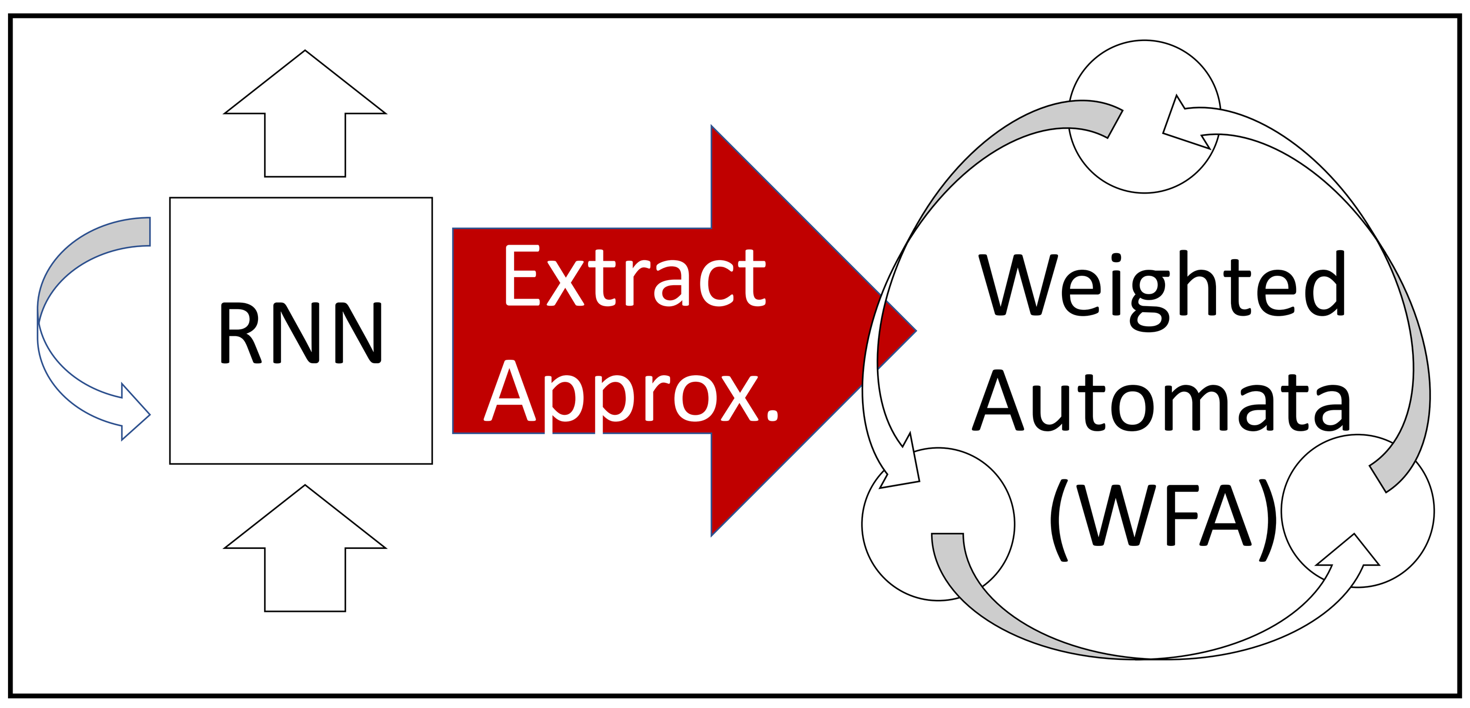
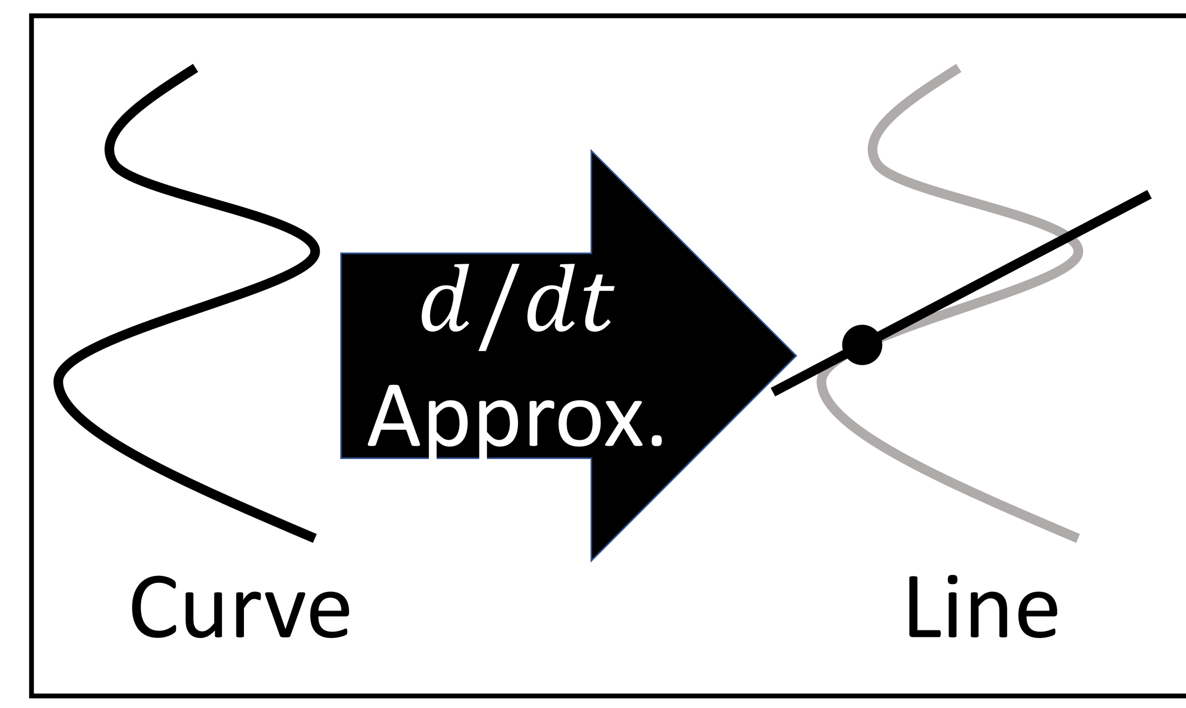
Weighted Automata Extraction from Recurrent Neural Networks via Regression on State Spaces

