Neural Dynamic Programming for Musical Self Similarity

Christian J. Walder [1,2] and Dongwoo Kim [1,2,3]

[1] Data61, CSIRO, Australia [2] The Australian National University [3] Data to Decisions CRC, Kent Town, SA, Australia

☆ See also our related poster ☆

Self-Bounded Prediction Suffix Tree via Approximate String Matching

Hall B #112

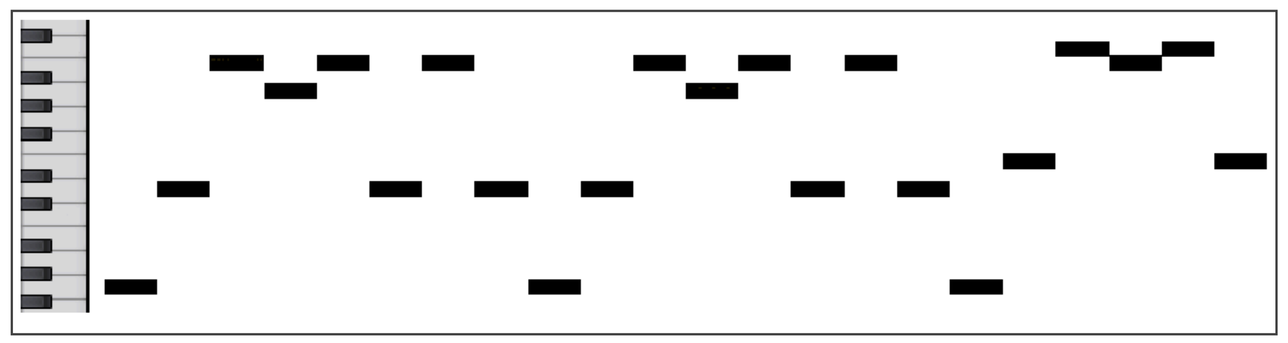
Abstract

We present a neural sequence model designed specifically for symbolic music. The model is based on a learned edit distance mechanism which generalises a classic recursion from computer science, leading to a neural dynamic program.

Repeated motifs are detected by learning the transformations between them. We represent the arising computational dependencies using a novel data structure, the edit tree; this perspective suggests natural approximations which afford the scaling up of our otherwise cubic time algorithm.

We demonstrate our model on real and synthetic data; in all cases it out-performs a strong stacked long short-term memory benchmark.

What Real Music Looks Like



How do you expect the above sequence continue?

Answer:

by ***analogy*** *with first two cycles, the last two notes may repeat immediately.*

Note that in the third cycle of the motif above, we observe a non-trivial diatonic (i.e. within musical scale) shift of the upper notes

Detecting and completing the pattern may be more important than learning general short-sequence regularities, for symbolic music!

More Sequence Completion Puzzles

1, 4, 9, 16, ___

1, 64, 9, 100, 25, 16, ___

1, 7, 2, 9, 1, 7, 2, ___

1, 3, 1, 12, 17, 101, 103, 101, 112, ___

1, 3, 1, 12, 17, 101, 103, 33, 101, 112, ___

1, 3, 1, 12, 17, … , 101, 103, 33, 101, 112, ___

Due to the chain rule of probability:

$$p(S) = \prod_{i=0}^{|S|-1} p(s_{i+1}|S(:i))$$

we can model transformed motifs by comparing the current suffix with all previous subsequences!

