Classical Composer Authorship:

By Alexander Taoultsides

May 3, 2013

Methods:
Joseph Haydn, W.A. Mozart, and Ludwig Van Beethoven are composers from the classical era spanning from the mid 1700s to the early 1800s. This Final project uses machine learning to discover whether or not a musical composition can be attributed to a composer by only using the frequency of individual notes and chords in the piece.

The three composers were chosen because of the similarity of their composing styles and the close time frame of when they were composing. Midi files were used in order to analyze the musical compositions from each of the three composers. In order to read the midi files into a python script, the middle man software gnmidi was used to convert the midi file into a readable .csv file. The pieces were written in different keys which could mean a variation in the frequency of notes and chords played. If that is the case with the latter then the final results would be skewed, so all pieces would need to be transposed to one key. The key of C-major was chosen for pieces in the major key and a-minor for the pieces in the minor key. C major and a minor both contain the same notes but the results may become skewed because the A note may be played more in a-minor because it is the tonic note of the scale. The tonic note is a beginning note and name sake of a key, and every classical piece will end on the tonic.

All pieces were in piano sonata form and were played and composed on a piano so the tonal range would be the same. Even though Beethoven wrote music in his later career with a piano with more keys, it did not show up in the final results. The piano sonata is composed of 3 to 4 parts, and each individual part was used in the experiments. There were 68 parts in total, split amongst each composer. The Beethoven folder has 27 parts, the Mozart folder has 21 parts and the Haydn folder has 20 parts. All folders were "globbed" in the Python script and each file was parsed to remove any information except the name of the file and notes and chords. All notes were counted in the file and placed into a hash. The time each note gets played indicates whether or not it is a part of a chord, and is then concatenated into one hash key. The end result after using the Python script is a matrix of 6944 x 68. Every note or chord that does not get played in a piece is filled in with a 0 in the spot. The features run across the first row in the .csv file and the name of the piece and composer is written down the first column. The frequency of each note and chord is across each row to be ready for the CART machine learning algorithm using the R language script provided for the Federalist Papers experiments. After the results of the first group of compositions, a second group was created. All minor key pieces written by Beethoven were replaced with pieces transposed to the key of C-major so that all compositions in the study were on equal footing.

90 percent of the data was used to train the CART algorithm for guessing the last 10 percent. This was performed at 10 different folds so that the training portion included the test pieces for different tests. 6 pieces were tested per fold.

RESULTS: CART predicted the correct composer for each piece an average of 71 percent of the time after 25 runs per dataset. This proves that it is possible to predict a composer based solely on the amount of notes or chords the composer uses.

Figure 1: Shows a 68 percent average for predicting the correct composer for the set with the Minor key compositions written by Beethoven. Each test is a fold with 90 percent training and 10 percent test pieces.

```
 [1] "--- Test -------------------------------------"
[1] "WRONG(actual:pred) - [ 30 H ] -  M"
[1] "WRONG(actual:pred) - [ 68 M ] -  H"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "----------------------------------------------"
Pause...
[1] "--- Test -------------------------------------"
[1] "WRONG(actual:pred) - [ 49 M ] -  H"
[1] "WRONG(actual:pred) - [ 12 B ] -  M"
[1] "WRONG(actual:pred) - [ 34 H ] -  M"
[1] "WRONG(actual:pred) - [ 19 B ] -  M"
[1] "Number correct is:  2"
[1] "Number wrong is:  4"
[1] "Percent correct is 33.33"
[1] "----------------------------------------------"
Pause...
[1] "--- Test -------------------------------------"
[1] "WRONG(actual:pred) - [ 56 M ] -  H"
[1] "Number correct is:  5"
[1] "Number wrong is:  1"
[1] "Percent correct is 83.33"
[1] "----------------------------------------------"
Pause...
[1] "--- Test -------------------------------------"
[1] "WRONG(actual:pred) - [ 10 B ] -  M"
[1] "WRONG(actual:pred) - [ 9 B ] -  H"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "----------------------------------------------"
Pause...
[1] "--- Test -------------------------------------"
[1] "WRONG(actual:pred) - [ 41 H ] -  B"
[1] "Number correct is:  5"
[1] "Number wrong is:  1"
[1] "Percent correct is 83.33"
[1] "----------------------------------------------"
```

```
Pause...
[1] "--- Test ---------------------------------------"
[1] "WRONG(actual:pred) - [ 58 M ] -  B"
[1] "WRONG(actual:pred) - [ 15 B ] -  H"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test ---------------------------------------"
[1] "WRONG(actual:pred) - [ 17 B ] -  M"
[1] "WRONG(actual:pred) - [ 65 M ] -  H"
[1] "WRONG(actual:pred) - [ 57 M ] -  B"
[1] "Number correct is:  3"
[1] "Number wrong is:  3"
[1] "Percent correct is 50.00"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test ---------------------------------------"
[1] "WRONG(actual:pred) - [ 60 M ] -  H"
[1] "WRONG(actual:pred) - [ 18 B ] -  H"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test ---------------------------------------"
[1] "WRONG(actual:pred) - [ 26 B ] -  H"
[1] "Number correct is:  5"
[1] "Number wrong is:  1"
[1] "Percent correct is 83.33"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test ---------------------------------------"
[1] "WRONG(actual:pred) - [ 14 B ] -  M"
[1] "Number correct is:  5"
[1] "Number wrong is:  1"
[1] "Percent correct is 83.33"
[1] "-----------------------------------------------"

> valid <- valid / k                    #calculate the average correctness of all folds
> print(valid)                          #print it out
[1] 0.6833333
```

