Automatic Real-Time Music Generation for Games

Steve Engels, Fabian Chan, and Tiffany Tong
Department of Computer Science, Department of Engineering Science, and Department of Mechanical and Industrial Engineering
40 St. George Street
Toronto, Ontario, M5S 2E4

Abstract

Music composition can be a challenge for many small- to medium-sized game companies, largely due to the expense and difficulty in creating original music for each level of a game. To address this, we developed a tool that automatically generates original music, by training a music generator on pieces whose style the game designer wishes to imitate. The generator then creates original music in that style in real-time, and switches between styles when signaled by the game. This software has been refined to produce music that is coherent and imitates a composer's larger music structure.

Introduction

Music is an essential part of the video game experience, but having a variety of original, high-quality soundtracks into a game can be expensive. There is a financial cost involved in commissioning a soundtrack, and a memory cost in storing them. Ideally, in-game music should be cheap to compose and store, and each region of a game level should produce original music endlessly, segueing seamlessly and sensibly from one soundtrack to the next.

Early work in automatic music generation involved techniques such as handcrafted rule-based composition, and extraction and recombination of music segments from existing pieces (Cope 2006). Other research has created genre recognition systems effective at distinguishing between composers, and allude to their potential application to music generation (Lo & Lucas, 2006).

Recent systems train on existing pieces to generate original compositions. One example involved work in rhythm generation, where stochastic models analyze and store rhythmic patterns from existing pieces to generate similar but novel patterns (Marchini and Purwins 2011; Sioros and Guedes 2011). Chord extraction from music has also previously explored (Simon, Morris and Basu, 2008).

The Music Generation System

We have implemented a music composition tool that generates original music that is inexpensive to produce and store. By training itself on existing compositions, this tool composes original and sensible musical scores in real-time.

The tool is based on a variation of a Markov model, which is a technique that has been also been used by (Marchini and Purwins 2011; Pachet, 2003; Sioros and Guedes 2011). The core generator was first created in 2011 by (Engels and Eisner, 2011), using a Markov model where each state is a collection of notes that share a common starting position within a bar. A composition is broken down into these states, and the resulting Markov model would stores the frequencies of notes and transitions within that composition (Figure 1). Each model could then be used to generate polyphonic segments with output that respects rhythmic boundaries, similar to the original piece.

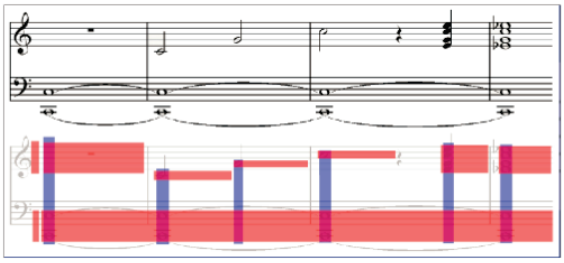

Figure 1: Extraction of polyphonic segments from original piece.

The resulting tool generated novel music in the style of an original piece, by recombining portions of the original work in new ways, within the bounds of the note transitions in the original piece. The trained models are small and thus easy to store (around 500kb per model), and could generate the notes of a new piece of music in real-time.

Our automatic music generation system facilitates track switching as in-game characters move from one level region to another. Instead of relying on fading between tracks or human-designated transition points, the system

transitions when the current model generates a note sequence that the next model might also generate. The new model continues composing its soundtrack where the previous model leaves off, resulting in a seamless transition.

The Hierarchical Markov Model

More sophisticated music required a hierarchical Markov model, to produce the different sections of a piece (e.g. overture, crescendo, denouement). We automatically segmented each piece into sections, trained separate models for each section, and calculated the transition probabilities between sections to use in the generation stage.

Our automatic segmenter detects similar recurring passages in a piece, and segments the entire piece into large component passages. The similarity metric used is based on a Levenshtein distance calculation, using the pitch, duration, timbre and volume as the main factors for its edit distance calculation (Figure 2).

