Composer Style Attribution

Kevin Laube
kevlaube@stanford.edu

Naroa Zurutuza
naroas@stanford.edu

CS 229 Final Project
December 13, 2013

Abstract

Our project was done in conjunction with the Stanford Music Department's Josquin Research Project. Josquin was one of the most famous composers from the early Renaissance era, and his works were so popular that others put his name on anonymous pieces, most likely in an attempt to boost sales. As a result, over 370 works have been attributed to him, although the true author of many of the pieces is disputed. Recent challenges have been made to the true authorship of the pieces, mostly on grounds of stylistic features or manuscript format.

Our goal was to apply artificial intelligence and machine learning techniques to quantify Josquin's style and make predictions about the true author of the disputed set of scores.

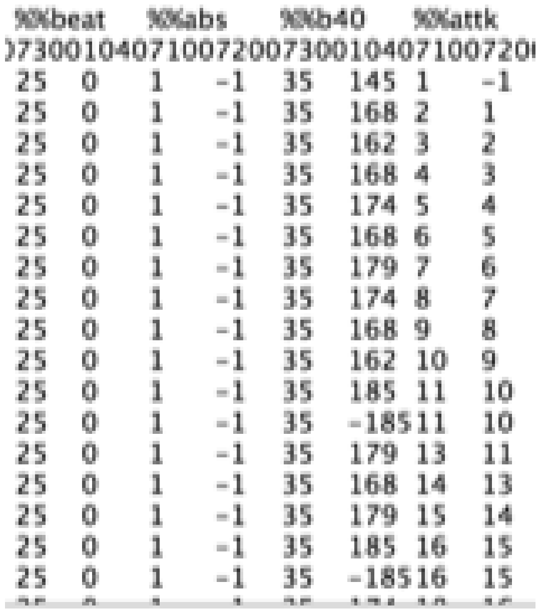
Task Definition

Music classification can be a very challenging task, and is one most humans struggle to do with high accuracy. However, this is in large part to a general inability of humans to track a la rge number of subtle, infrequent patterns that are stylistic marks of most composers. Ideally, a computer could run an unsupervised learning algorithm to extract any recurring feature from a composer's pieces, but this style of pattern recognition is currently beyond anything we have currently been exposed to. This limited us to defining a set of features that we thought would be relevant across all the pieces in our training dataset, which spanned 4 different composers.

Our first goal was to decide on features to use for a model. We suspected this would be the most challenging aspect of the project, especially given the fact trained musicians have not been able to decide on the real author of the Josquin pieces. While many trained ears are able to distinguish between two pieces, the problem of choosing a consistent set of features that can distinguish between scores of different composers seems much harder. Using too few features would lead to an inability to express the music in a manner that would allow machine learning methods to distinguish between scores, while too many features could lead to overfitting and similar issues distinguishing between composers. Ideally, the music would be converted to set of features that causes scores by different composers to cluster. Another issue is the feasibility of representing music in the manner described. While it seems intuitively possible to do it, and others have had reasonable degrees of success attempting similar projects, to resolve the dispute we would need features resulting in a classification accuracy in the high 90 percentile range.

Data Description

Music can be represented as an audio file or in a numerical format, and the types of analysis that can be done differ based on which type of file is used. We used a notearray program formatting, which takes **kern files and extracts useful information into a 2 dimensional matrix format. This results in a matrix with a few columns for each voice, one containing note information and the others containing auxiliary information. Each new row indicates a movement by one voice in the score, which results in a variable number of rows between songs. An example for one voice is shown below.



```
 %%beat    %%abs     %%b40     %%attk
073001040710072007300104071007200
 25    0     1    -1    35    145  1     -1
 25    0     1    -1    35    168  2      1
 25    0     1    -1    35    162  3      2
 25    0     1    -1    35    168  4      3
 25    0     1    -1    35    174  5      4
 25    0     1    -1    35    168  6      5
 25    0     1    -1    35    179  7      6
 25    0     1    -1    35    174  8      7
 25    0     1    -1    35    168  9      8
 25    0     1    -1    35    162  10     9
 25    0     1    -1    35    185  11     10
 25    0     1    -1    35   -185 11     10
 25    0     1    -1    35    179  13     11
 25    0     1    -1    35    168  14     13
 25    0     1    -1    35    179  15     14
 25    0     1    -1    35    185  16     15
 25    0     1    -1    35   -185 16     15
```

In the example above, the column with numbers above 100 is the most significant. This shows the exact pitch values for each note being played, where negative values indicate the note was sustained to the next timestep. The notearray format we chose uses the base 40 numbering sytem, which is an efficent method for representing diatonic pitches.

