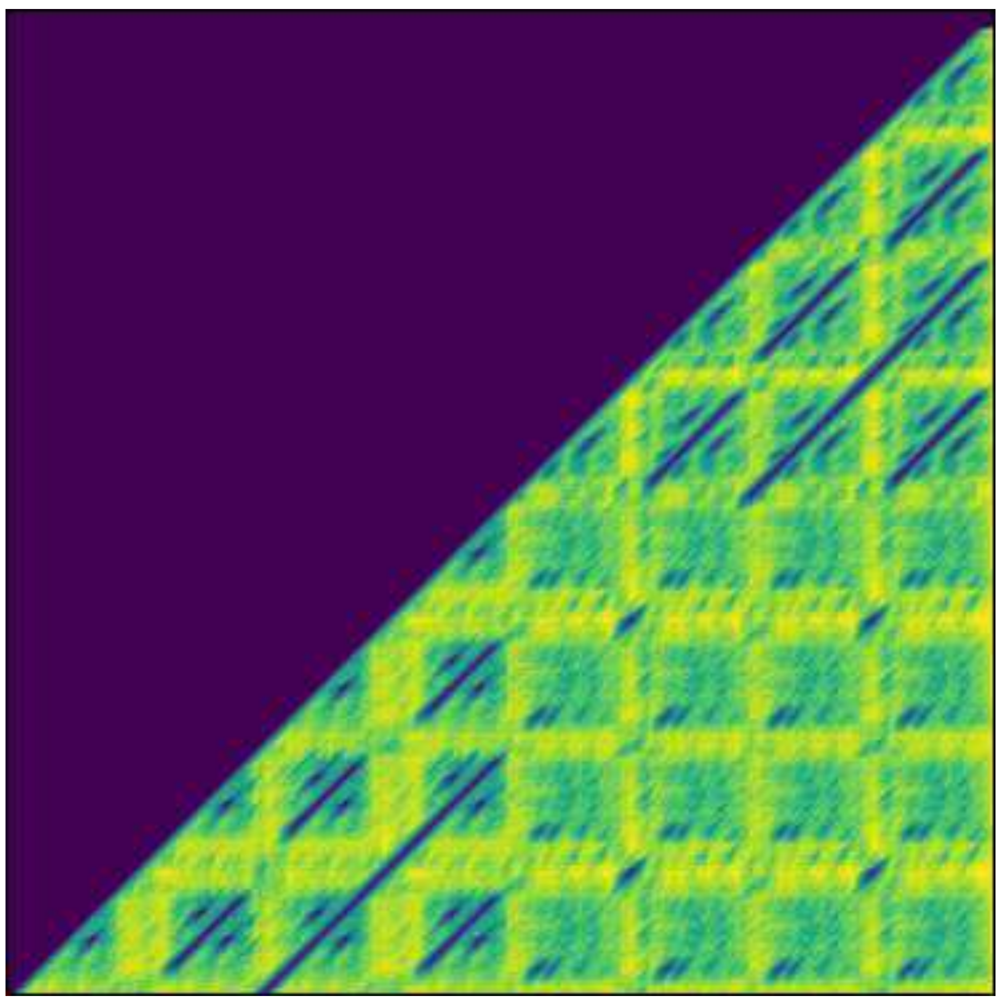
Neural Dynamic Programming for Musical Self Similarity



Christian Walder[1,2]
with Dongwoo Kim[2]

[1]Data61
[2]Australian National University

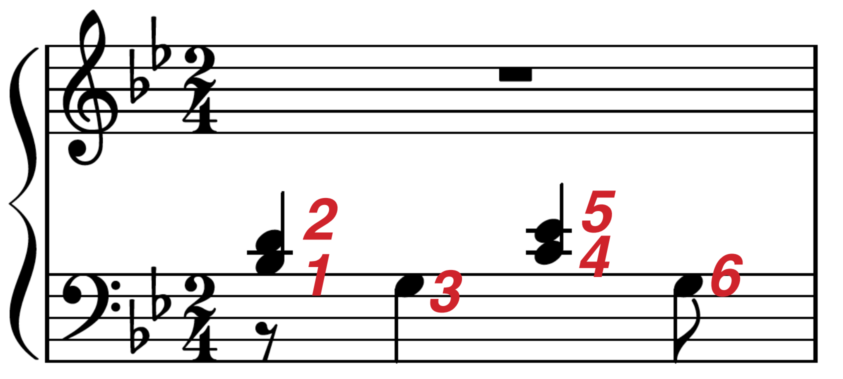
Overview

- Background: Models of Symbolic Music
    - representation and sequence models
    - limitations
    - canonical example model: prediction suffix tree
- Modelling Goals
- Model: Simplified Setup
    - autoregressive / “method of analogy” with edit distance
- Model: General Setup
    - generalised edit distance
    - forecasting by analogy
- Edit Tree
    - basic idea
    - comparison with suffix tree
- Summary / Outlook

