COPYFORWARD: POINT-SET MATCHING FOR PREDICTING PATTERNS

Timothy de Reuse
Centre for Interdisciplinary Research in Music Media and Technology
McGill University
`timothy.dereuse@mcgill.ca`

ABSTRACT

This document describes *CopyForward*, an algorithm submitted to the 2019 MIREX task on Patterns for Prediction, for the symbolic polyphonic and symbolic monophonic versions of subtask 1, which involves predicting the continuation of a given excerpt of music. *CopyForward* is a simple point-matching algorithm that predicts the continuation of an excerpt of music by selecting a section of the excerpt, copying it, and translating the copy to the end of the excerpt. Its limitations are discussed, as well as possible avenues for improvement.

1. INTRODUCTION

A stated goal of the Patterns for Prediction MIREX task is "to interrogate the idea that patterns are abundant in music and always informative in terms of predicting what comes next" [3]. The *CopyForward* [1] algorithm addresses this idea directly, motivated by the geometric pattern-matching approaches to pattern discovery used by algorithms like COSIATEC [2]. In contrast to previous entries to this task, this model searches explicitly for patterns in its input data; it uses no machine learning techniques, knows nothing of tonal expectations or musical convention, and relies entirely on the assumption that future events will be repetitions of events that have already happened. The algorithm is designed to run on `.csv` files containing lists of (note onset time, MIDI note number) pairs, ignoring the other per-note fields included in the Patterns for Prediction Development Dataset (PPDD). It is also agnostic to whether its input is monophonic or polyphonic, so the same code can be used for both tasks.

Training neural networks to learn from and predict long-term dependencies is a notoriously difficult problem [1]. It is hoped that this simple algorithm might prove a useful benchmark against which more sophisticated algorithms can be judged; intuitively, an algorithm that can learn to detect and replicate patterns in a rather naive way ought to do at least as well as *CopyForward*.

[1] `github.com/timothydereuse/copy-forward`

2. ALGORITHM DETAILS

The assumption underpinning *CopyForward* is that most input primes contain their continuations, and so the act of predicting future events can be reduced to choosing an excerpt from the prime and translating it forward in time (and, optionally, transposing it up or down in pitch). In effect, the task has been reduced to finding a single vector which defines a region of the prime to copy as a continuation. Given a desired continuation length c, a window length $w < c$, and a prime represented as a series of (note onset time, MIDI note number) pairs, where all its events lie between p_{start} and p_{end}:

1. Split the prime into two parts: the events which lie after $p_{end} - w$ (the *fixed window*), and all preceding events (the *sliding window*).
2. Find a vector $v \in \mathbb{Z}^2$ such that translating the sliding window by v maximises the number of coinciding points between the sliding window and the fixed window.
3. After translating the sliding window, extract all points that lie between p_{end} and $p_{end} + c$; this new set of points is the predicted continuation of the prime.

The algorithm as submitted contains other pre-processing steps and failsafes in place to avoid degenerate solutions. First, it multiplies all time values by 12 and then quantizes to the nearest integer to ensure that no errors are caused by floating-point inaccuracies. On output all time values are divided by 12 and rounded to five decimal places, to match the convention of the PPDD. In the scenario where $p_{end} - p_{start} < c$, the prime is assumed to repeat backwards in time indefinitely. It is also possible for the predicted continuations to contain just one event for certain inputs; when the fixed window is sparsely populated with events, the best possible translation vector v might be very small, shifting few events from the prime into the range $(p_{end}, p_{end}+c)$. These cases are exceptionally rare, and on encountering them the continuation is filled with a series of "dummy" notes so as not to produce an empty output.

For this MIREX task, the value for c has been set to ten quarter-note beats, so the only parameter left to set in this algorithm is the size of the fixed window w. Brute-force testing over a range of choices for w showed that using a window size of 8 quarter-note durations is moderately

	Polyphonic			Monophonic		
Algorithm	Recall	Precision	F1	Recall	Precision	F1
CopyForward	0.451	0.463	0.445	0.496	0.503	0.493
CopyForward with “cheating”	0.562	0.553	0.547	0.645	0.644	0.638

Table 1. The results of evaluating CopyForward on the polyphonic and monophonic versions of PPDDs. The scoring system here is the cardinality score, as defined in the guidelines for the MIREX task; results are the average over 1000 randomly selected entries from the monophonic and polyphonic versions of the PPDD.

more successful than shorter or longer windows in evaluation on the PPDD for both the monophonic and polyphonic data, so this is what the parameter is set to in the submission.

To get a sense of the limits of this approach, we also evaluate using a method that can “cheat” by finding the translation vector v that results in the highest number of coinciding points between the sliding window and the true continuation. A comparison between this method and the submitted version of *CopyForward* is shown in Table 1. The results represent the best possible performance for any algorithm which assumes the continuation to be an unedited repetition of material from somewhere in the prime.

3. DISCUSSION

Compared to results from 2018, the only previous year when this task was run, *CopyForward* performs similarly to the first-order Markov model baseline on the monophonic task, but significantly better than the Markov model and the one other submission on the polyphonic task (assuming that the test results from that year can be compared directly with evaluation results on the PPDD). On both tasks, however, *CopyForward* performs significantly worse than than the “cheating” method, indicating that a more sophisticated method of choosing the translation vector v could still lead to significant gains in performance.

It is worth noting here a few of the techniques attempted during the development of this algorithm that did *not* improve its performance. Splitting each prime into its constituent MIDI channels and recombining the individual predicted continuations for each channel resulted in some predicted continuations becoming more accurate but an equal amount becoming less accurate. A variety of different similarity functions were also tested out, with the rationale that perhaps maximising some metric of similarity between the fixed window and sliding window other than simple pointwise overlap would result in better performance; these efforts also generally performed worse than the simple method described here.

These failed efforts lead to the hypothesis that at least some of the discrepancy between the current performance of *CopyForward* and its “ideal“ performance is unavoidable. On primes whose true continuations are not at all repetitions of content contained in the prime, there likely exist translation vectors that result in some amount of overlap just by pure chance; however, it would be nearly impossible to identify these vectors by looking only at the prime. While some improvement to this algorithm is possible, there is likely a limit to how much better it can be without using more sophisticated methods capable of incorporating information on musical style and tonal expectations.

4. REFERENCES

[1] Hochreiter, Sepp, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. “Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-term Dependencies.” In Stefan C. Kremer & John. F. Kolen (Eds.) *A Field Guide to Dynamical Recurrent Neural Networks.* IEEE Press, 2001.

[2] Meredith, David. “COSIATEC and SIATECCompress: Pattern Discovery by Geometric Compression.” In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, Curitiba, Brazil, 2013.

[3] The MIREX Wiki. “2019:Patterns for Prediction.” `www.music-ir.org/mirex/wiki/2019:Patterns_for_Prediction`. Accessed 8 September 2019.