The dataset, which is available online at jrp.ccarh.org, contains 736 pieces written during Josquin's period, divided into 2 main categories.

- 447 works by known composers, including:
  - 131 scores by Josquin
  - 93 scores by Johannes Ockeghem
  - 40 pieces by Marbrianus de Orto
  - 183 pieces by Pierre de la Rue
- 289 scores attributed to Josquin but with disputed authorship

Feature Selection

Our features were chosen by a combination of researching what stylistic elements vary the most between composers, and examining what seemed reasonable to extract given the format we had access to.

Number of notes in piece

This feature was simply the total number of notes in the song.

Sequences of Notes

- This feature consisted of all sequences of notes of a selected length in a song. We tried several different sequence extraction methods, including
  - All sequences of length K, for $K = 2 \vee 3 \vee ... \vee 7$ (i.e., K takes on one value). Our optimal result, measured by maximizing classification acccuracy, from this selection was for K = 4.
  - All sequences of length K, for K in a range of values from 2 to 7. However, this resulted in lower classification accuracy.

In the end, we decided to use all sequences of length 4, which empirically gave us the highest classification accuracy. We also excluded any sequences containing a rest.

Intervals between notes

This feature was calculated by examing the difference between pitches of consecutive notes. We considered both individual intervals and sequences of intervals, but found empirically that we acheived higher classification accuracy when considering individual intervals. Thus, for each score we created a list of the frequency of all the intervals it contained.

Note Length

For this feature we calculated the mean note duration in each score, as well as the variance.

Pitch Frequency

The base-40 numbering system takes on 40 values, which are multiplied by their octave to get the final value. Thus, to get the correct pitch we calculated the pitch value as pitch = note % 40. Then we added a feature for each pitch, and normalized by dividing the number of occurences of each pitch in a song by the number of notes in the song.

Octave Frequency

To calculate the octave of each note, we calculated floor(octave/40). Then we added a feature for all possible octaves and normalized by dividing the total number of occurrences of notes in each octave by the total number of notes in the song.

Average Movement in Piece

For this feature, we looked at whether the pitch increased or decreaesd between all sequential pairs of notes, and then added features for mean and variance of the movement of the notes for each song.

Final Feature Comments

Of the features mentioned above, sequences of notes and intervals between notes, created variable length feature vectors, while the rest created a constant length feature vector. This was fine for a One vs All classifier, which associates a weight with each feature after training, but was not ideal for a support vector machine or neural network, which require fixed input lengths. The easiest solution to this was to trim the weight vectors learned by the One Vs All classifier and take the most heavily weighted features, the intuition being that these features were the most indicative of the score being by a specific composer. Each of these features could then be added to the feature vectors used by the SVM and neural network.

Models

We tried a few different models, and found a high variation in classification accuracy between them. Initially, we tried multi-class classification schemes for our models, in an attempt to accurately predict from 4 composers. The intuition here was to maximize prediction accuracy with these models and find a set of features so that the data would cluster in the higher-dimensional feature vectors, which would only increase classification accuracy when we reduced the classification problem to a binary one (Josquin or non-Josquin). We abandoned these models once we had decided on our features, and moved on to binary classification to see if we could apply any other optimizations towards our final goal of classifying Josquin pieces accurately.

One Vs All Classifier

First we implemented a One vs All Classifier, which had a weight vector for all features from Josquin scores and one for all features from non-Josquin scores. For this model, we used sequences of notes and intervals between notes as our features. This classifier ended up being fairly accurate, which confirmed the belief that recurring sequences of notes are a major stylistic mark of a composer.

Naive Bayes

Going off the fact that sequences and intervals seemed to be fairly accurate features, we decided to try Naive Bayes. This was very straightforward to implement, and we used sequences and intervals as our only features. Naive Bayes ended up being slightly more accurate then the One vs All classifier, and was overall our most accurate.

K means clustering

After Naive Bayes, we decided we had reached a limitation on the accuracy that could be achieved through our current features, and expanded our feature set to use the rest of the features described above. This required trimming the weight vectors learned from the one vs all classifier to include the most naive bayes. Once we had the feature vectors in place, we ran a k means clustering algorithm to see if the data was clustering. We initialized one cluster for each composer, and ran the classifier. Unfortunately, we barely beat random chance, achieving a testing accuracy of 38%.

