NONNEGATIVE MATRIX FACTORIZATION FOR REAL TIME MUSICAL ANALYSIS AND SIGHT-READING EVALUATION

Chih-Chieh Cheng, Diane J. Hu, and Lawrence K. Saul

Department of Computer Science and Engineering
University of California, San Diego
{chc028,dhu,saul}@cs.ucsd.edu

ABSTRACT

Sight-reading is the ability to read and perform music from a written score with little or no preparation. Though an integral part of musicianship, it is rarely or minimally addressed in traditional music lessons. In this paper, we describe a real-time system for sight-reading evaluation of solo instrumental music. The system is trained to recognize monophonic and polyphonic music from acoustic instruments without digital pickups. The pattern-matching in the back-end is achieved by nonnegative matrix factorization, an algorithm that represents notes as combinations of learned templates and chords as combinations of single notes. As part of the user interface, an animated musical score provides beginning musicians with instant visual feedback as they practice to improve their sight-reading.

Index Terms— Non-negative Matrix Factorization, Polyphonic music processing, Machine learning, Sight-Reading evaluation

1. INTRODUCTION

One of the most important parts of musicianship is the ability to sight-read or play unfamiliar music from a written score. A proficient sight-reader can more easily take part in ensembles, accompany other musicians, and learn new pieces for long-term study. Contrary to popular belief, however, sight-reading is a learned skill that is largely independent of a musician's physical technique or level of ear-training. To become a proficient sight-reader requires the constant practice of sight-reading [1]. Unfortunately, the beginning musician often struggles without the help of a teacher who can provide immediate feedback. Such feedback is not only necessary to identify correctly versus incorrectly played notes, but also to sustain interest when progress is often slow and unquantifiable.

To address these challenges, we have designed a prototype system that helps beginning musicians practice their sight-reading in the absence of human teachers. The intended operation of this system is shown in Fig. 1. The front-end graphical interface is designed specifically to facilitate sight-reading practice in a way that agrees with commonly accepted pedagogical methods. The machine displays an animated score, "listens" to the player's instrument, and provides instant visual feedback distinguishing correctly versus incorrectly played notes. While many existing systems have related goals, our application distinguishes itself by (1) requiring no digital pickups or electronic instruments, (2) targeting the needs of beginning musicians, and (3) providing immediate, visual feedback.

The back-end of this system must operate in real-time to determine which notes have been correctly played by the user. One approach to do this is by analyzing the spectrum of the notes. The spectral profile (or timbre) of a pitched musical instrument is characterized by the relative strengths of its harmonics in the frequency domain. For a single note, these harmonics occur at integer multiples of the fundamental frequency of vibration, f_0. The value of f_0 indicates the periodicity of the acoustic waveform generated by the instrument. For a percussion, wind, or string instrument, the value of f_0 also corresponds essentially to the perceived pitch of whatever note is being played.

The harmonics of a musical signal appear as peaks in the magnitude power spectrum computed by the fast Fourier transform (FFT). When a single note is played, the pattern of harmonics is clear and easily resolved, even by visual inspection. But when more than one note is played, the pattern of harmonics is more complicated, and the chord structure is not apparent from visual inspection. The analysis of monophonic and polyphonic musical signals presents an interesting challenge in acoustic pattern recognition [2, 3].

One useful model for musical analysis is to view musical chords as constructive combinations of their constituent notes. Specifically, in the frequency domain, the nonnegative superposition of magnitude power spectra for individual notes provides an excellent approximation to the magnitude power spectrum of those notes played in combination. Based on this observation, many researchers have suggested nonnegative matrix factorization (NMF) [4] as a strategy for learning useful represention of pitched notes in polyphonic music [5, 6, 7, 8, 9].

In this paper, we describe a similar framework for the back-end of a real-time system for sight-reading evaluation. We use NMF to learn nonnegative basis templates for each note and to evaluate whether the sound from the user's musical instrument matches the notes on a given musical score. NMF has two distinct advantages for the pitch-tracking problem in our application. First, it gracefully handles the problem of detecting multiple pitches. Second, it provides an efficient framework for estimating the harmonic templates of specific instruments. Such templates can potentially yield more accurate pitch-tracking than generic schemes that do not exploit this form of prior knowledge.