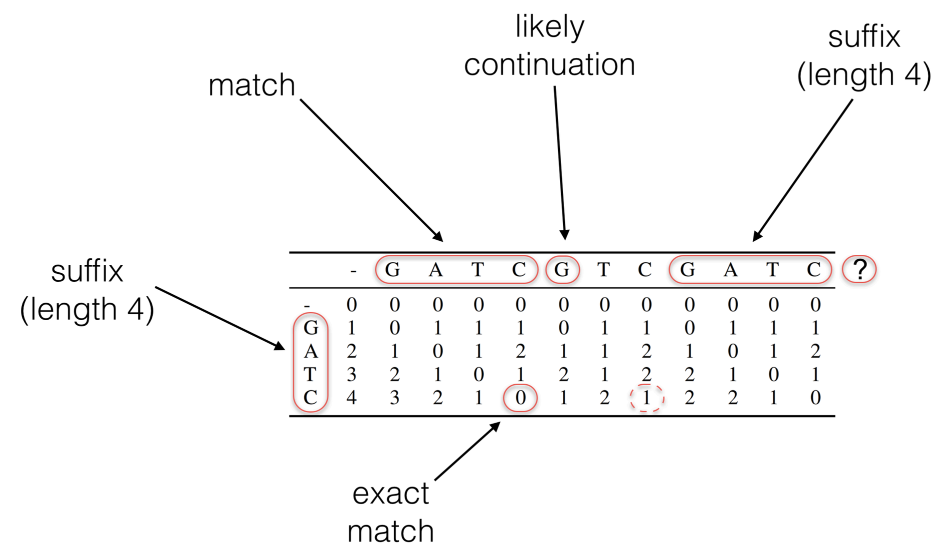
Edit Distance / Self Matching

- *The edit distance is the minimum total cost of edit operations (insertions, deletions, and substitutions) which transform one sequence P to another T*
- *It is computed by the dynamic program:*

$$D(i,j) = \min \begin{cases} c(p_i \to \epsilon) + D(i-1,j) \\ c(p_i \to t_j) + D(i-1,j-1) \\ c(t_j \to \epsilon) + D(i,j-1) \end{cases}$$

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

- *With Seller's modification, this allows searching within a longer string:*



Searching for exact self matches and forecasting based on the subsequence continuation, is a natural scheme which generalises various models (context tree weighting, prediction suffix trees, variable order hidden Markov models..).

- *Further slight generalisation of the dynamic program yields all self matches:*

$$D_s(i,j,k) = \min \begin{cases} c(s_i \to \epsilon) + D_s(i-1,j,k-1) \\ c(s_i \to s_j) + D_s(i-1,j-1,k-1) \\ c(s_j \to \epsilon) + D_s(i,j-1,k). \end{cases}$$

current time (12 above) previous time match length (4 above)

This is the key program control flow. We generalise the entire setup by replacing the constituent operations with parameterised functions. The parameters are learned end to end via gradient based optimisation.

MotifNet Generalisation

Simplified Setup	MotifNet
categorical symbol	learned embedding vector
unit edit cost	learned edit cost function
scalar distance	generalised distance vector
distance ← distance + cost	distance ← GRU(distance, cost)
dynamic programming min	arg max w.r.t. learned scoring function
forecast by exact matching	learned weighting function averaging
identity forecast	analogy forecast

The ***analogy*** *function is an important component. If we have seen 1, 3, 1, 12, 17, then by* ***analogy*** *after 101, 103, 101, 112 we expect to see 117.*

The Basic (exact / slow) MotifNet

Notation	Interpretation of Elements
$\Sigma = \{1,2,\ldots,\|\Sigma\|\}$	Discrete symbol
$\mathcal{E} = \mathbb{R}^{N_E}$	Embedding
$\mathcal{C} = \mathbb{R}^{N_C}$	Generalised edit cost
$\mathcal{D} = \mathbb{R}^{N_D}$	Generalised distance
$\mathcal{O} = \mathbb{R}^{N_O}$	Penultimate layer

Notation	Role	Architecture	Mapping
f_E	embedding	lookup	$\Sigma \to \mathcal{E}$
f_D	deletion	FF	$\Sigma \to \mathcal{C}$
f_S	substitution	FF	$\Sigma \times \Sigma \to \mathcal{C}$
f_A	addition	GRU	$\mathcal{D} \times \mathcal{C} \to \mathcal{D}$
f_W	scoring*	FF	$\mathcal{D} \to \mathbb{R}$
f_G	analogy	FF	$\mathcal{D} \times \mathcal{E} \to \mathcal{O}$
f_F	forecasting	FF	$\mathcal{O} \to \mathbb{R}^{\|\Sigma\|}$

Algorithm 1 MotifNet generalised distance.