Background:
Models of Symbolic Music

Symbolic Music Models

- Convert to a sequence, *e.g.*



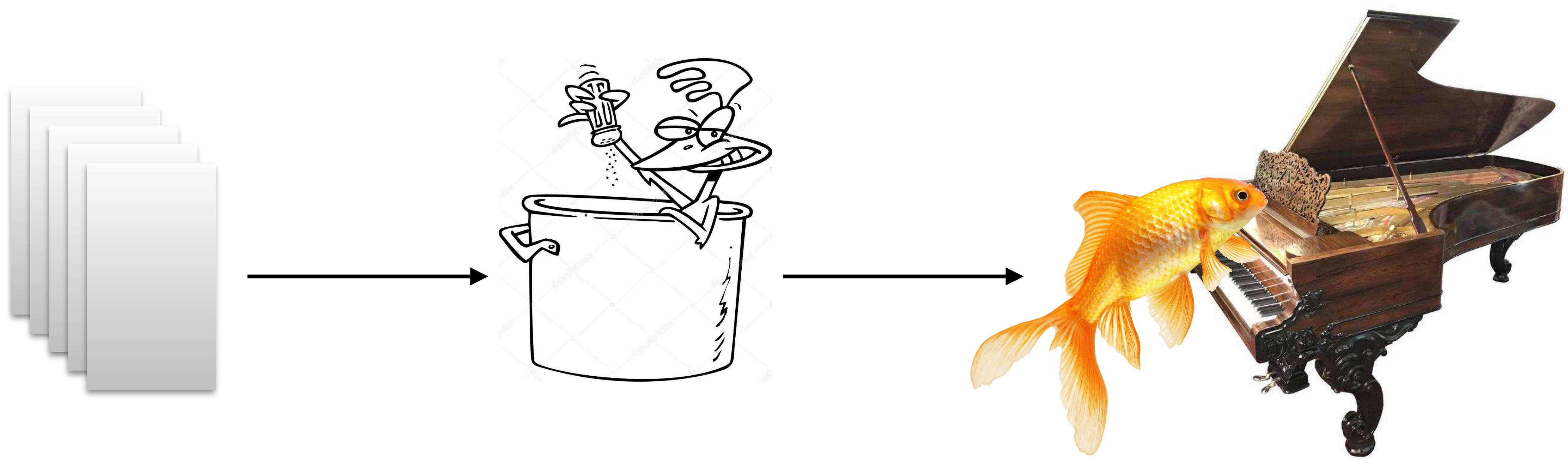
- Exploit the chain rule:

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

- Apply a sequence model, typically the LSTM

Why This is Lacking

- The holy grail of RNNs is long term dependence:
    - LSTM style additive updates help
    - memory remains highly limited in practice
- With a short memory we learn simple regularities:
    - predict the immediate future from the immediate past
- Longer term we only capture simple regularities like musical key
- We mainly compare the recent past to
    - similar scenarios in our training set
    - (rather than autoregression, *i.e.* comparing to similar scenarios in the current piece)
- Those models that do autoregression rely on simple exact matching (next slide)



data: pieces of music $\mathcal{D}$

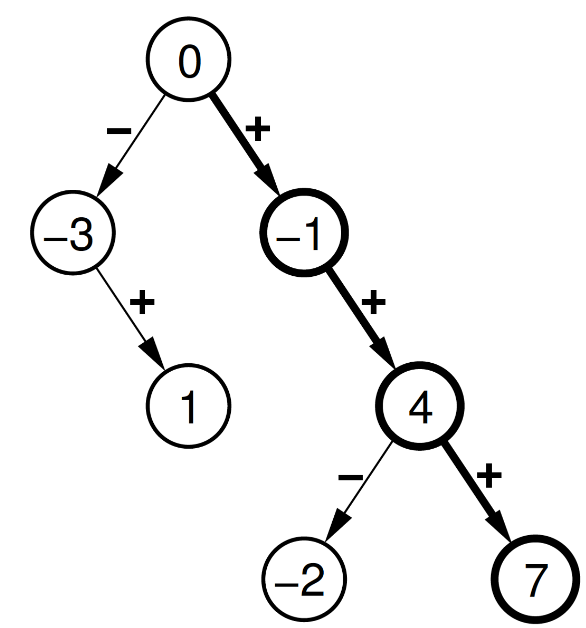
soup of
(subsequence, next symbol)
pairs

amnesiac music: compares the **suffix** of previous output to the soup of training data fragments

Example Sequence Model: Prediction Suffix Tree

$$h(S(1:i)) = \text{sign}\left(\sum_{k:S(i-k:i)\in\mathcal{T}} f(S(i-k:i))\right)$$

- **f**: nodes → scores
- **h** forecasts the next symbol
- Example:
  - $\mathcal{T}$ = {ε, -, +, +-, ++, -++, +++}
  - **f**: see the figure
  - S(1:6) =(-,-,+,+,+,+)
  - forecast: S(7) = sign(-1+4+7)=+1

