Efficient Adversarial Sequence Generation for RNN with Symbolic Weighted Finite Automata

Mingjun Ma, Dehui Du, Yuanhao Liu, Yanyun Wang, Yiyang Li

Software Engineering Institute
East China Normal University, Shanghai, China

Presenter: Yuanhao Liu

01/03/2022, Virtual, SafeAI 2022-Workshop@AAAI-22
Outline

• Background
• Preliminaries
• Main approach
• Experiments
• Related work
• Conclusion and discussion
The Gist

- Efficient adversarial sequence generation approach for RNN by SWFA
  - Extract SWFA from RNN with the symbolic extraction algorithm Fast $k$-DCP
  - Perturb the symbolic input to generate adversarial sequences

- Adversarial sequences generated by our approach are more covert
  - Keep perturbation within the human-invisible range

- Implement adversarial sequence generation generation algorithm
  - Outperform the state-of-art attack methods with 112.92% improvement and 1.44 times speedup
Background

Robustness of DNN

Interpretable Models

Data → DNN/RNN → Prediction

Interpretable Model

Adversarial Attacks

White-box attack
FGSM, PGD, etc.

Black-box attack
NewtonFool, etc.

Not applicable for RNN.
The cyclic structure of RNN makes it difficult to craft adversarial examples on sequential data.

Interpretable Models

Derministic Finite Automata (DFA)
Probalistic Finite Automata (PFA)
Weighted Finite Automata (WFA)
Symbolic Weighted Finite Automata (SWFA)
Preliminaries

➢ Recurrent Neural Network
   • RNN is denoted as a 6-tuple $R = (H, X, Y, h_0, f, g)$.

➢ Symbolic Weighted Finite Automata
   • As well as WFA, SWFA can perform real-value operations
   • SWFA is denoted as a 5-tuple $\Upsilon = (G, Q, \alpha, \beta, A)$. 
Symbolic Weighted Finite Automata

- Transition edges are labelled by *functions*
- Enhance the *abstraction ability* of WFA
- Can deal with a *possibly infinite alphabet* efficiently
- *First* use SWFA for perturbation
Main Approach

- Symbolic Weighted Finite Automata Extraction
  - Abstract the symbolic input
  - Abstract the SWFA from RNN

- Adversarial Sequences Generation by SWFA
  - Gain the symbolic input sequences
  - Screen out the symbolic adversarial sequences
Symbolic Weighted Finite Automata Extraction

- The \( k\)-DCP captures the top \( k \) ranked class labels as well as their prediction confidence levels.
- **High Efficiency**: Discarding the time-consuming \( k \)-means clustering and establishing symbolic blocks directly.
- **Symbolic Abstraction**: Extending to the infinite alphabet, which deals with input symbolically.

From Du et al. 2019 directly
- From Du et al. 2019 and improved
- Newly proposed in our approach
Fast $k$-DCP
(Our New Contribution)

- Time complexity: $O(mns)$
- Space complexity: $O(T^S)$
- Suitable for large-scale tasks

Du et al. 2019:

```
Algorithm 1: RNN-SWFA by Fast $k$-DCP

input: RNN $R = (H, X, Y, h_0, f, g)$
Input sequences $W$
$K, T$

output: SWFA $\mathcal{Y} = (G, Q, \alpha, \beta, A)$

1. Initialize $Q' = [s_0], \Sigma' = [], Q = [], \Sigma = []$;
2. Initialize $A = [], \alpha = \pi q_0, \beta = []$;
3. $d = \text{ComputeDistance}(W)$;
4. $T_{input} = \left\lceil 10/(d)/(|W| \times |w|) \right\rceil$;
5. for $w \in W$ do
   6. $s = [h^{(i)}(w)]_{i=0}^{|w|}$;
   7. for $i = 1$ to $w$ do
      8. $Q').add(s_i)$;
      9. $\Sigma').add(w_{i-1})$
   10. for $q' \in Q'$ do
      11. $Q).add(fkdcp^{K,T}(q'))$;
      12. for $\sigma' \in \Sigma'$ do
          13. $\Sigma).add(fkdcp^{\sigma'|T_{input}}(\sigma'))$;
      14. for $\sigma \in \Sigma$ do
          15. $A_{\sigma} = \text{BuildTransitionMatrix}(\sigma)$;
          16. $A).add(A_{\sigma})$;
      17. for $q \in Q$ do
          18. $\beta_q = 0$ with length $|L|$;
          19. for $q' \in Q'$ do
              20. if $f\text{kdcp}^{K,T}(q') == q$ then
    21. $\beta_q[\text{argmax}(g(q'))] + 1$;
    22. $\beta_q = \beta_q / \sum(\beta_q)$;
    23. $\beta_q.add(\beta_q)$;
      24. $G = \text{GuardFunctionLearning}(Q, \alpha, \beta, A)$;
      25. return SWFA $\mathcal{Y} = (G, Q, \alpha, \beta, A)$
```
Adversarial Sequence Generation

Step 1: Set an appropriate $T_{in}$

Step 2: Abstract input space into blocks
- Divided by Fast $k$-DCP
- An interval [0,1] can be divided into $3 \times 3 \times 3$ blocks

Step 3: Find Direction and Perturbation

Step 4: Check “000-status”
- Represent the input exceeds the generalization limit of our model (omitting details, cf. the paper)
Experiment Setting

Public Datasets:

- NGSIM
  - Next Generation Simulation (NGSIM) program collected detailed vehicle trajectory data on southbound US 101 through a network of synchronized digital video cameras.

