A Machine Learning Approach to Musical Style Recognition

Roger B. Dannenberg, Belinda Thom, and David Watson
School of Computer Science, Carnegie Mellon University
{rbd,bthom,dwatson}@cs.cmu.edu

Abstract

Much of the work on perception and understanding of music by computers has focused on low-level perceptual features such as pitch and tempo. Our work demonstrates that machine learning can be used to build effective style classifiers for interactive performance systems. We also present an analysis explaining why these techniques work so well when hand-coded approaches have consistently failed. We also describe a reliable real-time performance style classifier.

1 Introduction

The perception and understanding of music by computers offers a challenging set of problems. Much of the work to date has focused on low-level perceptual features such as pitch and tempo, yet many computer music applications would benefit from higher-level understanding. For example, interactive performance systems [3] are sometimes designed to react to higher-level intentions of the performer. Unfortunately, there is often a discrepancy between the ideal realization of an interactive system, in which the musician and machine carry on a high-level musical discourse, and the realization, in which the musician does little more than trigger stored sound events. This discrepancy is caused in part by the difficulty of recognizing high-level characteristics or style of a performance with any reliability.

Our experience has suggested that even relatively simple stylistic features, such as "playing energetically," "playing lyrically," or "playing with syncopation," are difficult to detect reliably. Although it may appear obvious how one might detect these styles, good musical performance is always filled with contrast. For example, energetic performances contain silence, slow lyrical passages may have rapid runs of grace notes, and syncopated passages may have a variety of confusing patterns. In general, higher-level musical intent appears chaotic and unstructured when presented in low-level terms such as MIDI performance data.

Avoiding higher-level inference is common in other composition and research efforts. The research literature is filled with articles on pitch detection, score following, and event processing. There are also interactive systems that respond to simple features such as duration, pitch, density, and intervals, but there is relatively little discussion of higher-level music processing.

In short, there are many reasons to believe that style recognition is difficult. Machine learning has been shown to improve the performance of many perception and classification systems (including speech recognizers and vision systems). We have studied the feasibility of applying machine learning techniques to build musical style classifiers.

The result of this research is the primary focus of this paper. Our initial problem was to classify an improvisation as one of four styles: "lyrical," "frantic," "syncopated," or "pointillistic" (the latter consisting of short, well-separated sound events). We later added the additional styles: "blues," "quote" (play a familiar tune), "high," and "low." The exact meaning of these terms is not important. What really matters is the ability of the performer to consistently produce intentional and different styles of playing at will.

The ultimate test is the following: Suppose, as an improviser, you want to communicate with a machine through improvisation. You can communicate four different tokens of information: "lyrical," "frantic," "syncopated," and "pointillistic." The question is, if you play a style that you identify as "frantic," what is the probability that the machine will perceive the same token? It is crucial that this classification be responsive in real time. We arbitrarily constrained the classifier to operate within five seconds.

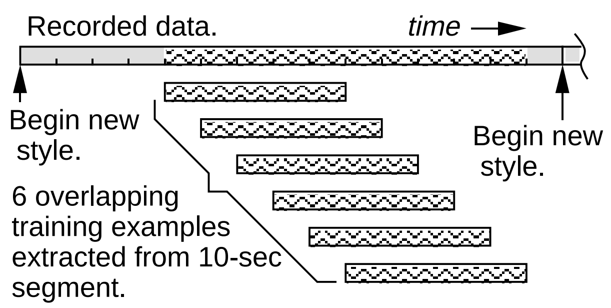
Recorded data.
time
Begin new style.
Begin new style.
6 overlapping training examples extracted from 10-sec segment.

Figure 1: 6 overlapping training examples are derived from each 15 seconds of performance.

2 Data Collection

To study this problem, we created a set of training data recorded from actual performances. So far, our experiments have used trumpet performances recorded as MIDI via an IVL Pitchrider 4000 pitch-to-MIDI converter for machine-readable data, and DAT tape for audio data. We also devised software to help collect data. The performer watches a computer screen for instructions. Every fifteen seconds, a new style is displayed, and the performer performs in that style until the next style is displayed. Because the style changes are abrupt, we throw out the first four seconds of data, allowing for a "mental gear shift" and a new style to settle in. For each style we retain 10 seconds of good data. (The 15th second of data is also discarded.)

We want our classifier to exhibit a latency of 5 seconds, so it must only use 5 seconds of data. Therefore, the MIDI data is divided into six overlapping intervals with durations of five seconds, as shown in Figure 1. Since we are using supervised training, we need to label each interval of training data. Rather than use the labels that were displayed for the performer during the recording, we decided to have the performer rate the data afterward to make sure each sample realizes the intended style. These ratings also give a richer description of the data; for example, a performance might be both "frantic" and somewhat "syncopated," and this additional information can be used in some machine learning approaches.

We recorded 25 examples each of 8 styles, resulting in 1200 five-second training examples. Ideally, we would rate each five-second example on each style, but this would require 9600 separate ratings. To reduce the number of ratings, we rated ten-second segments and assigned the rating to each of the six derived training examples. Thus, each individual rating is shared by six training examples. The training data was presented for rating in random order.