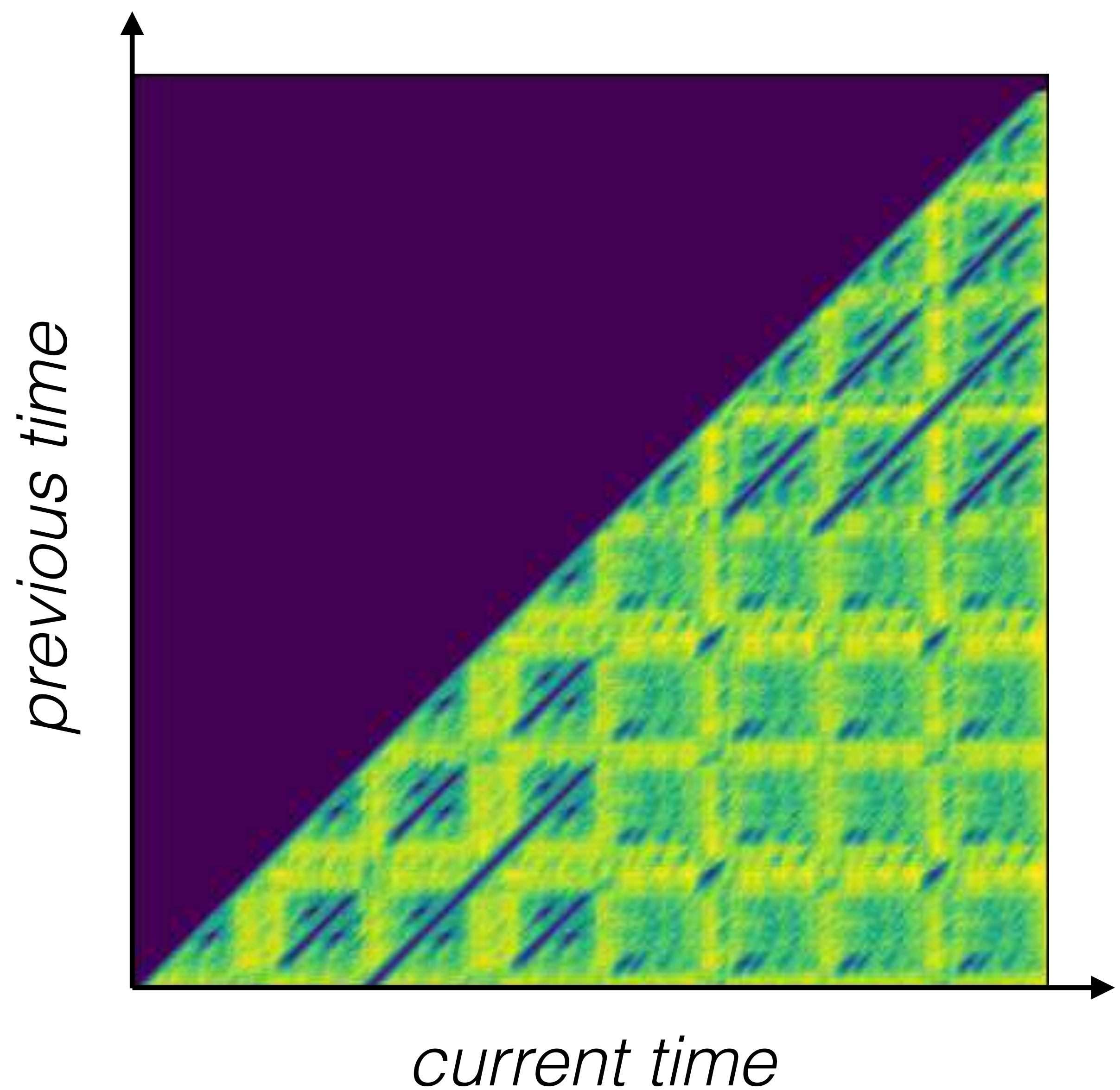


The Power of Selective Memory: Self-Bounded Learning of Prediction Suffix Trees.
Ofer Dekel, Shai Shalev-Shwartz, Yoram Singer
NIPS 2005

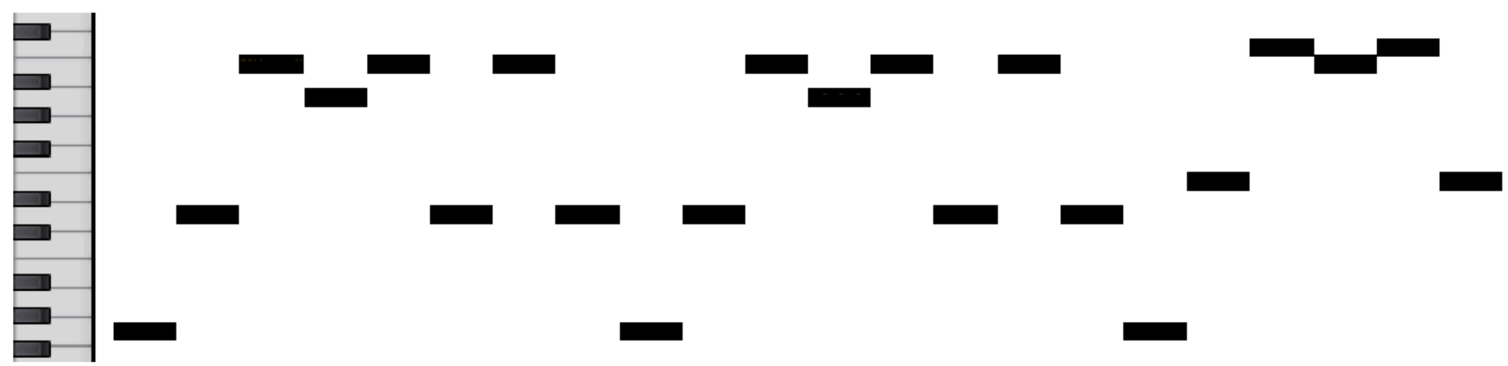
Modelling Goals

What Real Music Looks Like (1)