Figure 2: Shows another set of tests with using the same R script to run the CART algorithm. This set uses the C-Major only pieces with a total of 67 parts instead of 68. These were 90/10 splits of training to test data. This run faired better than the last for an average of 73 percent correctness.

```
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 10 M ] -  B"
[1] "Number correct is:  5"
[1] "Number wrong is:  1"
[1] "Percent correct is 83.33"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 60 B ] -  H"
[1] "WRONG(actual:pred) - [ 52 B ] -  H"
[1] "WRONG(actual:pred) - [ 20 M ] -  H"
[1] "Number correct is:  3"
[1] "Number wrong is:  3"
[1] "Percent correct is 50.00"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 49 B ] -  M"
[1] "WRONG(actual:pred) - [ 39 H ] -  M"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 37 H ] -  M"
[1] "WRONG(actual:pred) - [ 64 B ] -  H"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 11 M ] -  B"
[1] "WRONG(actual:pred) - [ 13 M ] -  H"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "-----------------------------------------------"
Pause...
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 16 M ] -  H"
[1] "WRONG(actual:pred) - [ 12 M ] -  H"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "-----------------------------------------------"
```

```
Pause...
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 54 B ] -  M"
[1] "Number correct is:  5"
[1] "Number wrong is:  1"
[1] "Percent correct is 83.33"
[1] "----------------------------------------------"
Pause...
[1] "--- Test --------------------------------------"
[1] "Number correct is:  6"
[1] "Number wrong is:  0"
[1] "Percent correct is 100.00"
[1] "----------------------------------------------"
Pause...
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 8 M ] -  H"
[1] "WRONG(actual:pred) - [ 22 H ] -  B"
[1] "Number correct is:  4"
[1] "Number wrong is:  2"
[1] "Percent correct is 66.67"
[1] "----------------------------------------------"
Pause...
[1] "--- Test --------------------------------------"
[1] "WRONG(actual:pred) - [ 33 H ] -  M"
[1] "Number correct is:  5"
[1] "Number wrong is:  1"
[1] "Percent correct is 83.33"
[1] "----------------------------------------------"
Pause...
>
> valid <- valid / k                    #calculate the average correctness of all folds
> print(valid)                          #print it out
[1] 0.7333333
```

R created dendrograms after each run indicating the chord and note splits and buckets to which each composition fell into.

Figure 3 shows a dendrogram of the dataset using compositions in a-minor. The buckets are designated by the last name Initials for each composer. If a bucket has higher number than the other buckets then the split goes to that respective composer.