Input: $S = s_1 \cdot s_2 \cdots s_{|S|}, f_E, f_A, f_S, f_D, D_0$
Output: $D(i,j,k)$
for $i = 1$ **to** $|S|$ **do**
  for $k = 1$ **to** i **do**
    if $k = 1$ **then**
      for $j = 1$ **to** i **do**
        $D(i,j,k) \leftarrow f_A(D_0, f_S(s_i, s_j))$
      end for
    else
      for $j = 1$ **to** i **do**
        $D_\downarrow \leftarrow f_A(D(i-1,j,k-1), f_D(s_i))$
        $D_\searrow \leftarrow f_A(D(i-1,j-1,k-1), f_S(s_i,s_j))$
        $D_\rightarrow \leftarrow f_A(D(i,j-1,k), f_D(s_j))$
        $D(i,j,k) \leftarrow \underset{D' \in \{D_\downarrow, D_\searrow, D_\rightarrow\}}{\operatorname{argmax}} f_W(D')$
      end for
    end if
  end for
end for

- *Given the above generalised distances D(i, j, k), the penultimate layer is*

$$O_i = \sum_{0 \le j < i} \sum_{0 \le k < i} w_{i,j,k} f_G(D(i,j,k), f_E(s_{j+1}))$$

where

$$w_{i,j,k} = \frac{\exp(f_W(D(i,j,k)))}{\sum_{0 \le j' < i} \sum_{0 \le k' < i} \exp(f_W(D(i,j',k')))}$$

and the forecast is given by $s_{i+1}|S(:i), \ldots \sim \mathrm{Discrete}(f_F(O_i))$.

Edit Tree

- *The generalised distances D(i, j, k) are functions of an* ***alignment sequence****.*
- *This suggests the edit tree, which differs from a suffix tree as follows:*

	Suffix Tree	Edit Tree
Edges	Symbols: s_i	Matches: $s_i \to s_j$
Nodes	Sequences	Alignments
Fan Out	Linear (in the alphabet size)	Quadratic
Candidate Matching Paths for Prediction	One	Many
All Candidates Relevant?	Yes (exact matches)	No (scoring function required)
Prediction	Direct Correlation (reoccurance)	Analogy
Generalises the Other	No	Yes
Theoretical Guarantees?	Yes	Not yet
Studied?	Yes; extensively	No

We **prune the edit tree** using a learned heuristic function, obtaining tractability!

Experiments

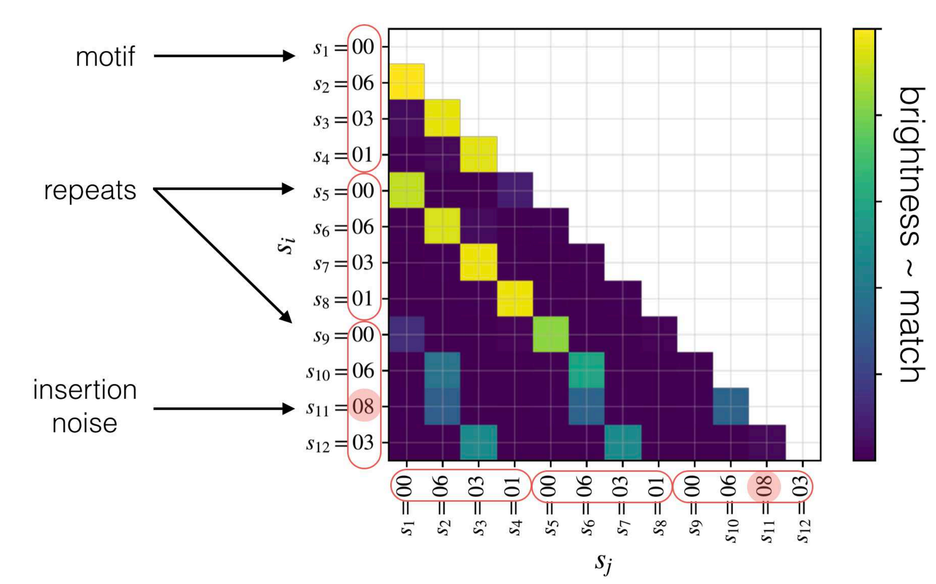


Table 2. Average test set negative log likelihood for a stacked LSTM, MotifNet, and their combination (see subsection 4.3) on real symbolic music problems. See subsection 6.2 for more details.

	JBM	MUS	NOT	PMD
LSTM	1.82	2.03	1.03	2.67
MotifNet	1.77	1.88	0.81	1.90
MotifNet+LSTM	1.79	1.83	0.73	1.85

These are basic CPU-only experiments, but we already improve on the LSTM. Scaling up **MotifNet** seems promising!