Darker ~ More Similar (edit distance)



What Real Music Looks Like (2)



Bach's Cello Suite No. 1 in G major: Prelude

- By **analogy** with first two cycles:
  - the last two notes should repeat immediately
- In the third cycle of the motif we observe:
  - a non-trivial musical shift of the upper notes

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, ___

We Need to Learn to Transform and Repeat

- The next note in a piece depends **more** on
  - the ~~similarity~~ relationship between the current suffix and the previously emitted substrings in the same piece
- and **less** on
  - the similarity between the current suffix and the suffixes in the training soup.
- The key is learning the transformations between subsequences within a piece.
- This differs from the method of analogues which considers identity transformations (AFAIK).
- Equivalently to the slide title: due to the chain rule we need to detect transformations from earlier subsequences to the prefix and analogise them.

Model:
Simplified Setup

Simplified Setup

- Compare a suffix S(i - k : i) to all previous suffixes, to forecast S(i + 1)
- Consider all suffix lengths k
- Use approximate matching with edit distance
  - already somewhat novel - see e.g. Dongwoo's talk next week
  - models musical transformations: passing tone, substituted note, *etc*
  - results in self alignment
- Assume history tends to repeat
- This motivation is rather weak, but:
  - it is leading up to general model which achieves our stated goals!

Edit Distance

- The minimum total cost of edit operations (insertion, deletion, substitution) which transform one sequence P to another T

- Solved 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

- Seller's modification:

	-	G	A	T	C	G	T	C	G	A	T	C
-	0	0	0	0	0	0	0	0	0	0	0	0
G	1	0	1	1	1	0	1	1	0	1	1	1
A	2	1	0	1	2	1	1	2	1	0	1	2
T	3	2	1	0	1	2	1	2	2	1	0	1
C	4	3	2	1	0	1	2	1	2	2	1	0

Self Matching Forecast

					j	j+1	i-k		i	?

- The minimum edit distance from S(i-k : i) to S(: j) satisfies the related recursion

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

- Combine with the subsequent continuations (at j+1) to make the forecast (for i+1):

$$s_{i+1} \mid S(:i) \sim \mathcal{S}\left(\{(D(i,j,k), s_{j+1})\}_{0 \le j < i,\, 0 < k < i}\right)$$

- Considering D(i, j, k) = 0 only, gives a prediction suffix tree.

Model:
General Setup

General Setup

- Retain the dynamic programming scheme for self alignment.
- But, in the style of end to end deep learning:
    - infer all the components from data using gradient descent

Generalised Edit Distance

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

- Above: recursion for simple edit distances with ranging i, j, k as before
- Let's generalise:
  - symbols → learned embedding vectors
  - insertion/deletion/substitution cost → learned generalised cost function (vector)
  - scalar distance → generalised "distance" vector
  - addition → recurrent neural network update (GRU):

    $D_{\text{new}} = f_A(D_{\text{old}}, \text{cost})$

  - min → arg max w.r.t. a learned score function

Generalised Forecast: Analogy

- Current time i
- Combine forecasts for all j, k, based on
  - the “distance” D(i, j, k)
  - the observed continuation s_{j+1}
- Simple case: if D(i, j, k) is small s_{j+1} is likely to reoccur
  - e.g. the prediction suffix tree
- General case: **analogy function**

Analogy Function by Example

$$f_G(D(i, j, k), f_E(s_{j+1}))$$

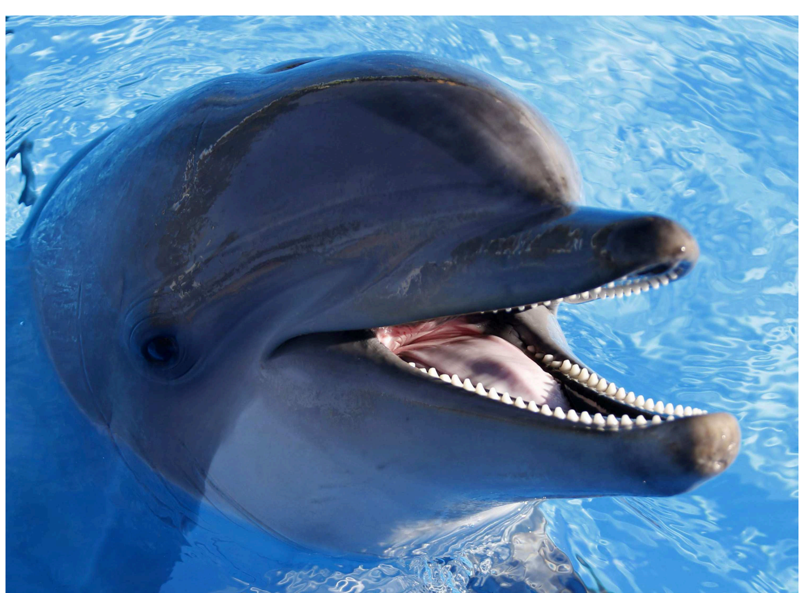
- 1, 3, 1, 12, 17, … , 101, 103, 33, 101, 112, ___
- k = 5
- S(i-k+1 : i) = 101, 103, 33, 101, 112
- S(: j) = 1, 3, 1, 12
- Alignment: (1 → 101), (3 →103), (ε → 33), (12 → 112)
- D(i, j, k): “high certainty add 100 transformation”
- s_{j+1} = 17
- Prediction “high certainty 100 + 17 = 117”