3 Classification Techniques

We constructed several classifiers using naive Bayesian, linear, and neural network approaches. We will describe the basic ideas here. To build a classifier, we first identified 13 low-level features based on the MIDI data: averages and standard deviations of MIDI key number, duration, duty factor, pitch and volume, as well as counts of notes, pitch bend messages, and volume change messages. (Pitch differs from key number in that pitch-bend information is included. Duty factor means the ratio of duration to inter-onset interval.)

3.1 Bayesian Classifier

The naive Bayesian classifier [1] assumes that the features are uncorrelated and normally distributed. (Neither of these is true, but this approach works well anyway.) Given a vector of features, F, we would like to know which classification C is most likely. Using our assumptions and Bayes' Theorem, it can be shown that the most likely class is the one whose mean feature vector has the least "normalized distance" to F. The "normalized distance" is the Euclidean distance after scaling each dimension by its standard deviation:

$$\Delta_C = \sqrt{\sum_{i=1}^{n} \left(\frac{F_i - \mu_{C,i}}{\sigma_{C,i}} \right)^2},$$

where C indexes classes, i indexes features, $\mu_{C,i}$ are means, and $\sigma_{C,i}$ are standard deviations.

3.2 Linear Classifier

Linear classifiers compute a weighted sum of features, where a different set of weights is used for each class. The linear classifier algorithm we used [4] also assumes features are normally distributed, but not that they are uncorrelated. A linear classifier tries to separate members of the class from non-members by cutting the feature vector space with a hyperplane. Depending on the features, this division may or may not be very successful.

3.3 Neural Networks

Of the three approaches we tried, neural networks are the most powerful because they incorporate nonlinear terms and they do not make strong assumptions about the feature probability distributions. [1] We used a Cascade-Correlation architecture [2] which consists initially of only input and output units

Number of Classes	Bayesian	Linear	Neural Network
4	98.1	99.4	98.5
8	90.0	84.3	77.0

Table 1: Percentage of correct classifications by different classifiers.

(equivalent to a linear classifier). During training, hidden units are added with connections to inputs, outputs, and any previously added hidden units. Effectively, there are many hidden layers, each with one node. Every hidden unit applies a non-linear sigmoid function to a weighted sum of its inputs. There is one output per class; outputs are regarded as boolean values that predict membership in a given class.

4 Results

All three classifiers produced excellent results with 4 classes and impressive results with 8 classes. We measured performance by training a classifier on 4/5 of the data and then classifying the remaining 1/5. (Since the data is in groups of 6 overlapping—and therefore correlated—training examples, entire groups went into either the training or the validation sets.) This was repeated 5 times with different partitions of the data so that each example was classified once. The classifier outputs Yes or No for each of 4 or 8 classes, and we report the total percent of correct answers. We were surprised by the reliability of the classifiers. Table 1 shows the numerical results. Training time for the Bayesian and Linear classifiers occurred in seconds, but the neural net took hours.

We implemented a real-time version of the naive Bayesian classifier. A circular buffer stores the last five seconds of MIDI data, and every second, features are computed and a classification is made. A fifth classification, "silence," was added for the case where the number of notes is zero. The execution time is well under 1 ms per classification, so computation cost is negligible.

Note that many of the misclassifications in the 8-class case involve the "quote" style, which one would expect to require knowledge of familiar melodies and some encoding of melody into the features.

5 Discussion

Our work has demonstrated that machine learning techniques can be used to build effective, efficient, and reliable style classifiers for interactive performance systems. However, we wondered why these systems worked well when hand-coded approaches have failed. We also wondered under what circumstances these machine learning systems would also fail.

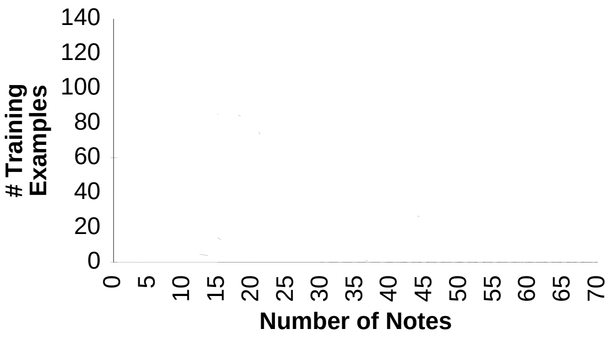

Figure 2: Histogram of the number of notes in each sample for a given style.

