Feature-Based Analysis of Haydn String Quartets

Lawson Wong

5/15/12

1 Introduction

When listening to multi-movement works, amateur listeners have almost certainly asked the following situation[1]: Am I still listening to the same movement? If not, which movement is being played now? Is this still the same work? These questions motivate the key questions in this study. First, is it possible to distinguish between different movements of the same multi-movement work? Further, can one identify which movement an excerpt comes from? That is, for the four-movement works that will be studied here, are there defining characteristics that distinguish between movement types? Second, given previously-heard excerpts of movements belonging to the same multi-movement work, does the current excerpt belong to the same work? More informally, if I listened attentively to the first movement and then dozed off in the second and third, can I identify, when waking up in the fourth movement, that I am still listening to the same work? Through several feature-based classification tasks, I have found that movement type classification is reasonably achievable with a slight modification in the problem formulation. However, identification of a movement using features from other movements within the same multi-movement work is more challenging.

To perform the above analysis, a corpus of multi-movement works is necessary. The `music21` corpus[2] contains many works, in particular many string quartets by Haydn, Mozart, Beethoven, and other composers. String quartets are a good initial object of study due to their well-defined four-movement structure (with some exceptions), as will be seen in the study of movement type classification. For this study, I will look at Haydn's string quartets. Haydn wrote 68 string quartets, and in doing so essentially defined the structure of the classical string quartet. I will use a feature-based approach by extracting features from a subset of Haydn's string quartets, using the feature extraction classes of `music21` [2]. Details on the dataset will be described next, followed by discussion of the classification tasks.

[1] This has certainly happened to myself for an embarrassingly many number of times.

[2] `http://mit.edu/music21/doc/html/referenceCorpus.html`

2 Description of data

To ensure a feasible study for the given time constraints, I used a small dataset of 10 Haydn string quartets. The list of works is in the appendix. Haydn composed string quartets throughout his life, resulting in many opuses of string quartets, usually six at a time. I took 10 six-quartet opuses and used the first string quartet in each opus for my dataset, spanning the years 1771-1799. Haydn in fact wrote two opuses earlier in 1762-1765, but these are five-movement works and so were not included in the corpus. I will return to opus 1 later.

The `music21` features include many of the over 100 features from `jSymbolic` [3], as well as some new features. I chose a relevant subset of 56 features for this study. The features cover aspects from melody (e.g., interval histograms), pitch (e.g., pitch prevalence and histograms), and rhythm (e.g., note durations, time signature). The full list of features used can be found in the appendix. Since many of these features have vector values, when fully expanded this gives 365 numbers in total for each excerpt analyzed. To more accurately model the majority of listeners who do not have cannot recognize absolute pitch, I also considered removing 5 features that require this ability (e.g., pitch class histograms). The omitted features are also listed in the appendix. This 'NoPitch' setting gives a total of 190 numbers for each excerpt.

Since there are so many features compared to the amount of data in the corpus (there are only 10 pieces of each movement type!), it was necessary to create more data points. Moreover, since many of the features are quite local (intervals, note durations), it seems wasteful to extract features from an entire movement when its representational power does not go beyond a few measures. Hence each movement was split into non-overlapping segments of length 4 and 16 (two parameter settings); leftover measures were discarded. This approach gave a more reasonable dataset size as shown by the numbers in Table 1. Notice that although there are 10 pieces of each movement type, movement 4 has a tendency to be longer and hence the greater amount of data. (Because of this, movement 4 is the majority baseline class for movement type classification.) Although there are still more features than desirable, I attempt to avoid overfitting by using appropriate classification techniques. The interpretation of this approach is in the spirit of ensemble of weak learners: when predicting for a new excerpt, features of short segments can be extracted to give a 'weak' prediction, then predictions from multiple segments can be combined to vote for a 'strong' prediction.

	Movement 1	Movement 2	Movement 3	Movement 4	Total
Frag-4	378	192	225	501	1296
Frag-16	90	44	53	122	309

Table 1: Number of data points (per movement) in corpus.