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	$\mathcal{O} \to \mathbb{R}^{\lvert\Sigma\rvert}$

Notation	Interpretation of Elements
$\Sigma = \{1, 2, \ldots, \lvert\Sigma\rvert\}$	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

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**
        $D_\downarrow \leftarrow f_A(D(i-1, j, k-1), f_D(s_i))$
        $D_\searrow \leftarrow f_A(D(i-1, j-1, k-1), f_S(s_i, s_j))$
        $D_\rightarrow \leftarrow f_A(D(i, j-1, k), f_D(s_j))$
        $D(i, j, k) \leftarrow \operatorname*{argmax}_{D' \in \{D_\downarrow, D_\searrow, D_\rightarrow\}} f_W(D')$
      end for
    end if
  end for
end for



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

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

$$s_{i+1} | S(:i), \ldots \sim \mathrm{Discrete}(f_F(O_i))$$

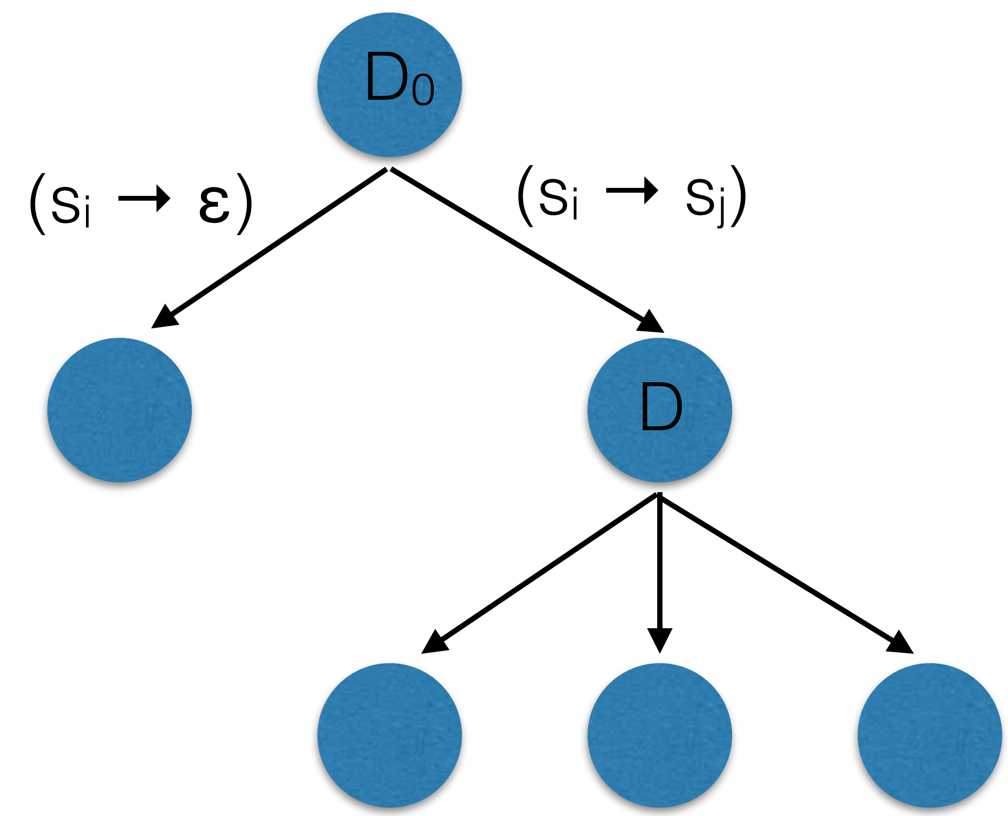
Scoring Function

- Maps generalised distances to scalars
- Performs three roles:
    - Alignment via arg max in distance recursion
    - Weighting of forecasts from analogy function
    - Pruning the edit tree (next slide)
- This coupling is justified by ablative studies