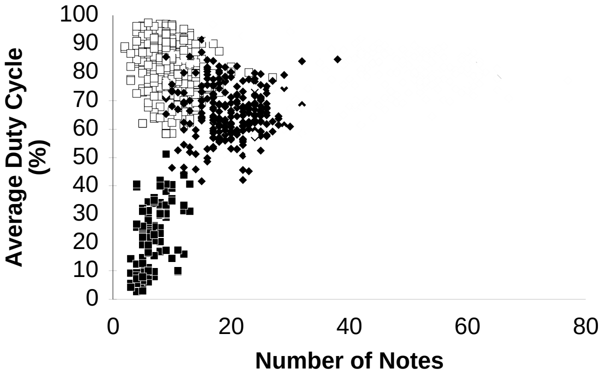

Figure 3: Average duty cycle versus number of notes (scatter plot).

To answer the first question, we have attempted to visualize the data by constructing histograms and scatter plots. Since there are 13 features, one can only look at projections onto one or two dimensions. Figure 2 shows a histogram of the number of notes feature. The number of notes per second is a common input parameter in performance systems because it has a high correlation with the concept of "frantic" or "fast." However, as Figure 2 shows, there is considerable overlap among the histograms even though the class ratings are almost all mutually exclusive. Other histograms show similar overlap, so classifications based on a single feature are not very useful.

Figure 3 shows a scatter plot of two dimensions: average duty cycle and number of notes. Here, the four classes almost separate. Given Figure 3, can we still argue that machine learning is necessary? To build a good classifier without machine learning, one would first need to identify good features. Individ-

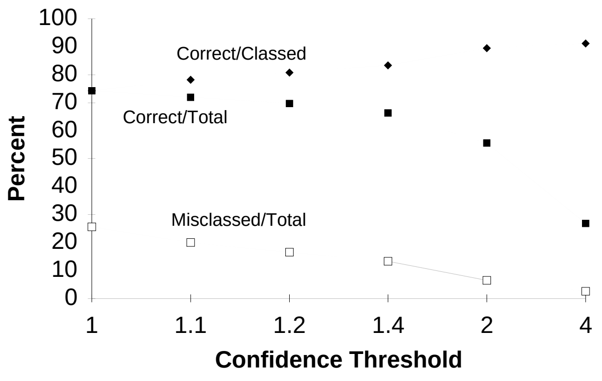

Figure 4: As the confidence threshold increases, the total number of classified examples decreases, but the number of misclassified examples decreases faster, so the ratio of correctly classified to all classified examples increases.

ual features do not work well, and in our study there are 132 different pairs of features one might consider. Most pairs of features will not lead to good classifiers, and even if a good pair is known, parameters must be chosen accurately. Casually guessing at good features or combinations, plugging in "reasonable" parameters and testing will almost always fail. In contrast, machine learning approaches that automatically take into account a large body of many-dimensional training data lead to very effective classifiers.

It is worth noting that we never hand-tuned the original 13 features; rather, we allowed the inference algorithms to determine the effective use of the data. The classification of stylistic intent is relatively simple for machine learning algorithms (at least in the 4-class case), which allows one to spend less time on problems of representation.

5.1 Live Performance

Experience with the 4-class classifier in live performance has provided a new subjective evaluation. The classifier is fast and effective, but not as reliable as the data would predict. Experiments have shown that the performer's "style," or feature vector distribution, changes when the performer interacts with computer-generated music. Thus, training examples should be captured in context. In other words, a "syncopated" style in isolation differs from a "syncopated" style performed in an interactive composition.

Another problem with live performance has been the "false positives" (misclassifications which erroneously imply the performer is playing a particular style). It may be better to report nothing than to make a wrong guess.

An experiment with our naive Bayesian classifier suggest that simple confidence measures can dramatically reduce false positives. Recall that this classifier makes decisions based on normalized distances from means. If the distance to the means of two classes are nearly equal, our confidence in the decision should be reduced. Therefore, we simply reject classifications when the least distance is not less than a given fraction of the next-to-least distance. This type of analysis is one of the beauties of a statistical approach. Figure 4 illustrates the reduction of false positives using this technique.

6 Summary and Conclusion

Machine learning techniques are a powerful approach to music analysis. We constructed a database of training examples and used it to train systems to classify improvisational style. The features used for classification are simple parameters that can be extracted from MIDI data in a straightforward way. The styles that are classified consist of a range of performance intentions: "frantic," "lyrical," "pointillistic," "syncopated," "high," "low," "quote," and "blues." The first four of these styles are classified with better than 98% accuracy. Eight classifiers, trained to return "yes" or "no" for each of the eight styles had an overall accuracy of 77% to 90%. Confidence measures can be introduced to reduce the number of false positives.

Further work is required to study other classification problems, feature selection, and feature learning. We believe this work has applications in music performance, composition, analysis, and education to mention just a few. We expect much more sophisticated music understanding systems in the future.

References

[1] Bishop, C. M. *Neural Networks for Pattern Recognition.* Clarendon Press, 1995.

[2] Fahlman, S. E., and Lebiere, C. The Cascade-Correlation learning architecture. Tech. Rep. CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Feb. 1990.

[3] Rowe, R. *Interactive Music Systems.* MIT Press, 1993.

[4] Rubine, D. The automatic recognition of gestures. Tech. rep., School of Computer Science, Carnegie Mellon University, 1991.