3 Movement type classification

The first task I studied was one can use the features described above to determine which movement an excerpt came from. From a machine learning standpoint, this is essentially a multi-class classification problem, with one class for each of the four movements.

There are many possible classifiers for multi-class classification, including schemes for converting binary classifiers into a multi-class setting. For the tasks in this study, I used k-nearest-neighbors (k-NN) and random forests, each with three different parameter settings. Both classifiers are inherently multi-class, and have been implemented in the Python package `Scikit-learn` [4]. k-NN is a simple baseline approach that classifies a given data point to the majority class of its k nearest neighbors (in feature space with Euclidean metric); ties are broken randomly. $k = 3, 5, 10$ were used (3NN, 5NN, 10NN respectively in tables below).

Random forests (RF) [1] is a relatively recent classification method that combines the flexibility of decision trees with the robustness of ensemble methods ('weak' learners that vote to give a 'strong' learner). A RF consists of a collection of decision trees (hence *forest*), but randomized such that each decision tree is built with a small random subset of the features. To avoid overfitting, each tree in the forest is usually quite shallow, and hence each tree's performance is typically worse than a single decision tree fit using all available features. However, because many 'weak' trees are combined by voting, specific errors tend to be averaged out, while generalization performance tends to be significantly better due to lower overall overfitting (precise characterization of this empirical finding is still an active area of research). For this study, I used three parameter settings: 50 depth-10 trees (RF10), 100 depth-5 trees (RF5), and 200 depth-3 trees (RF3).

A leave-one-out cross-validation (LOOCV) scheme was used. Each hold-out fold consisted of all movements of a string quartet. For example, `opus17no1` was removed, then each classifier was trained on the remaining 9 string quartets; then their performance was tested on `opus17no1` features. This is then repeated for each of the 10 string quartets. Average results per movement and overall are reported in Table 2. Random forests (especially RF10) performs quite well, although it is significantly worse at predicting movements 2 and 3.

Movement	3NN	5NN	10NN	RF10	RF5	RF3	Majority
1	0.66	0.65	0.70	**0.87**	0.79	0.58	0
2	**0.21**	0.17	0.13	0.20	0.06	0.01	0
3	0.12	0.16	0.14	**0.61**	0.58	0.21	0
4	0.49	0.51	0.56	0.79	0.84	**0.88**	1
Average	0.37	0.37	0.38	**0.62**	0.56	0.42	0.39

Table 2: LOOCV performance for 4-class movement type classification.

Movement	Frag4-NoPitch	Frag4-AllFeats	Frag16-NoPitch	Frag16-AllFeats
1	**0.87**	0.80	0.84	0.78
2	0.20	0.19	0.29	**0.34**
3	**0.61**	0.50	0.50	0.52
4	0.79	0.79	**0.82**	0.81
Average	**0.62**	0.57	0.61	0.61

Table 3: LOOCV performance for RF10 across different parameter settings.

Table 3 shows LOOCV performance of the RF10 across different parameter settings. Somewhat surprisingly, the NoPitch setting tends to work better (most likely because it does not overfit as much), and the use of either segment length setting (4- or 16-measure) does not affect the task much. Since 4-measure segments without pitch features achieves slightly better average performance, only the results for these parameter settings are shown in the previous table as well as in results below.

The results on the 4-class classification task given in Table 2 were modest (and significantly beat the baseline majority and k-NN classifiers), but the poor performance on movements 2 and 3 were unsatisfactory. A closer look at the errors reveal the main source of confusion. Table 4 shows the confusion matrix of the RF10 classifier. This is the 4×4 matrix of (true class, predicted class) instances. For example, the entry in row 2, column 3 shows that there were 91 data points from movement 2 (true class 2) that were incorrectly predicted as belonging to movement 3 (predicted class 3). Since there are only 192 data points in movement 2 (see Table 1), this is a major souce of errors! In particular, this error type is more prevalent that correctly classifying a movement 2 excerpt as class 2. Movement 3 excerpts also suffer from a similar problem, though not as profound.