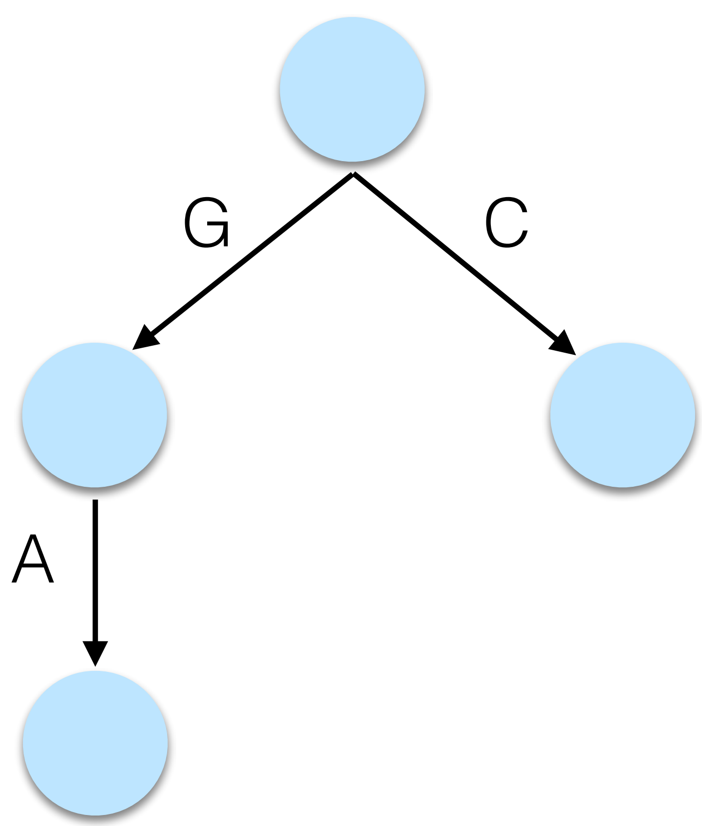
Edit Tree

Edit Tree

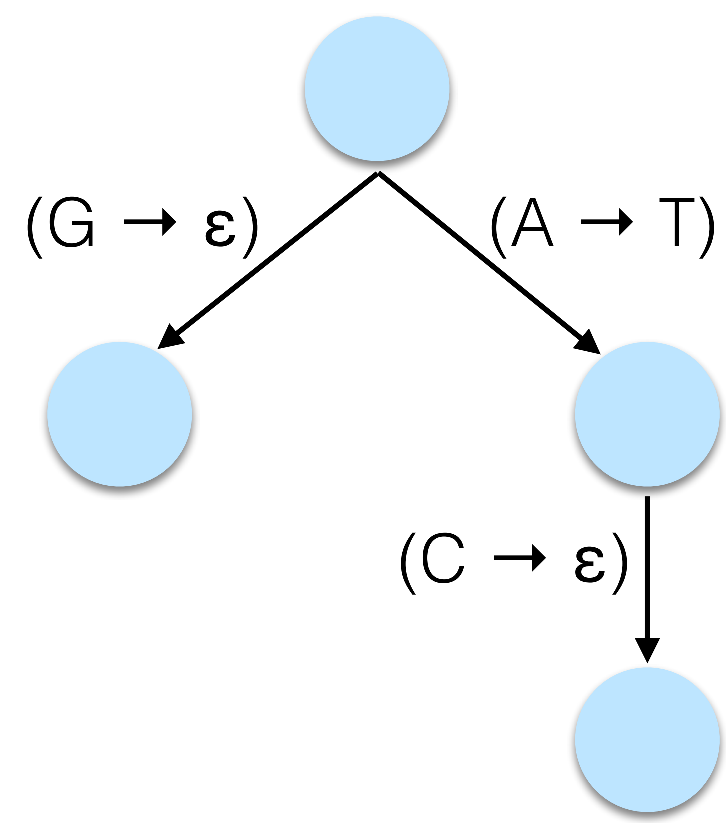
- Edges: insertions/substitutions
- Nodes: generalised distances
- Paths: alignments
- <u>Approximation</u>: prune using scoring function
- Highlights the difference to prediction suffix trees *etc.*
- Prediction requires a list of continuations s_{j+1} at each node.