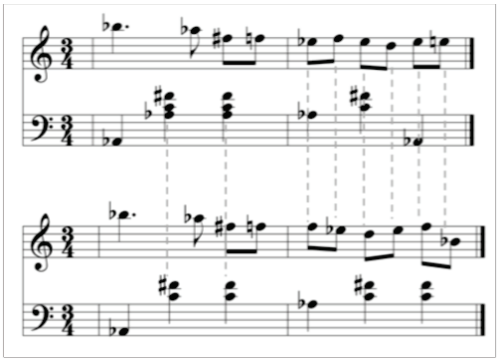

Figure 2: Calculating edit distances between two segments of Chopin's Minute Waltz. Vertical dotted lines denote mismatches.

A support vector machine classifier with a radial basis function (rbf) kernel is used on the results of this calculation, to determine matching passages. The classifier is trained using labeled data from the original piece. Finally, the segmenter finds the optimal set of segments with non-overlapping coverage, by optimizing Equation 1:

$$S = \sum_{segments}(frequency^2 + length) \qquad \text{Eq. 1}$$

By treating this as an interval-scheduling problem that uses greedy selection based on the value of S, the result is a near-optimal segmentation, calculable in polynomial time.

The segmentations are illustrated with a recurrence plot below, whose major axes represent the piece's bars, and darker elements indicate higher similarity (Figure 3). The segmenter uncovers the largest similar sections, and trains a Markov model within each section, thus producing music that is characteristic to the beginning, middle and end of a piece, depending on its original structure.

Figure 3: Recurrence plot for Hilarity by J. Scott. Notice small- and large-scale similarity structures marked in red.

Chord Progression as Hidden States

In addition to segmenting music pieces into sections for training purposes, this tool incorporates hidden states within each section to produce a higher-level progression. After exploring several options, chord progression were the strongest predictor for continuous, coherent note generation between sections.

This produced a hidden Markov model (HMM), where chords served as the hidden states, similar to the work of (Simon, Morris and Basu, 2008). In cases where the composer had not tagged the training data with chords, our tool tagged bars automatically with the chord that best matched the notes and similarly tagged bars from other pieces. The result created a more natural flow throughout the generated piece, and also allowed models trained on multiple pieces.

A Mixture of Models

By adding chord label information to each state, the tool could now train a single model on multiple pieces. This reduces sparseness in each model, while maintaining musical coherence through the chord structure. Adjustments still need to be made during training, since pieces may be slightly incompatible due to key or meter differences. To address this issue, multiple input pieces were pitch-shifted to ensure that note transitions were compatible.

Discussion

The result of this work is a tool that can produce original music in real-time, in the style of a training piece. It can change styles from one game region to another, and presents little additional load on the processor. The incorporation of high-level progression with hidden states based on chord transitions has added a higher level of composition to the generated pieces. The result is a tool that allows game studios to have original music for their games, while minimizing the time, memory or financial cost of requiring human composers.

References

Cope, D. eds. 2006. *Computer Models of Musical Creativity*. *Cambridge*, MA: MIT Press.

Lo, M. Y. and Lucas, S. M. 2006. Evolving Musical Sequences with N-Gram Based Trainable Fitness Functions. In *IEEE Congress on Evolutionary Computation*.

Engels, S. and Eisner, D. 2011. Automatic Real-Time Music Generation. *Game Developers Conference*, San Francisco, CA.

Marchini, M. and Purwins, H. 2011. Unsupervised Analysis and Generation of Audio Percussion Sequences. *Lecture Notes in Computer Science* 6684:205-218.

Pachet, F. 2003. The Continuator: Musical Interaction With Style. *Journal of New Music Research* 32:333-341.

Sioros, G. and Guedes, C. 2011. Automatic Rhythmic Performance in Max/MSP. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 88-91.

Simon, I., Morris, D., and Basu, S. 2008. MySong: Automatic Accompaniment Generation for Vocal Melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 725-734. Association for Computing Machinery, Inc.