The confusion matrix suggests that there is significant confusion between movements 2 and 3, which is intuitive since inner movements tend to be less strict than the outer ones. To explore this further, I considered lumping classes 2 and 3 together. Although this does not solve the initial problem, perhaps there is no clustering-based reason to separate the two.

True	Predicted class			
class	1	2	3	4
1	308	1	0	69
2	24	36	91	41
3	0	54	136	35
4	39	22	7	433

Table 4: Confusion matrix for RF10 on 4-class movement type classification.

Movement	3NN	5NN	10NN	RF10	RF5	RF3	Majority
1	0.64	0.63	0.66	**0.86**	0.81	0.61	0
2	0.40	0.47	0.40	**0.77**	0.72	0.71	0
3	0.31	0.38	0.40	**0.90**	0.90	0.90	0
4	0.49	0.48	0.50	0.79	0.83	**0.88**	1
Average	0.46	0.49	0.49	**0.83**	0.81	0.78	0.39

Table 5: LOOCV performance for 3-class movement type classification.

Movement	3NN	5NN	10NN	RF10	RF5	RF3	Majority
4-class	0.37	0.37	0.38	0.62	0.56	0.42	0.39
3-class	0.46	0.49	0.49	0.83	0.81	0.78	0.39
Improvement	0.09	0.12	0.11	0.19	0.25	0.36	0

Table 6: Performance comparison between 4-class (Table 2) and 3-class (Table 5).

Performing the new 3-class classification task (with a new combined 2/3 class) shows significantly improved performance as expected. Performance for movements 2 and 3 have increased greatly (by construction), as seen in Table 5, while performance for the outer movements is essentially unchanged. The improvement in average performance for each classifier is shown in Table 6. Interestingly, although the 2/3 class combines two previous classes, the baseline majority classifier still outputs class 4 and hence has no improvement. The numbers in the new confusion matrix in Table 7 show that the new RF10 classifier has lumped the movement 2 and 3 numbers together, with little improvement for the others.

Although the results of the 3-class classification task are mostly expected, the new performance figures are much more satisfactory and suggest that the movement-type classification task is feasible with a feature-based approach. The only compromise one must make is that the inner movements tend to be quite similar and even mixed, so it is inherently difficult to separate the two using the currently used features. There may be other features that can distinguish the two; for example, the tempo marking and pace of the second movement tends to be slower than the rest (hence usually referred to as the 'slow' movement).

True	Predicted class		
class	1	2/3	4
1	308	2	68
2/3	24	333	60
4	42	29	430

Table 7: Confusion matrix for RF10 on 3-class movement type classification.

4 Further explorations

One advantage of using random forests is that, by comparing the individual decision tree performance with the features chosen for that tree, a measure of feature importance can be found. Below are the 10 most important features identified by RF10 (this list is essentially the same as that found by all forests across all parameter settings):

```
Initial_Time_Signature_0
Triple_Meter
Initial_Time_Signature_1
Compound_Or_Simple_Meter
Minimum_Note_Duration
Tonal_Certainty
Maximum_Note_Duration
Pitch_Variety
Staccato_Incidence
Unique_Note_Quarter_Lengths
```

It is immediately clear that the classifiers generally distinguish movements by their time signature and meter, and to some extent their rhythm (note duration) and pitch variety. A depth-3 decision tree using only (a subset of) these 10 features is shown in Figure 1 and illustrates the choices made to distinguish between the three movement types. It is interesting to see that, for example, class 2/3 tends to be triple meter (upper numeral is 3).