Suffix Tree



Edit Tree

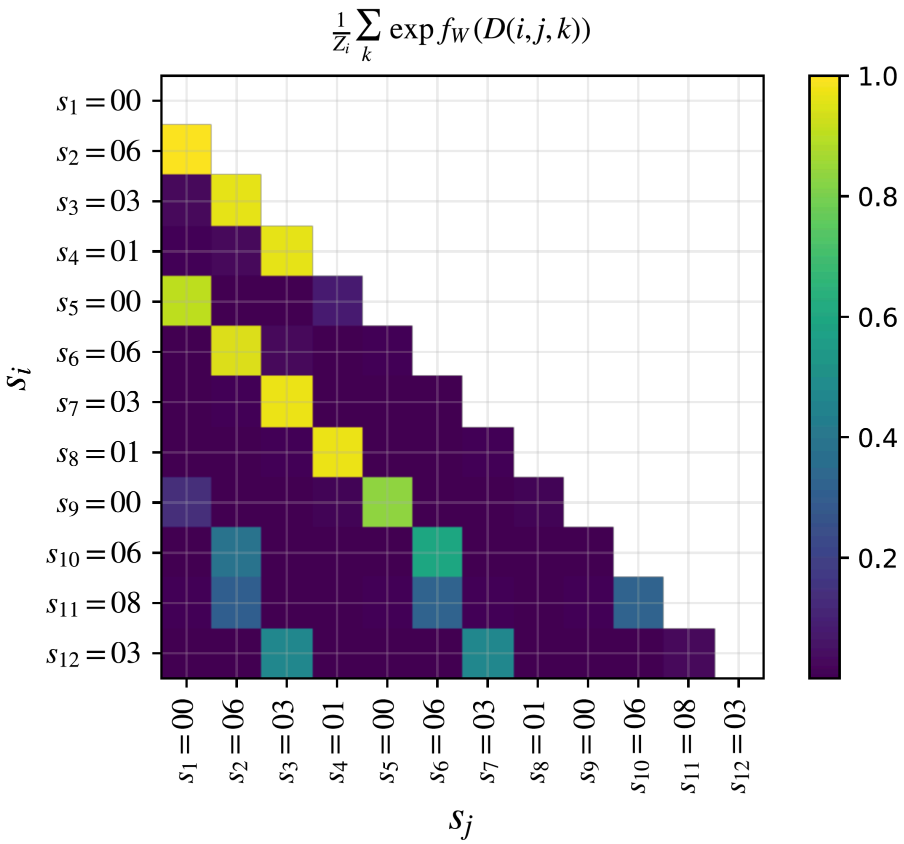


	Suffix Tree	Edit Tree
Fan Out	Linear (in the alphabet size)	Quadratic
Candidate Matching Paths for Prediction	One	Many
All Candidates Relevant?	Yes (candidates are exact matches)	No (scoring function required)
Prediction	Direct Correlation (reoccurance)	General Analogy
Generalises the Other	No	Yes
Theoretical Guarantees?	Yes	Not Yet

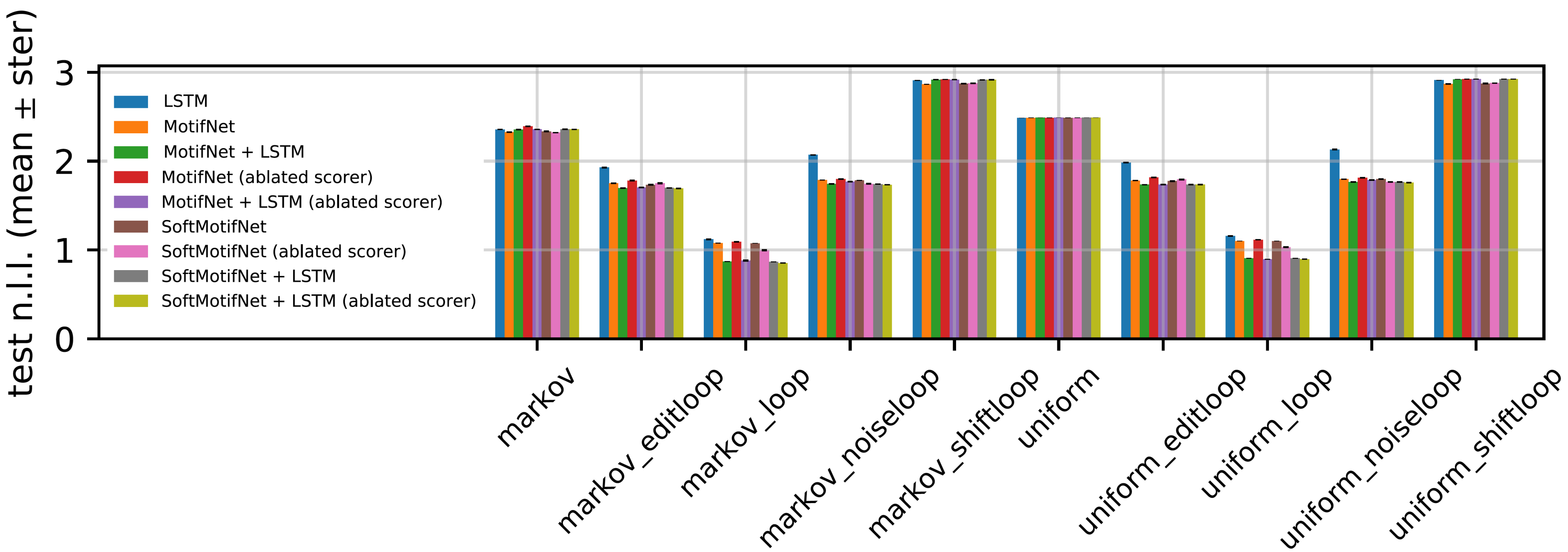
Experiments

Toy Data: Visualising Self-Alignment

- Trained on noisy repeats of motifs
- This example motif is 0,6,3,1
- Brightness ~ match
- Rows i = 5 to i = 8 align
- Rows i > 8 average over two valid alignments
- Insertion noise s_{11} handled