2. MUSICAL ANALYSIS BY NMF

2.1. Offline estimation of musical note templates

We begin by briefly reviewing NMF [4]. Given a large nonnegative matrix $\mathbf{Y}$, the goal of NMF is to derive a low-rank approximation $\mathbf{Y} \approx \hat{\mathbf{Y}}$ where $\hat{\mathbf{Y}} = \mathbf{W}\mathbf{X}$ factorizes as the product of two smaller nonnegative matrices $\mathbf{W}$ and $\mathbf{X}$. One appropriate cost function for NMF is the generalized Kullback-Leibler (KL) divergence:

$$G(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_{\alpha t} \Big[Y_{\alpha t} \log(Y_{\alpha t}/\hat{Y}_{\alpha t}) - Y_{\alpha t} + \hat{Y}_{\alpha t} \Big]. \qquad (1)$$

Fig. 1. Player demonstrates system setup on the piano.

This cost function vanishes if and only if $\mathbf{Y} = \hat{\mathbf{Y}}$. It can be optimized subject to the nonnegativity constraints on $\mathbf{W}$ and $\mathbf{X}$ by iterating the multiplicative updates:

$$W_{\alpha\beta} \quad \leftarrow \quad W_{\alpha\beta} \left[\sum\nolimits_t X_{\beta t} Y_{\alpha t} / \hat{Y}_{\alpha t} \right], \tag{2}$$

$$X_{\beta t} \quad \leftarrow \quad X_{\beta t} \left[\frac{\sum_\alpha W_{\alpha\beta} Y_{\alpha t} / \hat{Y}_{\alpha t}}{\sum_\gamma W_{\gamma\beta}} \right]. \tag{3}$$

In practice, the updates are applied in an alternating fashion, holding $\mathbf{X}$ fixed while re-estimating the elements of $\mathbf{W}$, then holding $\mathbf{W}$ fixed while re-estimating the elements of $\mathbf{X}$. Alternated in this way, the updates converge monotonically to a local minimum of the cost function in eq. (1).

NMF is very well suited to learning representations of musical notes based on their magnitude power spectra in the frequency domain. In our application, the matrices $\mathbf{Y}$, $\mathbf{W}$, and $\mathbf{X}$ have the following interpretation: (i) the columns of $\mathbf{Y}$ store the (nonnegative) power spectra from short analysis windows of single note recordings; (ii) the columns of $\mathbf{W}$ store basis templates for the magnitude power spectra of these notes; (iii) the columns of $\mathbf{X}$ store the coefficients used to reconstruct the actual observations in $\mathbf{Y}$ from the learned templates. The basis templates in $\mathbf{W}$ and the reconstruction coefficients in $\mathbf{X}$ are constrained to be nonnegative.

For our prototype system, we used NMF to learn basis templates in $\mathbf{W}$ for sixty notes spanning five octaves on the piano. The single note recordings were obtained from the University of Iowa Musical Instrument Samples data set. This data set contains samples for each note at three different levels: soft, medium loud, and loud. We used mono clips from all three levels (sampled at 44100 Hz) as training data, after removing beginning and ending silences from each waveform. FFTs were performed in 25 ms shifts on half-overlapping analysis windows of 50 ms duration. From the magnitude spectra of these FFTs, we trained 5 basis templates for each note. Multiple templates per note were used to model variations in timbre due to onset/offset effects and varying dynamics.

2.2. Note and chord verification

We use the estimated basis templates to evaluate whether the user is playing the desired note(s) on the musical score. Note that this problem is simpler than automatic transcription or score-following (not to mention source separation), since our only task is to verify whether correct notes are being played at the correct tempo. In particular, our application does not require the system to identify wrong notes; it only needs to detect them. Also, our application does not need to track variations in tempo; indeed, one important goal of sight-reading is to play strictly in time. Our framework does not attempt

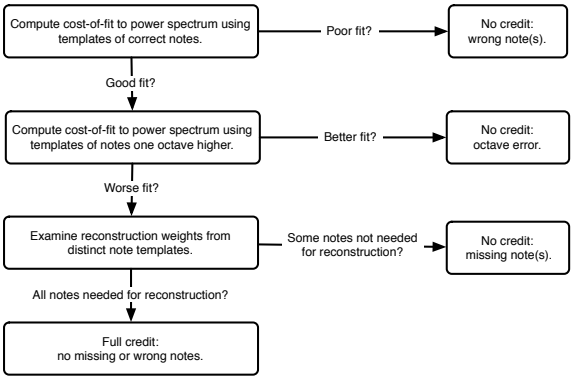

Fig. 2. Flow chart of decision-making process for sightreading evaluation; see text for details.

to solve the general problems of automatic transcription or score-following, which would require more detailed modeling and extensive pattern-matching. Instead, the framework focuses on providing feedback in real-time, which restricts the number of available cycles we have for computation.