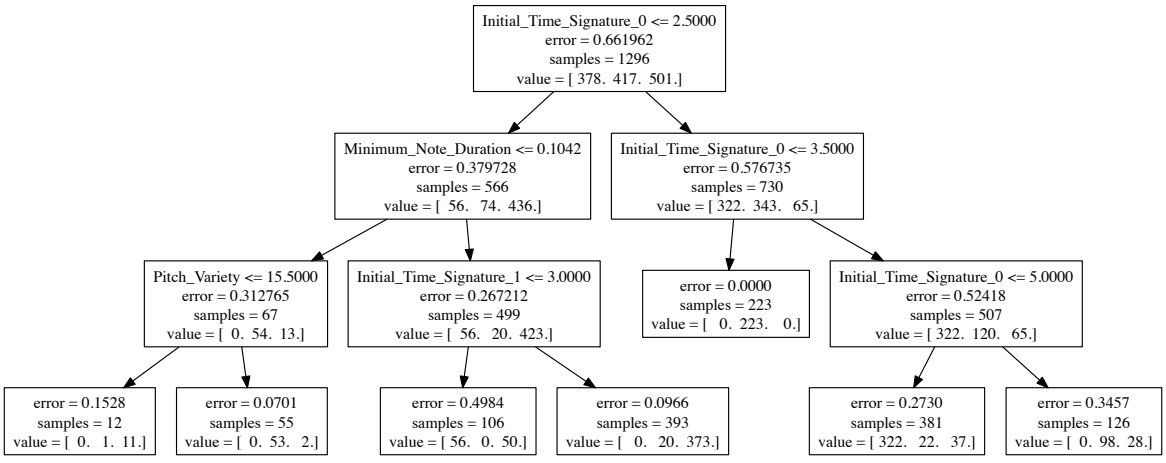

Figure 1: Depth-3 decision tree trained using the 10 most important features. The first line in each inner node indicates the choice at each node, going to the left child if true and right if false. The last line in each node indicates the number of data points in each class at that node *before* the node's splitting decision (if any) is made.

Work	Movement 1			Movement 2			Movement 3			Movement 4			Movement 5		
	1	2/3	4	1	2/3	4	1	2/3	4	1	2/3	4	1	2/3	4
opus1-no0			1		1				1		1				1
opus1-no1		1			1		.1	.7	.2		1				1
opus1-no2		1			1		.5	.5			1				1
opus1-no3		1			1				1		1			1	
opus1-no4		1			1		.9	.1			1				1
opus1-no6		1			1			.3	.7		1				1

Table 8: Prediction (proportion of segment votes) of movement type by RF10 on Opus 1.

Another interesting task that could be performed with the available features and classifiers is the characterization of non-standard string quartets. As mentioned earlier, Haydn wrote a set of 6 five-movement string quartets (Opus 1) early in his life around 1762. Since they have five movements, it is interesting to see what 'type' they belong to, of the three types that have been identified in the previous section. Features were extracted in a similar fashion for all six string quartets, and their movement type was predicted by RF10. The proportion of votes (over four-measure segments in each movement) are shown in Table 8.

The results show that each movement, apart from those in movement 3, are remarkably consistent, even across different string quartets in the same opus. Looking at the scores, movements 2 and 4 tend to be denoted as minuets, hence have triple meter and are all classified as class 2/3. Interesting, the first movement usually also had triple meter, hence also giving a class 2/3 prediction. It appears that the usual common time/ cut time signature of the first movement was a later development. Movement 4, similar to later eras, was usually in 2/4 time. Movement 3 has the greatest uncertainty, and perhaps can be seen as the 'extra' movement of the four (with respect to the later standard four-movement structure).

5 String quartet identification

The second task of string quartet identification was also attempted, but time constraints will not allow a detailed description (sorry!). Since there are 10 classes for our dataset (and more for the Haydn corpus), I framed this task as a simpler binary classification problem of identifying movements between two string quartets. Still, this was a challenging task for the feature-based approach described above, especially if without pitch features. In this scenario, all features essentially have performance similar to the majority / random baseline, indicating that the features do not distinguish individual string quartets well. Including pitch features significantly improves the performance overall, since they essentially identify the key, which is generally different between pairs of string quartets.

