other values in this row, the prediction for a *D* in the next step will (only) spike whenever there is *C♯* activity in the previous step. Moreover, as the value of 2.399 is so much higher than the other values in this column, it is very unlikely that any other chroma than *D* will follow, after a tactus with *C♯* active.

The value in the matrix in the *F♯* column, in the *D* row, is -0.855 . From this, we can infer that it is very unlikely that within *Helas mon coeur*, a tactus with *F♯* active is followed by a *D*. Whenever the state vector for a tactus indeed contains activity in the *F♯* pitch class, the positive value that indicates this *F♯* activity, is multiplied by this -0.855 in estimating the *D* activity of the next tactus. This will produce a negative result. Still; the products between the current state vector and the other values in the *D* row might increase the value a bit, but it will likely not be positive (which would indicate *D* activity).

Furthermore, from the example matrix in Figure 3, we can see that the cells in the *B* and *G♯* rows and columns are all equal to zero. This is a direct result of the fact that there are no notes of these pitch classes in the studied work. As such, some minimal global information about the work is present in the matrix. Note that we should not simply omit these rows and columns; they are likely to be non-zero in transition matrices of other works, so the zero values will definitely be relevant when comparing different works.

2.5 Modelling the Harmonic Progression of the Song more Precisely

Next, we try to increase the modelling performance of the state-space model. To this end, we extend the number of previous states which the model can see, as its input to predict what the next state will look like. As a first experiment, we extend this history to include the previous two states, where we previously had only one previous state. Nevertheless, the above formula that predicts the chroma activity of the following tactus changes depending on the order. With order two, it becomes:

$$\begin{bmatrix} \text{tactus}_{t+1} \end{bmatrix} = X \cdot \begin{bmatrix} \text{tactus}_t \\ \text{tactus}_{t-1} \end{bmatrix}$$

In this framework, the matrix *X* is now larger; the two vectors are concatenated to make a single vector of length 24. In order to still predict a single state vector (of length 12), the matrix *X* should be of size 12×24 .

Every value in the matrix *X* now links the activity of a chroma in the next step to the activity of a chroma in one of the two previous states. For this reason, the matrix now has 24 columns; one for every chroma and previous state. Again, using the least-

¹³ We can immediately see that there are two outliers; from *C♯* to *D*, and from *F♯* to *G*. This means that whenever the *F♯* pitch class is active in a tactus, it is almost guaranteed that in the next time-step, the tension in the *F♯* will be resolved into a *G*, likely sung by multiple voices.

Figure 5: Chroma activity distribution of *Helas mon cueur*⁴

squares method, we can find the values for the matrix X such that it predicts the next tactus as accurately as possible.

Partly because of this enlargement, the matrix X that describes how the next state is estimated, becomes more difficult to inspect manually. The main reason is that within the previous two states, there often are autocorrelations present; correlations between, for example, the same note in the previous tactus and the tactus before that. The least-squares method searches for the best way to estimate the next tactus, so if a better prediction can be made based on an older tactus, it might completely discount the existing correlation with a specific chroma in the previous step. This way, the existing correlation with the previous step is not included in the transition matrix. If we want to compare the transition matrices for higher-order models, we would have to keep this in mind, but also track the possible autocorrelations that are hidden in the search for the best next-state estimator.

As we shall see in Section 3.1, we use the matrix X as a vector representation for the whole song, in machine learning methods. These methods rely on the property that items of a similar class, or label (in our case, the composer), will have similar vector representations. But, because of these autocorrelations, only small changes in the signal (the musical work) can result in a big change in the vector representation. Consequently, the machine learning methods will have a hard time identifying truly similar items, and make inaccurate classifications as a result. This is the reason why we limit the number of steps we look back to a maximum of two states in this new set-up. In the next section, we compare the accuracy of the classification algorithms, and make an informed decision on exactly how many steps our model should look back.