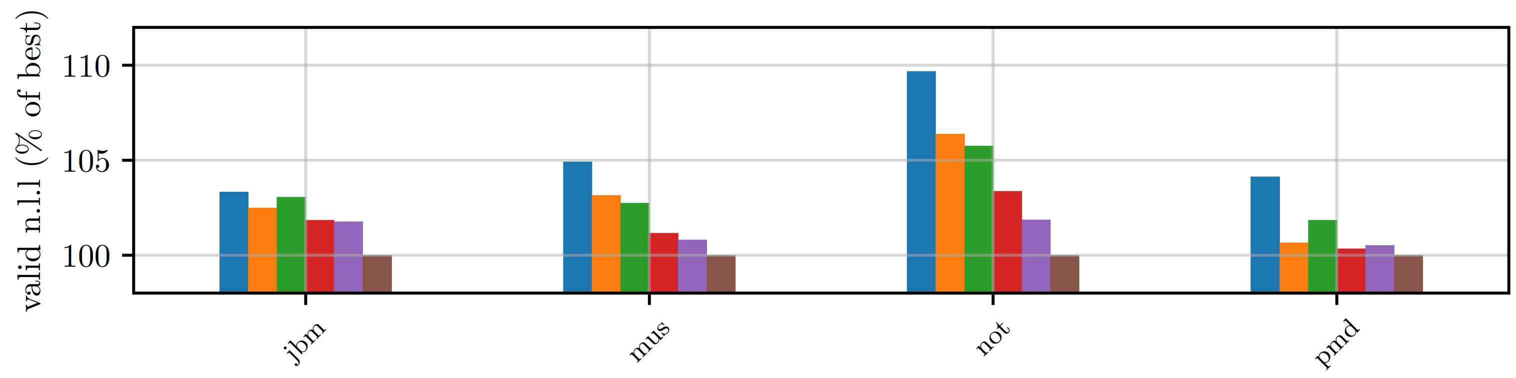
Toy Data



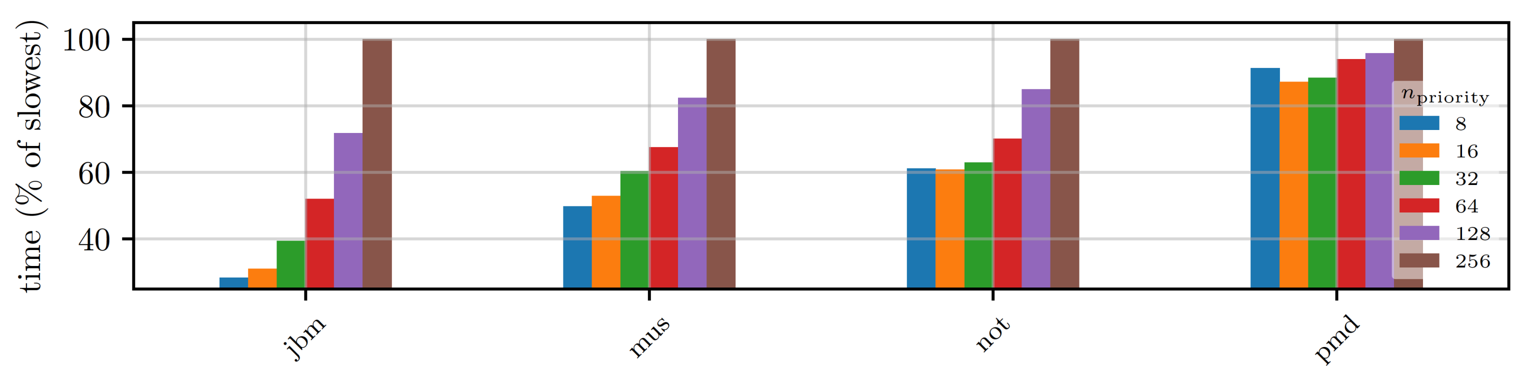
- A suite of test problems
- MotifNet vs LSTM:
  - superior when there is self similarity
  - similar when there is not
- Re-use of scoring function is crucial
  - allows training to succeed despite non differentiable arg max

Real Music Data

- Pruning of the edit tree does indeed trade speed and accuracy
- With enough time we beat the LSTM
- Proof of concept implementation:
    - scaling up required



(a) test set negative log likelihood



(b) median time per epoch

	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

test set negative log likelihood

Summary / Outlook

Summary / Outlook

- Music involves a family of transformations between subsequences of the same piece
- Local correlation (suffix tree, etc) methods cannot capture this.
- HMM, LSTM, etc. can capture this in theory but not in practice.
- We have a more explicit scheme which can capture it.
    - Suffix tree → **edit tree**
    - Correlation forecast → **analogy forecast**
- To do
    - Faster data structures and algorithms + more data → better machine music
    - Simplified models with more rigour
    - Genetics
    - Language: translation, tense changes *etc.*
    - Non sequence data: images *etc.*

Title:

Neural Dynamic Programming for Musical Self Similarity

Abstract:

https://arxiv.org/abs/1802.03144

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.

Bio:

Christian Walder obtained a Bachelor of Engineering from the University of Queensland, a PhD in machine learning from the Max Planck Institute in Germany, and seven years' industrial experience applying advanced analytics in the finance and telecommunication industries. He is presently employed as a senior researcher at Australia's governmental research, CSIRO Data61, and an adjunct Professor at the Australian National University.