Support Vector Machine

Our next choice in a model was a support vector machine, and we used an implementation provided by the sci-py library. We had high hopes for the SVM, with the intuition that if we chose the right features we could represent the scores in a manner that would allow them to cluster in the feature space and be linearly separable. While the SVM acheived 98% accuracy on the training set, we were unable to get it to perform well on the testing, despite tweaking the regularization parameter and exploring different kernels a fair amount. The high training accuracy does suggest promise for the SVM, however, so given more time we would have liked to experiment with different kernel functions and different training parameters.

Neural Network

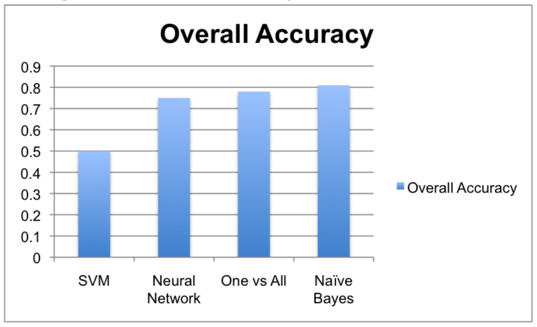
Our final attempt was a neural network. We used a feed forward network with back propagation for training. We used an implementation provided by pybrain. The neural network proved very inconsistent, acheiving anywhere from 60 to 90 percent accuracy on a separate training set. The variation in accuracy most likely came from the random elements included in the testing set, which we discuss further below.

Results

Testing Methodology

We performed the majority of our training using the full set of scores with known composers, and removed 10-20% of the scores each time to use as a separate testing set. We also made a testing set containing 130 songs by Josquin and 130 by the other three composers, the intuition behind this being a larger number of training samples from non-Josquin composers would skew the prediction results away from Josquin. Additionally, when selecting the testing set we split it evenly between Josquin and non-Josquin scores. Given the small number of available scores, we experimented with 10-fold cross validation to improve our results. Unfortunately, this didn't affect our results significantly. Our classification accuracy is reported below. For the neural network, we reported the 10-fold cross validation results since there was a significant variation between consecutive attempts.

Testing Set Accuracy



As we see above, we were unable to do much better then 80% accuracy. However, this significantly outperforms randomly guessing, and suggests there was a reasonable degree of significance in the features we chose.

Error Analysis

The majority of our errors clearly resulted from not choosing meaningful features. While we were unable to decide on features, one addition we would have liked to make would be more research into significant sequence and interval patterns in Josquin's work. Our approach was probably aimed a bit too generally at choosing a more universal set of features that would distinguish between any of the composers, but we probably could have honed in on more specific marker's of Josquin's style.

In terms of models used, we also think we could have acheived better accuracy with the neural network and SVM. For the SVM we were worried about overfitting and changed to a linear kernel in hopes of achieving better results, and reduced our regularization constant when we were still unable to attain reasonable accuracy. Given the black box nature on SVMs and neural networks, we were unable to tune the parameters further to fit the data better.

Neither one of us has much of a musical background, so our approach may have missed several higher level features of music that we could have looked at. In particular, counterpoint, which describes the movement of different voices in conjunction, and harmony between voices were two features that are very apparent in the music when listening to it, but proved too difficult for us to successfully quantify and extract. Given the level of difficulty of the problem of choosing accurate features, given more time we would have spent all of it researching better features.

Conclusions

While we were unable to make predictions on the testing set with enough accuracy to begin making predictions on the set of controversial songs, we were still able to glean a few useful insights into the problems of selecting features and models.

In terms of features, we found sequences of notes to be one of the best predictors of which composer a piece is written by. This makes sense, given many composers are known to repeat similar patterns or sequences in a large portion of their scores. After sequences, we found intervals and sequences of intervals between notes were also very indicative of style. Pitch frequency was also a significant indicator, and after that our features dropped off in relevance.

References

- We would like to thank the Stanford Music Department for making all the song files we used available. Specifically, we would like to thank Jesse Rodin and Craig Sapp for providing both musical and technical advice.

- NoteArray manual page
  extras.humdrum.org/man/notearray/

- *Music Style Attribution*, Kranenburg and Backer, 2004

- SciPy for an implementation of an SVM and K means clustering algorithm

- Pybrain for providing a Neural Network implementation