Figure 5 shows how the prediction of the next tactus chroma distribution becomes more accurate when increasing the order of the model from one to two. We can see that the two-step predictions are much more accurate than the predictions of the first-order

model; the predictions of the first-order model in the middle hardly differ throughout the whole song, while the predictions of the second order model are more detailed, even from one step to the next. However, an accurate prediction of the next tactus is not the end goal of our experiment; it is the classification accuracy of the machine learning algorithms used in Section 3 which we value more highly.

The key idea is that this state transition matrix contains in some abstract sense the ideas and harmonic progressions contained in the songs. After all, they model both the short-term progressions in the music per chroma, as well as the presence of the individual pitch classes in the song. So, for every one of the 1248 works in the JRP dataset, we create such a transition matrix X . Numerical representations of uniform sizes—as we have here—can be easily used by machine learning algorithms to group items, to recognize overarching patterns in those representations.

3. Results

3.1 Composer Classification

We now use classifying algorithms that identify the composer of a work, solely based on its acoustic progression, by using the method we propose. Machine learning classification algorithms analyse the vector representations of items in a labelled dataset, and make inductions based on similar items (i.e. similar vector representations) with equal labels. In our case, we take the numbers of the state space model transition matrix of a work as the vector representation of the entire work (we simply concatenate the values in the transition matrix to one long vector). After all, this transition matrix encapsulates the harmonic progressions apparent in the song.

Before we apply the classifying algorithms to anonymous pieces (and suggest *new* attributions), we check how accurate the algorithms are on securely attributed works. To this end, we use cross validation.¹⁵ In cross validation, we divide the dataset into k sub-datasets, or folds. In turn, we leave one fold out of the dataset, train the classification algorithm on the rest of the data, and then measure its performance on the held-out fold. The measured classification accuracy of the algorithm is then the average accuracy over all of these folds.

We compare the accuracy between several encoding methods combined with different classification algorithms. To check if using higher-order models could improve the classification accuracy, we conduct our experiment with up to fifth-order models. To show the value of using the transition matrix technique, we also compare the results with a method where we simply take the correlation between the previous state and the current state. For every pair of chromae, we take the vector product of their activities in subsequent tacti.¹⁶ This also yields a 12×12 matrix, estimating how often a transition

¹⁴ On top, the actual chromae activity distribution is illustrated as a piano roll. The middle figure displays the prediction of a first-order model on this work; every state vector is based on the chroma activity of the previous state. Note that the first state cannot be estimated this way; there is no previous state for the model to estimate the next state with. In the same way, the first two states cannot be estimated in a second-order model, as can be seen in the bottom figure.

¹⁵ We used three-fold stratified cross validation, which means that we create three folds roughly equal size (277 items). When using stratified cross validation, the distribution of labels is roughly equal in each of the folds.

¹⁶ We define X here simply as $X = T_{1:n-1} \cdot T_{2:n}^T$.

Table 1. Composer classification accuracy under cross validation when classifying between the twenty composers in the JRP

	RandomForest ^a	XGBoost	Dummy
1st order	46.2%	51.0%	21.2%
2nd order	44.2%	44.7%	21.2%
3rd order	38.0%	38.5%	21.2%
4th order	38.5%	37.0%	21.2%
5th order	36.5%	37.5%	21.2%
1-step correlation	44.7%	45.7%	21.2%
4-step correlation	38.9%	32.7%	21.2%
jSymbolic representation	74.5%	78.4%	21.2%

^a This classification problem has twenty classes (the composers), of which the largest class (Pierre de la Rue) contains 21.2% of the items. Despite the high number of classes and a substantial class imbalance, we reach an accuracy is of 51.0% when we combine the 1st order model representation with the XGBoost classifier.

occurs, from each chroma to each chroma. This *correlation matrix* method is highly similar to the note transition probability technique of Simonton.