6 Conclusion

I have explored the use of `music21` features in the two tasks related to the analysis of string quartets. Movement-type classification performance was quite satisfactory using random forest classifiers, although it was difficult to distinguish between the inner movements. Using the feature importances determined by these classifiers, I found that time signature and rhythmic / pitch variety features tend to distinguish movement types. This information was also used to analyze the non-standard five-movement string quartets of Haydn's Opus 1, suggesting some significant differences in their structure compared to his later works that defined the classical string quartet structure. The same approach was also attempted on the task of distinguishing between excerpts of different string quartets, but without much success if absolute pitch-based features (that identified the key) were not included.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Michael Scott Cuthbert, Christopher Ariza, and Lisa Friedland. Feature extraction and machine learning on symbolic music using the music21 toolkit. In Anssi Klapuri and Colby Leider, editors, *ISMIR*, pages 387–392. University of Miami, 2011.

[3] Cory McKay. *Automatic Music Classification with jMIR*. PhD thesis, McGill University, Canada, 2010.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

A Corpus (Haydn four-movement string quartets)

opus17no1
opus20no1
opus33no1
opus50no1
opus54no1
opus55no1
opus64no1
opus71no1
opus76no1
opus77no1

B Features used

* = from `music21.native`, otherwise from `jSymbolic` ([3]); + = not used in NoPitch setting

```
*K  1 TonalCertainty
 M  1 MelodicIntervalHistogramFeature
 M  2 AverageMelodicIntervalFeature
 M  3 MostCommonMelodicIntervalFeature
 M  4 DistanceBetweenMostCommonMelodicIntervalsFeature
 M  5 MostCommonMelodicIntervalPrevalenceFeature
 M  6 RelativeStrengthOfMostCommonIntervalsFeature
 M  7 NumberOfCommonMelodicIntervalsFeature
 M  8 AmountOfArpeggiationFeature
 M  9 RepeatedNotesFeature
 M 10 ChromaticMotionFeature
 M 11 StepwiseMotionFeature
 M 12 MelodicThirdsFeature
 M 13 MelodicFifthsFeature
 M 14 MelodicTritonesFeature
 M 15 MelodicOctavesFeature
 M 17 DirectionOfMotionFeature
 M 18 DurationOfMelodicArcsFeature
 M 19 SizeOfMelodicArcsFeature
 P  1 MostCommonPitchPrevalenceFeature
```

```
 P  2 MostCommonPitchClassPrevalenceFeature
 P  3 RelativeStrengthOfTopPitchesFeature
 P  4 RelativeStrengthOfTopPitchClassesFeature
 P  5 IntervalBetweenStrongestPitchesFeature
 P  6 IntervalBetweenStrongestPitchClassesFeature
 P  7 NumberOfCommonPitchesFeature
 P  8 PitchVarietyFeature
 P  9 PitchClassVarietyFeature
 P 10 RangeFeature
+P 11 MostCommonPitchFeature
 P 12 PrimaryRegisterFeature
 P 13 ImportanceOfBassRegisterFeature
 P 14 ImportanceOfMiddleRegisterFeature
 P 15 ImportanceOfHighRegisterFeature
+P 16 MostCommonPitchClassFeature
+P 19 BasicPitchHistogramFeature
+P 20 PitchClassDistributionFeature
+P 21 FifthsPitchHistogramFeature
 P 22 QualityFeature
*Q 11 UniqueNoteQuarterLengths
*Q 12 MostCommonNoteQuarterLength
*Q 13 MostCommonNoteQuarterLengthPrevalence
*Q 14 RangeOfNoteQuarterLengths
 R 15 NoteDensityFeature
 R 17 AverageNoteDurationFeature
 R 19 MaximumNoteDurationFeature
 R 20 MinimumNoteDurationFeature
 R 21 StaccatoIncidenceFeature
 R 22 AverageTimeBetweenAttacksFeature
 R 23 VariabilityOfTimeBetweenAttacksFeature
 R 24 AverageTimeBetweenAttacksForEachVoiceFeature
 R 25 AverageVariabilityOfTimeBetweenAttacksForEachVoiceFeature
 R 30 InitialTempoFeature
 R 31 InitialTimeSignatureFeature
 R 32 CompoundOrSimpleMeterFeature
 R 33 TripleMeterFeature
```