Takamasa Okudono, Masaki Waga, Taro Sekiyama, Ichiro Hasuo
@ National Institute of Informatics & the Graduate University for Advanced Studies (SOKENDAI), Japan

What's this?

We did 

like we do 

!

Experimental results

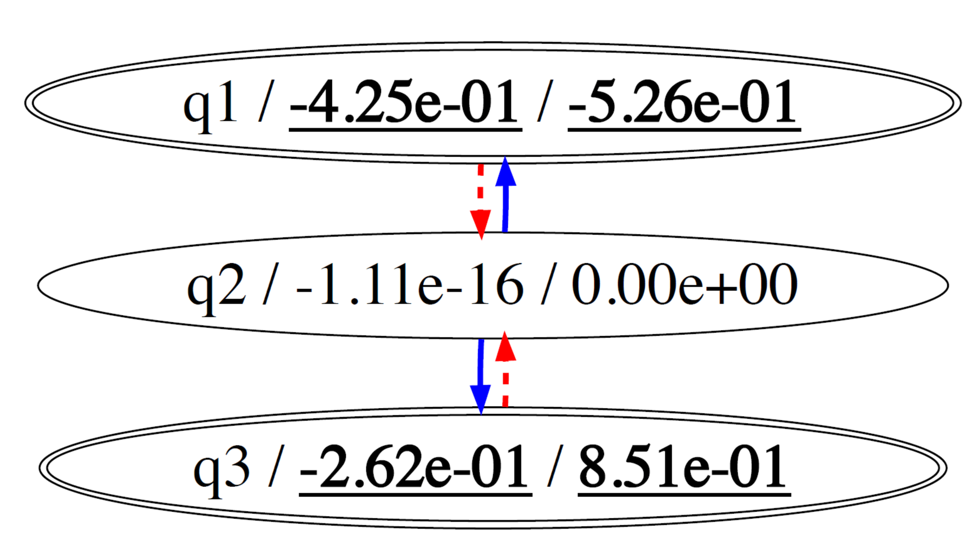
(1) Extraction is quick and accurate!