There exist multitudes of different classification algorithms, each working slightly differently internally, yielding different results. We use some of the most popular techniques for classification (K-nearest Neighbours, Ridge classification, Decision Trees, Support Vector Machines, etc.), but conclude that the XGBoost classifier¹⁷ and the Random Forest (with 100 trees)¹⁸ techniques consistently outperformed the other algorithms when classifying works to their composers. These are algorithms that are well-known for their classification accuracy, in other general classification problems. We also include a dummy classifier (i.e., a classifier which only predicts the most frequent class, in this case the composer Pierre de la Rue) to easily check if the algorithms could indeed identify any useful patterns in the dataset, in order to rise above the performance of a trivial classifier.

In Table 1 we summarize the results. The first-order method (where the representation of a piece is the transition matrix of its state-space model which can only look one step back) produces the best classifying accuracy. When combined with the XGBoost algorithms, it reaches an accuracy of 51.0% under cross validation. This means that, in 51% of the cases, the algorithm's best guess corresponds with the actual composer of the work. This seems low, but when we consider that the objective is to distinguish the pieces of twenty different composers purely based on the chromatic progressions present in the music, this is an acceptable percentage.

When we look at the other rows, we can see that the accuracy drops, the higher the order of the state-space model. Indeed, although the predictions of the state-space model itself might be better when they are of a higher order, it seems that their

¹⁷ Tianqi Chen and Carlos Guestrin, 'XGBoost: A Scalable Tree Boosting System', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, 13 August 2016 (San Francisco, 2016), 785-94.

¹⁸ Leo Breiman, 'Random Forests', in *Machine Learning* 45 (2001) 5-32.

Table 2: XGBoost classification output and accompanying probabilities for the ten works where the algorithm is most certain of its classification

JRP id	Work title	Predicted composer	Probability
Jos2720	<i>La belle se siet</i>	De la Rue	99.77%
Jos2912	<i>La Spagna</i>	De la Rue	99.75%
Jos9909	<i>In deficiendo</i>	Martini	99.70%
Jos2008	<i>Nunc dimittis</i>	De la Rue	99.69%
Jos9901d	<i>Missa Mi mi - Sanctus</i>	De la Rue	99.67%
Rue3013	<i>Incessament mon povre</i> ^a	De la Rue	99.66%
Jos2909	<i>Incessament mon povre</i>	De la Rue	99.66%
Ano3093	<i>Sil vous plaist bien</i>	Busnoys	99.61%
Jos0903b	<i>Missa Di dadi - Gloria</i>	De la Rue	99.43%
Jos9904	<i>Proch dolor</i>	De la Rue	99.37%

^a Note that the work *Incessament mon povre* was included twice in the dataset; once with the (insecure) label ‘de la Rue’, and once as ‘Josquin’.

Table 3: Probabilities for the unica of the Leuven Chansonnier in the dataset

JRP id	Work title	Predicted composer	Probability
Ano3225	<i>Helas mon cueur</i>	Busnoys	88.04%
Ano3226	<i>Oublie oublie</i>	De la Rue	32.00%
Ano3229	<i>Par malle bouche</i>	Busnoys	39.24%
Ano3230	<i>Escu dennuy</i>	Busnoys	53.46%
Ano3231	<i>Si vous voulez</i>	Busnoys	26.20%
Ano3232	<i>Ou beau chastel</i>	Josquin	29.11%

representing power is the highest when they are only of order one. Another possible cause of the decreasing accuracy might be the number of variables in the representing vector; every time we increase the order of the model, the representation contains 144 more variables than previously. As we only have a dataset with 832 labelled items (secure attributions), there is a substantial risk of overfitting; with almost as many variables as training set items, the classification algorithm is bound to find some correlations between some variable and the composer, whether this correlation constitutes a valid induction or not.

When we compare our method of representing a work, with a representation built by feature extraction through the jSymbolic toolkit, we see that the classification is much more accurate. Of course, the jSymbolic toolbox was built exactly for the analysis of early music, and extracts features that are handcrafted for this problem. Moreover, the toolbox extracts features that are derived from other aspects than the melodic, chromatic progressions in the music. Hence the representation contains more than only these aspects, and the resulting representing power is higher.