To analyze short-time power spectra of the user's playing, we attempt to reconstruct them from the basis templates estimated by NMF. Roughly speaking, if the templates for the desired notes are necessary and sufficient to reconstruct the observed power spectra to a high degree of accuracy, then the user is credited for having played the correct notes. A high level flow chart of the decision-making process is shown in Fig. 2. The remainder of this section provides more details.

The main criterion we use for note and chord matching is a normalized form of the reconstruction error from NMF. In particular, let $\mathbf{y}_t$ denote the power spectrum from one analysis window of the user's playing centered on time t. Also, let $\mathbf{W}_t$ collect the columns of $\mathbf{W}$ whose templates represent notes also indicated by the musical score at this time. The optimal reconstruction $\hat{\mathbf{y}}_t = \mathbf{W}_t \mathbf{x}_t$ is obtained by minimizing the generalized KL divergence in eq. (1) between $\mathbf{y}_t$ and $\hat{\mathbf{y}}_t$ over the nonnegative coefficients of $\mathbf{x}_t$. Finally, to obtain an overall cost-of-fit at time t, we normalize the reconstruction error by the total power in the window:

$$\varepsilon_t = \frac{1}{\|\mathbf{y}_t\|} \sum_\alpha \left[y_{\alpha t} \log(y_{\alpha t} / \hat{y}_{\alpha t}) - y_{\alpha t} + \hat{y}_{\alpha t} \right]. \tag{4}$$

The normalization ensures that the reconstruction error is evaluated in proportion to the overall signal level.

For fixed basis templates $\mathbf{W}_t$, minimizing the reconstruction error in eq. (4) is a convex optimization, and the iterative update in eq. (3) converges quickly to the global solution. Typically, less than 20 iterations are required for satisfactory convergence. In practice, this is sufficient for implementation in our real-time system which verifies the user's playing at a frame rate of 40 frames per second.

To verify that the user is playing the desired note(s) at a given time t, we use a multi-step process. The first step attempts to reconstruct the observed power spectrum $\mathbf{y}_t$ with templates from these notes. At the end of this first step, we label the note(s) as misplayed if the cost-of-fit in eq. (4) from the reconstruction exceeds a fixed threshold. In this case, the process terminates. We have found that with a carefully chosen threshold, this step yields very few false negatives (i.e., rejecting correct notes as incorrect ones).

Further steps are required if the cost-of-fit in the first step is below threshold, indicating a good fit. These further steps are required

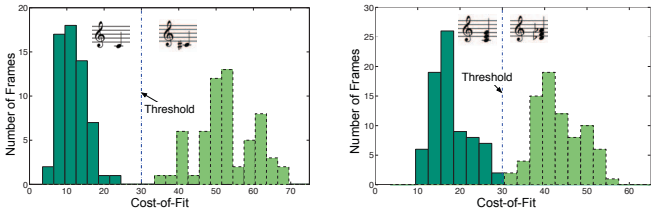

Fig. 3. *Left*: histograms of costs-of-fit for short-time magnitude spectra on the piano. The left histogram shows the costs-of-fit when a middle C is reconstructed by the (matched) templates for middle C; the right histogram shows the costs-of-fit when a C# (one half-step higher) is reconstructed from the (mismatched) templates for middle C. The histograms are well-separated; thus, by choosing an appropriate threshold, we can detect notes off by a half-step and reject them as errors. *Right*: analogous histograms when the chords CEG and D♭FA♭ are considered as potential matches to the desired notes CEG. This simple thresholding scheme works to detect errors from other intervals, with the exception of single octave errors.