- UCR time-series datasets
  - Introduced in 2002, open source time-series data, with at least one thousand making use of these datasets.
Experiment Setting

Our Datasets:

- **ADD (Proposed by this paper)**

  - Autonomous Driving Datasets Generated by Carla (Zhang et al. 2021)

<table>
<thead>
<tr>
<th>Serial number</th>
<th>Time step</th>
<th>Longitude Coordinate</th>
<th>Latitude Coordinate</th>
<th>Vehicle Width(m)</th>
<th>Head turn</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-41.2</td>
<td>50.2</td>
<td>2</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-43.1</td>
<td>50.5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>-44.5</td>
<td>50.6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-48.6</td>
<td>50.4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-48.6</td>
<td>50.4</td>
<td>2.5</td>
<td>Right</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-44.5</td>
<td>50.1</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>-43.1</td>
<td>49.2</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>-41.2</td>
<td>49.8</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>1</td>
<td>100.2</td>
<td>1.2</td>
<td>2.5</td>
<td>Straight</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>101.9</td>
<td>2.3</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>100.1</td>
<td>2.1</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>99.5</td>
<td>2.0</td>
<td>2.5</td>
<td></td>
</tr>
</tbody>
</table>

*Car is turning left*  

*Our data structure*
Experiment I: \textit{RNN-SWFA Extraction}

Table 1: Comparison between SWFAs extracted by Fast $k$-DCP on various time-series data

<table>
<thead>
<tr>
<th>Datasets</th>
<th>AoR(%)</th>
<th>AoS(%)</th>
<th>ET(s)</th>
<th>RT(s)</th>
<th>ST(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>99/97</td>
<td>89/79</td>
<td>421.536</td>
<td>4.982</td>
<td>3.214</td>
</tr>
<tr>
<td>NGSIM</td>
<td>91/86</td>
<td>77/73</td>
<td>28.704</td>
<td>3.016</td>
<td>2.971</td>
</tr>
<tr>
<td>PPOAG</td>
<td>75/88</td>
<td>35/43</td>
<td>2.667</td>
<td>0.224</td>
<td>0.443</td>
</tr>
<tr>
<td>CT</td>
<td>53/74</td>
<td>53/73</td>
<td>0.026</td>
<td>0.005</td>
<td>0.004</td>
</tr>
<tr>
<td>EQ</td>
<td>82/75</td>
<td>82/76</td>
<td>7.229</td>
<td>1.112</td>
<td>1.194</td>
</tr>
</tbody>
</table>

\. AoR: Accuracy of RNN (training/test)
\. AoS: Accuracy of SWFA (training/test)
\. ET: Extraction Time of SWFA
\. RT: Running Time of RNN
\. ST: Running Time of SWFA

- RNN\textsubscript{Acc} $\approx$ SWFA\textsubscript{Acc}
- RNN\textsubscript{RunningTime} $\approx$ SWFA\textsubscript{RunningTime}

- Reuse the time-consuming extraction
- Work in \textit{infinite alphabets}
### Experiment II: SWFA-based adversarial sequence generation

Table 2: Comparison between abstraction-based adversarial sequence generation approach and other adversarial attacking algorithms on the autonomous driving dataset

<table>
<thead>
<tr>
<th>Category</th>
<th>White Box</th>
<th>Black Box</th>
<th>Our Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>FGSM</td>
<td>PGD</td>
<td>NewtonFool</td>
</tr>
<tr>
<td>Perturbation(δ)</td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>ASR(%)</td>
<td>0.00</td>
<td>0.33</td>
<td>21.66</td>
</tr>
<tr>
<td>Time(s)</td>
<td>-</td>
<td>3.15</td>
<td>10.00</td>
</tr>
</tbody>
</table>

**Our Approach**
- Outstanding success rate: 81.11%, 94.56% and 112.92%
- Fast Speed: 1.44 times
Related Work

- More efficient in generating adversarial sequences
- With more subtle perturbations
- Take advantage of the real-value operation ability of WFA to simulate RNN.
- Use the symbolic characteristics of SWFA, which enhances generalization.

Reference:


Conclusion

Main Contribution:

- The novel \textit{Fast k-DCP} symbolic extraction algorithm
- \textit{Efficient adversarial sequence generation} by SWFA

Main Advantage:

- Applicable to generate \textit{covert} adversarial sequences
- Perturbation within \textit{human-invisible range}
- Suitable for \textit{Spatio-temporal} sequential tasks
Discussion

Drawbacks:

➤ Not yet adapting to large-class sequential data
➤ Should study on various datasets.

Future work:

➤ Further optimize our approach.
➤ Investigate the reachability analysis of SWFA.
➤ Explore more valuable properties of SWFA for improving efficiency.
Thank you for your attention

Questions? (dhdu@sei.ecnu.edu.cn)

- Acknowledgements: Financial support for this work, provided by the National Natural Science Foundation of China under Grant No.61972153, the Key projects of the Ministry of Science and Technology under No. 2020AAA0107800, is gratefully acknowledged.