Now that we know exactly which setup yields the highest classification accuracy, we can reuse this setup, and finally apply the best classifier to the unlabelled items of the dataset: the compositions with uncertain, or anonymous attributions. We train a single XGBoost classifier to handle the first-order model representations on the complete JRP dataset, and let it estimate the probability for every piece to be written by every composer. The highest probabilities we found throughout all the possible work-composer pairings, are listed in Table 2.

Obviously, the classification probabilities are not always above 99%, as they are for the works in Table 3 (exactly the classifications with the highest probabilities are listed). For about half of the works, the classifier is only sure of its best guess at a rate of less than 50% (note that—when measuring the classification accuracy—this best guess is taken as the algorithm’s prediction, whether the prediction probability is higher than 50% or not).

Table 3, which contains the results and probabilities for the unica of the Leuven Chansonnier, shows a sample of these probabilities. Four of these works are classified as being by Busnoys, although the classification algorithm reports a rather low probability. Only for *Helas mon cuer*—the running example we have seen previously—is the classifier 88.04% sure. This means that the classifying model deems it of high probability that *Helas mon cuer* was written by Busnoys.

3.2 Unsupervised Analysis: Dimensionality Reduction

In this section, we shall use dimensionality reduction techniques to visualize the dataset in two dimensions, analogous to the experiments by Van Kranenburg (on the Bach dataset of fugues). Now that we know which representation method works best, we try to reduce this representation—with 144 variables—to a representation of two variables, which we can use as x and y coordinates in a scatterplot with every work. We shall see that—when using a slightly more advanced technique—we can create such a reduced representation, where the works of some authors are grouped together.

The Principal Component Analysis (PCA) algorithm¹⁹ is the best-known method of dimensionality reduction. Put simply, the PCA algorithm finds a small number of perpendicular directions in the dataset, such that the dataset is most spread out alongside these directions. The N perpendicular vectors that constitute these directions, are called the *principal components* of the dataset. When we express every item in the dataset by how much they align with these, we get a coarse view of how the items in the dataset are grouped, even if every item is expressed in a high-dimensional vector.

Figure 6 shows the result of PCA dimensionality reduction on the 1248 works in the JRP dataset. To create these figures, we first represented every work using the one-step transition matrix (to create a vector of size 144 for each work), and then transformed this 144-dimensional dataset to a two-dimensional one, using the PCA algorithm. The x and y coordinates in Figure 6 correspond to these two dimensions; every dot indicates

¹⁹ Svante Wold, Kim Esbensen, and Paul Geladi, ‘Principal Component Analysis’, in *Chemometrics and Intelligent Laboratory Systems* 2 (1987) 37–52.

a.

b.

Figure 6: Scatterplot visualizations of the JRP dataset, using Principal Component Analysis to display the one-step transition matrix-representations of the 1248 works on a 2D plane

- a. By genre
- b. By composer (only the eight most prevalent are shown)

Figure 7. UMAP representation of the works in the JRP dataset, using the first-order state-space model transition matrix representation²¹

- a. By genre
- b. By author

a work, and shows how much a work aligns with the two *principal components* of the dataset.

Unfortunately, the PCA algorithm cannot find any semantically meaningful correlation in the dataset (using only two principal components); all of the works in the dataset are mixed together, regardless of their genre (the left side of Figure 6), or their composer (the right side). This means that—in the way we represented every work in the dataset—by the direction in which the representations linearly differ most (along the principal components), there is no correlation between the harmonic progression on one side, and the genre or composer on the other side.