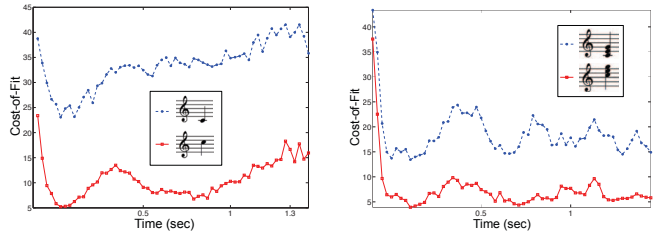

Fig. 4. *Left:* Costs-of-fit when a note one octave above middle C is matched to templates for the same note (bottom trace in red) and one octave below (top trace in blue). *Right*: Costs-of-fit when a triad one octave above middle C is matched to templates for the same notes (bottom trace in red) and one octave below (top trace in blue). The octave error is revealed by computing both costs-of-fit and comparing them directly.

to verify the identity of the played note and to rule out false positives in which incorrect notes are mistaken for correct ones. In practice, false positives from the first step are rare with the exception of single octave errors. This confusion occurs when the played notes are exactly one octave above the notes in the score. The second step of our verification process is designed to remove these false positives. This is done by also reconstructing the observed power spectrum $\mathbf{y}_t$ from the templates of notes one octave higher than those indicated in the score. If this latter fit is more accurate than the original one, then the user's playing is again labeled as incorrect.

Finally, one additional step is required to verify that the user's playing is correct in the case that the musical score indicates two or more notes (e.g., a chord). When a chord is indicated, not only do we verify that the templates from the correct notes yield a superior fit, but we also check that templates from all the notes of the chord are needed in the reconstruction. This is done by checking the reconstruction coefficients in $\mathbf{x}_t$. If this final check is satisfied, the chord is recognized as being played correctly. Otherwise, it is assumed that one or more notes from the chord are missing. See Fig. 2 for a schematic diagram of the overall note verification process.

Figs. 3 and 4 show some experimental results that were used in developing a prototype system. These experiments helped to determine appropriate thresholds for the branch points in Fig. 2. The samples in these tests were recorded by placing a laptop with a built-in microphone on top of the console piano in Fig. 1.

3. REAL-TIME SIGHT READING SYSTEM

We have implemented the back-end described in the previous section as part of an interactive system for sight-reading practice. Our system has been designed with an awareness that practicing to sight-read is very different than practicing for performance [1]. For effective sight-reading practice, the following three guidelines are widely accepted: (1) Proceed at a sufficiently slow tempo to play most notes correctly. (2) Never go backward to correct mistakes. (3) Pay attention to rhythm, either by counting out loud or using a metronome. We have attempted to incorporate each of these guidelines into the design of our interface, as described below.

To begin the application, the user chooses a sight-reading exercise at the appropriate level. Once selected, the musical score is displayed, and the user must play from the score at the specified tempo. Points are given for each note that is played correctly. However, if too many incorrect notes are played, the exercise restarts at a slower tempo. This restart mechanism forces a consistent tempo that is appropriate to the user's sight-reading level. In accordance with the tempo, the musical score scrolls to the left, and previously played notes disappear off the screen. This scrolling suppresses the instinctual tendency to back-track and correct previous mistakes. Further, to improve rhythmic literacy, "progress bars" are drawn above each note. The length of each progress bar is proportional to the duration of each note and "fills up" (with color) at the rate that the note's duration is passing.

At the conclusion of the piece, the player recieves a performance score based on the difficulty of the selected piece, the tempo at which the piece was played, and the number of notes played correctly. As a quantitative metric for self-evaluation, the performance score provides a well-defined target for further improvement, as well as an incentive for continued playing.

We have implemented a prototype of this system with basic functionality on the piano. All software was implemented in Objective-C/C on the Cocoa platform and tested on a 2.16 GHz Intel Core Duo MacBook Pro.

Fig. 5. Screenshot of the application in progress. The pink note indicates the current position of the musical score, while the blue notes indicate notes correctly played.

4. EXPERIMENTAL RESULTS

We evaluated the system by looking at its performance in off-line and on-line settings.

4.1. Offline Experiments

First, in an offline setting, we evaluated how well the algorithm distinguished correctly versus incorrectly played notes. In order to simulate the real-time environment, we recorded the data for these experiments by placing a laptop with a built-in microphone on top of the console piano in Fig. 1. Each test sample was 1.5 sec in duration, generating 60 frames to be analyzed by NMF. The computation time was approximately 2 ms per frame, thus meeting the need for a real-time application.