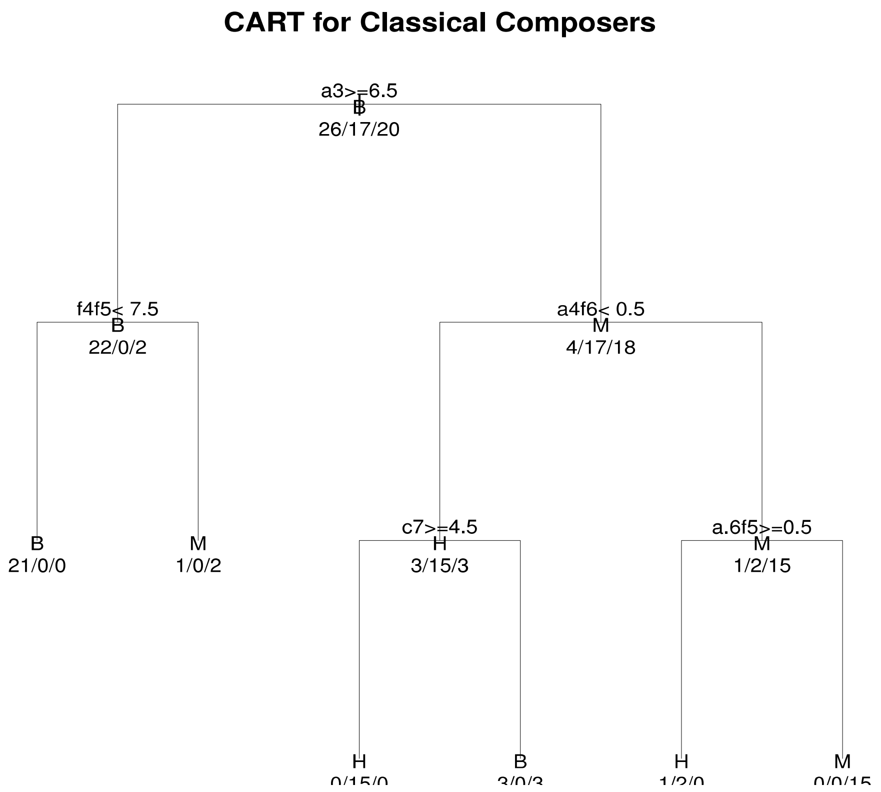

Figure 3: Rplot Dendrogram for CART composer authorship including the a-minor key pieces.

This example shows most of the Beethoven pieces splitting at pieces that contain less than 7.5 occurrences of the chord f4f5. The leaf nodes that indicate all numbers going to one composer indicate a strong preference in each composers musical style to use or not use those chords. Figure 4 shows a very similar looking dendrogram using the dataset with only compositions in C-major. This time most of the Beethoven pieces split to the left with the d6g4 chord being the deciding factor.

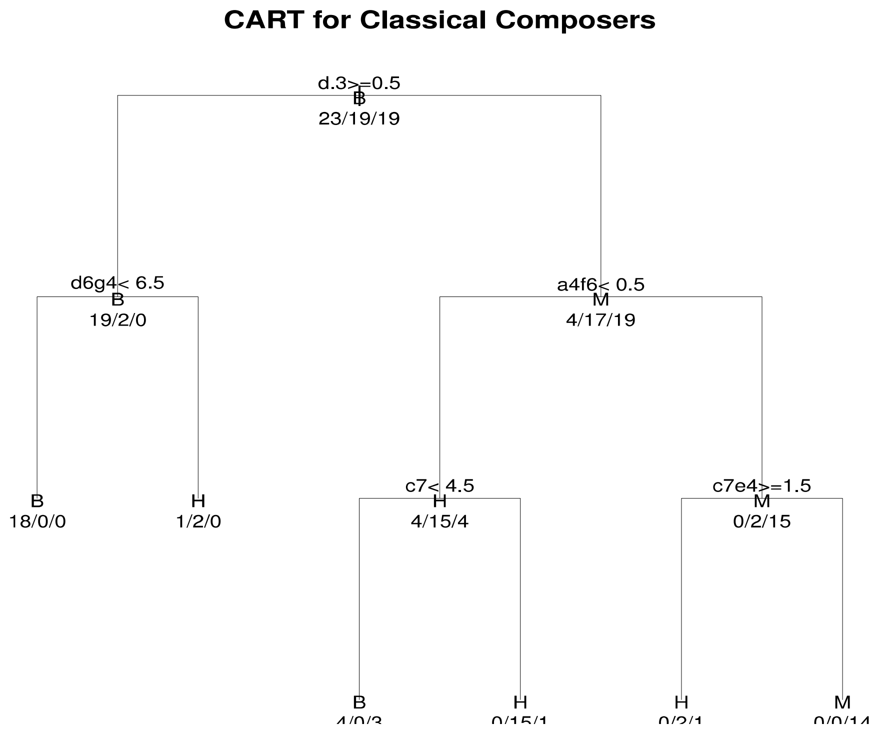

Figure 4: Rplot Dendrogram for CART composer authorship using only C-Major key pieces.

The final results are very similar between both data sets and show that the similarities between a-minor and C-major are prevalent enough to keep the results from skewing more towards one composer to another.