We can also try other techniques for dimensionality reduction. The *Uniform Manifold Approximation and Projection*²⁰ (UMAP) algorithm works differently from the PCA algorithm in that it does not look for a linear projection of the dataset. Instead, the UMAP algorithm looks for dense neighbourhoods in the highly dimensional space, and then tries to lay out these neighbourhoods in a lower dimensional space, while trying to keep the position of the items relative to each other consistent with the original data. This way, the mathematical link between the reduced, lower-dimensional representation of an item and the original vector becomes rather fuzzy; the placement of an item in the lower-dimensional space is based on items in its vicinity in the original high-dimensional space.

When we apply the UMAP algorithm to the JRP in the same way as we did the PCA algorithm in Figure 6, we get the results visualized in Figure 11. We can see that—contrary to the PCA algorithm—the UMAP algorithm does manage to identify meaningful clusters of items in the dataset, specifically when comparing the positions of works in the dataset with their composers.

On the right side of Figure 11, some clusters are created of works with the same author. This is another indication that renaissance composers can be algorithmically separated solely by the harmonic progressions in their works, even in an unsupervised setting. Josquin, La Rue, and Martini, in particular, have some works which the UMAP algorithm clusters closely together. It could be interesting to isolate these works, and compare them to the other works of the same author.

Conversely, we can see that UMAP also does not manage to find a difference between the genres in the dataset (song, mass, motet). This would indicate that—considering the harmonic progressions—there are no inherent differences of musical writing style between the genres, or at least no differences that can be explained in a two-dimensional plot.

We repeated all of these experiments using the features extracted by jSymbolic, but this did not cluster the composers as well as in Figure 11. This is in contrast to the results in Section 3.1, where the classification algorithms did manage to separate the works by author. It is possible that the machine learning algorithms—supervised or otherwise—need the correct labels in the case of the jSymbolic features, to cluster the item together in a semantically meaningful way. It is possible that some of the features it extracts really are completely uncorrelated to the composer of a work, which could

²⁰ Leland McInnes, John Healy, and James Melville. 'UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction', arXiv preprint (2018). <<https://arxiv.org/pdf/1802.03426.pdf>> (accessed on 28 September 2020).

²¹ We can see that the UMAP algorithm does manage to cluster some chunks of works, specifically of La Rue, Josquin, Busnoys, and Martini.

blur the clustering capacity of the dimensionality reduction techniques. The addition of the correct label (when classifying the works) apparently helped the classification algorithms identify the features correlate with a certain composer's writing style.

We also used Neighborhood Component Analysis for dimensionality reduction, but no combination of representation and dimensionality reduction techniques separated the composers very well. Likely due to the fact that the data is not linearly separable, the PCA algorithm—which creates a linear mapping—could not isolate the works very well by composer. The UMAP algorithm works in a non-linear way, and finds some common patterns between composers, which it could exploit to better arrange the works on a 2D plane.

4. Conclusion

We have here presented a novel way to study symbolic music algorithmically. We fit a musical piece into the framework of state-space models by defining music as discrete time signals, and used the transition matrices of these state-space models as their numerical representations in machine learning algorithms. We established that we could create meaningful classification predictions for the music in the JRP database, achieving a good classification accuracy under cross validation, even though we limited the representation of music to be a sequence of chroma activity per tactus.

We experimented with the order of the state-space models, yet higher order models resulted in a lower accuracy under cross validation. We explained that this could be the result of cross-correlated data, or overfitting due to too many variables. Using the most accurate setup we found using our methods, we trained a single, final classifier to handle insecurely attributed works in the JRP database. For some pieces, the classification algorithm was very certain of a specific label, yet for most of the works, it was not very conclusive.

Given the output classification probability of the algorithm, and the consideration that these predictions were only based on the melodic progression of the music, the attributions made by our classification algorithms are useful in authorship discussions of early music repertoire. For example, in the case of the Leuven Chansonnier unica in the corpus, it was reasonably sure that *Helas mon cueur* should be attributed to Busnoys.

As a final experiment, we tried to explore the dataset using dimensionality reduction techniques. Using our method for representing musical works, and UMAP for dimensionality reduction, we created a 2D visualization of the landscape of the corpus, that in itself separated the works of some composers, based solely on the melodic progressions of their repertoire.