Table 1 shows the results from these experiments on single notes. For each experiment, the results show the number of frames, out of 60, labeled correctly by the implementation described in Section 2. The second, third, and fourth rows simulate common errors made during sight-reading practice, in which the played note (indicated by the row) differs from the correct note (indicated by the column) by a half-step or a full octave. Overall, the vast majority of frames with half-step and octave errors are correctly labeled as errors.

Table 2 shows similar results for experiments on chords. We performed tests on five major triad chords to simulate correct playing as well as common errors. The second and third rows show the results for half-step and octave errors, as in the single note experiments. The fourth and fifth rows show the results when a chord is played with missing or wrong notes. In these cases, the system should detect that the partially correct chord is not completely correct.

Overall, in Tables 1 and 2, the frame-by-frame results are quite accurate, especially considering that in the actual system (as described in section 4.2), the results of individual frames are aggregated over the duration of an entire note or chord.

Table 1. Experimental results for single notes.

Test Cases	**C**	**D**	**E**	**F**	**G**
Correct Note	60	60	59	60	60
Half-step up (sharp)	2	0	0	1	0
Half-step down (flat)	0	0	0	0	1
One octave higher	0	1	0	0	0
False Negative: 0.3%		**False Positive: 0.4%**			

Table 2. Experimental results for major triad chords.

Test Cases	**C**	**D**	**E**	**F**	**G**
Correct Chord	60	58	60	60	60
Half-step up (sharp)	1	0	0	0	0
One octave higher	0	0	0	9	0
One wrong note	0	0	0	0	0
One missing note	1	0	0	0	10
False Negative: 0.6%		**False Positive: 1.8%**			

4.2. Real-time System Performance

To evaluate the accuracy of feedback in our real-time system, we randomly selected ten sight-reading exercises of varying difficulty and used music software to generate three audio files for each exercise, testing different conditions. The first file played the exercise perfectly; the second transposed the exercise to a different key; the third file transposed the exercise by exactly one octave. Each audio file was then played through a set of speakers next to a laptop running the sight-reading application. When the exercise was played perfectly, the system recognized 96% of the notes as correct; we note that exercises with slower tempos had nearly perfect accuracy while exercises with faster tempos contributed most to this overall error rate. On the other hand, when the exercise was played in a different key or octave, 98.2% of notes were successfully rejected. In future work, we intend to address these remaining errors by more systematic tuning of cost-of-fit thresholds and by rapidly adapting NMF templates to the student's individual instrument.

5. CONCLUSION

In this paper we have described a real-time system for sight-reading evaluation. The pattern-matching in the back-end of our system is based on NMF. In particular, we use NMF to learn templates for musical notes in an offline setting. Reconstruction errors from these templates are then used to analyze musical input in real-time and match it against a desired score. Overall, the system is very accurate. The system is designed for beginning sight-readers on acoustic instruments without digital pickups or MIDI interfaces. Future work will focus on the ability to give partial credit when only a subset of notes are played correctly, as well as adding new instruments, such as the recorder, violin, and saxophone.

6. REFERENCES

[1] A. Lehmann and V. McArthur, "Sight-reading: Developing the skill of reconstructing a musical score," in *Science and Psychology of Music Performance*, R. Parncutt and G McPherson, Eds., pp. 135 – 150. Oxford University Press, 2002.

[2] A. de Cheveigne, "Multiple $f0$ estimation," in *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*, D.-L. Wang and G. J. Brown, Eds., pp. 45–80. John Wiley & Sons, Inc, 2007.

[3] M. Goto, "Analysis of musical audio signals," in *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*, D.-L. Wang and G. J. Brown, Eds., pp. 251–296. John Wiley & Sons, Inc, 2007.

[4] D. D. Lee and H. S. Seung, "Learning the parts of objects with non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[5] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[6] S. A. Abdallah and M. D. Plumbley, "Polyphonic transcription by non-negative sparse coding of power spectra," in *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, pp. 318–325. Barcelona, Spain, 2004.

[7] F. Sha and L. K. Saul, "Real-time pitch determination of one or more voices by nonnegative matrix factorization," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds., pp. 1233–1240. MIT Press, Cambridge, MA, 2005.

[8] A. Cont, "Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical HMMs," in *IEEE International Conference in Acoustics and Speech Signal Processing (ICASSP)*, Toulouse, France, May 2006.

[9] T. Virtanen, "Monaural sound source separation by non-negative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15(3), pp. 1066–1074, 2007.