$(\|\Sigma\|, \|Q_{A\bullet}\|)$	**RGR**(2)	**RGR**(5)	**BFS**(500)	**BFS**(1000)	**BFS**(2000)	**BFS**(3000)	**BFS**(5000)
(4, 10)	2.17 / 286	2.39 / 338	26.8 / 165	9.77 / 279	4.36 / 545	4.07 / 716	2.33 / 1390
(6, 10)	2.45 / 1787	2.54 / 1302	6.99 / 386	4.48 / 641	4.08 / 1218	3.15 / 1410	2.28 / 2480
(10, 10)	4.68 / 7462	4.46 / 5311	22.5 / 928	11.9 / 1562	5.90 / 3521	4.55 / 3638	3.55 / 5571
(10, 15)	5.62 / 8941	5.78 / 8564	21.2 / 2155	10.6 / 4750	7.87 / 5692	5.71 / 7344	5.27 / 7612
(10, 20)	3.70 / 7610	3.79 / 7799	6.24 / 2465	10.1 / 2188	6.13 / 3106	3.70 / 5729	3.63 / 7473
(15, 10)	7.34 / 9569	5.52 / 10000	13.5 / 3227	8.01 / 6765	6.07 / 7916	5.98 / 8911	6.17 / 8979
(15, 15)	8.44 / 10000	5.58 / 9981	16.3 / 2675	9.24 / 4850	7.28 / 5135	9.88 / 7204	6.44 / 8425
(15, 20)	9.16 / 7344	5.15 / 7857	13.7 / 2224	7.26 / 3823	6.60 / 5744	4.96 / 5674	4.01 / 9464
Total	5.45 / 6625	4.40 / 6394	15.9 / 1778	8.92 / 3107	6.04 / 4110	5.25 / 5078	4.21 / 6549

$(\|\Sigma\|, \|Q_{A\bullet}\|)$	**RGR**(2)	**RGR**(5)	**BFS**(500)	**BFS**(1000)	**BFS**(2000)	**BFS**(3000)	**BFS**(5000)
(4, 10)	7.73 / 696	7.07 / 1135	15.0 / 199	7.96 / 424	6.62 / 650	6.61 / 762	9.06 / 1693
(6, 10)	4.92 / 1442	7.43 / 1247	1.46 / 552	6.95 / 660	5.90 / 1217	8.78 / 1557	3.54 / 2237
(10, 10)	5.02 / 5536	4.28 / 5951	7.70 / 1117	11.0 / 1738	4.77 / 2635	3.52 / 3926	4.52 / 4777
(10, 15)	7.15 / 6977	4.35 / 8315	19.4 / 1552	13.8 / 3271	16.8 / 3209	8.57 / 5293	5.08 / 6522
(10, 20)	6.98 / 4697	8.06 / 6704	18.6 / 1465	11.8 / 2046	12.7 / 2851	9.03 / 4259	8.01 / 4856
(15, 10)	5.97 / 8747	6.77 / 8882	23.3 / 2359	11.2 / 4668	9.88 / 6186	6.24 / 7557	6.02 / 8245
(15, 15)	5.78 / 8325	8.71 / 7546	16.6 / 2874	7.31 / 4380	9.92 / 6015	9.89 / 7110	6.40 / 8358
(15, 20)	4.60 / 7652	8.56 / 8334	36.9 / 1893	23.7 / 3069	12.8 / 3987	12.0 / 5262	8.38 / 6441
Total	6.02 / 5510	6.90 / 6015	19.0 / 1502	11.7 / 2532	9.92 / 3344	8.08 / 4466	6.38 / 5391

- We trained RNNs from randomly generated WFAs and applied the extraction to the RNNs. The cells show "{error}/{time}."
- Rows show various settings of the random WFAs / Columns show extraction methods: RGR(*) are ours, and BFS(*) are naïve baselines.
- Two tables use different ways to generate WFAs (see our paper!)

(2) Extraction yields interpretability!



- •We trained RNNs from a weighted language (see "Math stuff") that are so complex that any WFAs cannot express it and applied our extraction.
- The extracted WFAs represent a simplified version of the original language, and it can be graphically representable as above.

(3) Extraction makes the inference faster!

x1300 faster
by using WFA
Instead of RNN

- We used the same RNNs and extracted WFAs and compared inference time for random inputs.