We have published the Python programming code to perform all of these experiments and generate the resulting illustrations.²² This code can be reused to apply the same methods to other domains, as the results in this paper show that a purely melodic analysis of Renaissance music using mathematical techniques is not only possible, but can be useful in attribution discussions in early music repertoire.

²² The source code can be found at <https://github.com/geelenb/beat_aligned_chromagrams>, specifically the `jrpy.py` file.

Appendix. Transcription of *Hélas mon cœur tu m'occiras*

He - las mon cœur tu moc - ci - - - ras
Tous - jours lan - guir si me fe - - - ras
In - ces - sa - ment tu lar - moi - - - ras

He - las mon cœur tu moc - ci - - - ras
Tous - jours lan - guir si me fe - - - ras
In - ces - sa - ment tu lar - moi - - - ras

He - las mon cœur tu moc - ci - - - ras
Tous - jours lan - guir si me fe - - - ras
In - ces - sa - ment tu lar - moi - - - ras

9

Quant des da - mes de - par - ti - ras
Et ma mort en pour - cha - ce - ras
Tant plus que les es - lon - gne - ras,

Quant des da - mes de - par - ti - ras
Et ma mort en pour - cha - ce - ras
Tant plus que les es - lon - gne - ras,

Quant des da - mes de - par - ti - ras
Et ma mort en quit - ce - ras
Tant plus que les es - long - gne - ras,

16

En qui d'on - neur as veu l'es - li - te.
 Af - fin que de moy tu soiz quit - te. &
 Dont mes en rien ne me des - li - te;

En qui d'on - neur as veu l'es - li - te.
 Af - fin que de moy tu soiz quit - te. &
 Dont mes en rien ne me des - li - te;

En qui d'on - neur as veu l'es - li - te.
 Af - fin que de moy tu soiz quit - te. &
 Dont mes en rien ne me des - li - te;

24

Lors se - ra ta vi - e maul - di - te.
 Vien - gne vers moy la mort des - pi - te.

Lors se - ra ta vi - e maul - di - te.
 Vien - gne vers moy la mort des - pi - te.

Lors se - ra ta vi - e maul - di - te.
 Vien - gne vers moy la mort des - pi - te.

33

Car a - pres en brief temps mour - ras.
 Aus - si - tost qu'es - lon - gn' se - ras.

Car a - pres en brief temps mour - ras.
 Aus - si - tost qu'es - lon - gn' se - ras.

Car a - pres en brief temps mour - ras.
 Aus - si - tost qu'es - lon - gn' se - ras.

Abstract

In this paper, we propose a novel clustering method to represent and compare works of symbolic music, in particular the music in the database of the Josquin Research Project.²³ We also study the results of our methods on the database as a whole, as well as the results for some specific cases. The methods that we propose revolve around modelling the activity of the twelve pitch classes (i.e., chromae) over time. We suggest that by modelling this activity, we can capture the harmonic progressions in the music. By converting the models to a vector representation, we can analyse the works in the dataset using clustering techniques from machine learning. These techniques can be used to group, visualize, and classify works in the corpus. This way, we suggest attributions for insecurely attributed pieces in the dataset, based solely on the harmonic progressions contained in the songs. We also apply our final classification models to the unica of the Leuven Chansonnier, to give the first suggestions for the attribution of these works from the world of computational musicology. We then compare our methods to some existing methods for analyzing symbolic music, and find that our method is slightly worse at identifying composers of Renaissance works. This result was not completely unexpected, as the features we extract are based only the musical progressions present in the music. Finally, we use the best vector representation we devised together with unsupervised dimensionality reduction algorithms to create a scatterplot visualization of the entire Josquin Research Project dataset.

²³ Jesse Rodin, 'Josquin Research Project', Stanford University (2010), <<https://josquin.stanford.edu/>> (accessed 16 September 2020).