Method

Algorithm 1 Answering equivalence queries

1: **procedure** ANS-EQQ
Input: RNN $R = (\alpha_R, \beta_R, g_R)$, WFA $A = (Q_A, \alpha_A, \beta_A, (A_\sigma)_{\sigma\in\Sigma})$, error tolerance $e > 0$ and concentration threshold $M \in \mathbb{N}$
Output: a counterexample, or Equivalent
2: Initialize $p\colon \mathbb{R}^d \to \mathbb{R}^{Q_A}$ so that $p(\delta_R(\varepsilon)) = \delta_A(\varepsilon)$
3: queue $\leftarrow \langle\varepsilon\rangle$; visited $\leftarrow \emptyset$
4: **while** queue is non-empty **do**
5: $h \leftarrow$ **pop**(queue) ▷ Pop the element of the maximum priority
6: **if** $|f_R(h) - f_A(h)| \geq e$ **then**
7: **return** h ▷ return a counterexample
8: result $\leftarrow$ CONSISTENT?(h, visited, p)
9: **if** result = NG **then**
10: learn p by regression, so that $p(\delta_R(h')) = \delta_A(h')$ holds for all $h' \in$ visited $\cup \{h\}$
11: visited $\leftarrow$ visited $\cup \{ h \}$
12: visited′ $\leftarrow p(\delta_R($visited$))$
13: $\#vn \leftarrow |\{ x \in$ visited′ $\mid x \simeq_A p(\delta_R(h)) \}|$
14: **if** $\#vn \leq M$ **then**
15: pr $\leftarrow \min_{h' \in \text{visited}\setminus\{h\}} d(p(\delta_R(h)), p(\delta_R(h')))$ ▷ d is the Euclidean distance
16: **push** $h\sigma$ **to** queue with priority pr **for** $\sigma \in \Sigma$
17: **return** Equivalent

- Balle and Mohri's WFA learning algorithm reduces the problem of checking whether the target RNN and a given WFA are equivalent in respect of behavior.
- The algorithm is on the left. The main ideas are:
  - Run best-first search on Σ^* and find the difference between the RNN and the WFA (Line 4-7, 16)
  - Find the relation between the RNN and the WFA by making a map from the internal state of RNN to the internal state of WFA by GPR (Line 10)
  - Prioritize the visit where "the search is not enough." (Line 5, 14-15)
  - Prune the searching where "the search is enough." (Line 12-13)

Math stuff…

[Weighted language] Function $\Sigma^* \to \mathbb{R}$
[WFA] WFA A is a quadruple of
- Q_A ——— states
- $\alpha_A \in \mathbb{R}^{Q_A}$ ——— initial vector
- $\beta_A \in \mathbb{R}^{Q_A}$ ——— final vector
- $(A_\sigma)_{\sigma\in\Sigma} \in \mathbb{R}^{Q_A \times Q_A}$ ——— transition matrix

WFA induces its configuration δ_A and output f_A by:
- $\delta_A(w_1 \dots w_n) = \alpha_A^T A_{w_1} \dots A_{w_n}$
- $f_A(w_1 \dots w_n) = \delta_A(w_1 \dots w_n)\beta$

[RNN] RNN R is a triple of
- $\alpha_R \in \mathbb{R}^d$ ——— initial state
- $\beta_R \in \mathbb{R}^d \to \mathbb{R}$ ——— final state
- $g_R\colon \mathbb{R}^d \times \Sigma^* \to \mathbb{R}^d$ ——— transition func.

RNN induces its configuration δ_R and output f_A by:
- $\delta_R(w_1 \dots w_n) = \begin{cases} g_R(\delta_R(w_1 \dots w_{n-1}), w_n) & n > 0 \\ \alpha_A & n = 0 \end{cases}$
- $f_R(w_1 \dots w_n) = \beta_R(\delta_R(w_1 \dots w_n))$

Future work

- Improving scalability
  Currently, $|\Sigma| = 15$ is the maximum. Can we make it work for larger $|\Sigma|$ so that it is applicable for NLP?
- Giving a theoretical guarantee
  It is a heuristics now, and there is no guarantee for the superiority of our method nor termination.
- Applying for quality assurance of RNN
  Model-checking is a field to generate the proof of the (typically) safety of the system. Can we combine our technique with model checking of WFAs and make a technique for the quality assurance of RNN?
- Forcing the WFA to be probabilistic
  Even if the target system outputs the values in [0, 1], Balle and Mohri's algorithm does not necessarily output WFAs whose output is in it. Can we fix it and improve the applicability to the RNNs whose outputs are probabilistic?

This work is partially supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JSPS KAKENHI Grant Numbers JP15KT0012, JP18J22498, JP19K20247, JP19K22842, and JST-Mirai Program Grant Number JPMJMI18BA, Japan.