Learning Moore Machines from Input-Output Traces

Georgios Giantamidis\textsuperscript{1} and Stavros Tripakis\textsuperscript{1,2}

\textsuperscript{1}Aalto University, Finland
\textsuperscript{2}UC Berkeley, USA
Motivation: learning models from black boxes

Many applications:
- Verify that a black-box component is safe to use
- Dynamic malware analysis
- ...
Learning FSMs from input-output traces

IO-traces

- \(aa\) \(\mapsto\) 020
- \(bba\) \(\mapsto\) 0122
- \(abaa\) \(\mapsto\) 02220
- \(abba\) \(\mapsto\) 02220

Learned FSM

- States: \(q_0, q_1, q_2, q_3\)
- Transitions:
  - \(a\) from \(q_0\) to \(q_2\)
  - \(a\) from \(q_2\) to \(q_0\)
  - \(b\) from \(q_1\) to \(q_3\)
  - \(b\) from \(q_3\) to \(q_1\)

- Initial state: \(q_0\)
- Final states: \(q_1, q_3\)
Outline

1 Background
2 Formal problem definition
3 Related work
4 Identification in the limit
5 Our learning algorithms
6 Results
7 Summary & future work
Moore machines

- input alphabet, $I = \{a, b\}$
- output alphabet, $O = \{0, 1, 2\}$
- set of states, $Q = \{q_0, q_1, q_2, q_3\}$
- initial state, $q_0$
- transition function, $\delta : Q \times I \rightarrow Q$
- output function, $\lambda : Q \rightarrow O$

$(I, O, Q, q_0, \delta, \lambda)$
Moore machines

By definition, our machines are **deterministic** and **complete**.
Input-output traces

Moore machine

Some I/O traces generated by the machine

- $aa \rightarrow 020$
- $baa \rightarrow 0122$
- $bba \rightarrow 0122$
- $abaa \rightarrow 02220$
- $abba \rightarrow 02220$
Consistency

This machine is consistent with this set of traces.

\[ aa \mapsto 020 \\
baa \mapsto 0122 \\
bbba \mapsto 0122 \\
abaa \mapsto 02220 \\
abba \mapsto 02220 \]
This machine is inconsistent with this set of traces.

\begin{align*}
    aa & \mapsto 020 \\
    baa & \mapsto 0122 \\
    bba & \mapsto 0122 \\
    abaa & \mapsto 02220 \\
    abba & \mapsto 02220
\end{align*}
Outline

1. Background

2. Formal problem definition

3. Related work

4. Identification in the limit

5. Our learning algorithms

6. Results

7. Summary & future work
A first attempt at problem definition

Given ...
- Input alphabet, $I$
- Output alphabet, $O$
- Set of IO-traces, $S$ (the training set)

... find a Moore machine $M$ such that:
- $M$ is deterministic
- $M$ is complete
- $M$ is consistent with $S$
A trivial solution

\[
\begin{align*}
b & \mapsto 01 \\
aa & \mapsto 020 \\
ab & \mapsto 022
\end{align*}
\]

This is called the prefix-tree machine.

Not quite a solution: machine incomplete...
A trivial solution

\[ b \mapsto 01 \]
\[ aa \mapsto 020 \]
\[ ab \mapsto 022 \]

This is called the prefix-tree machine.
A trivial solution

\[ b \mapsto 01 \]
\[ aa \mapsto 020 \]
\[ ab \mapsto 022 \]

This is called the **prefix-tree machine**.
Not quite a solution: machine incomplete ...
A trivial solution

\[ b \mapsto 01 \]
\[ aa \mapsto 020 \]
\[ ab \mapsto 022 \]

... but easily completed with self-loops.
Problems with the trivial solution

(1) **Poor generalization**, due to trivial completion with self-loops
   - The machine may be consistent with the *training* set ...
   - ... but how *accurate* is it on a *test* set?
Problems with the trivial solution

1. **Poor generalization**, due to trivial completion with self-loops
   - The machine may be consistent with the *training* set ...
   - ... but how *accurate* is it on a *test* set?

2. Large number of states in the learned machine
   - The prefix-tree machine does not merge states at all.
Revised problem definition

The LMoMIO problem (Learning Moore Machines Input-Output Traces):

Given ...

- Input alphabet, $I$
- Output alphabet, $O$
- Set of IO-traces, $S$ (the training set)

... find a Moore machine $M$ such that:

- $M$ is deterministic
- $M$ is complete
- $M$ is consistent with $S$

... and also:

- $M$ generalizes well (good accuracy on a-priori unknown test sets)
- $M$ is small (few states)
- $M$ is found quickly (good learning algorithm complexity)
How to measure “accuracy”?  

We define three metrics: **Strong, Medium, Weak**

<table>
<thead>
<tr>
<th>test trace</th>
<th>machine output</th>
<th>strong acc.</th>
<th>medium acc.</th>
<th>weak acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc $\mapsto$ 1234</td>
<td>1234</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>abc $\mapsto$ 1234</td>
<td>4321</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>abc $\mapsto$ 1234</td>
<td>1212</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>abc $\mapsto$ 1234</td>
<td>3434</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>abc $\mapsto$ 1234</td>
<td>1324</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{2}$</td>
</tr>
</tbody>
</table>
Outline

1. Background
2. Formal problem definition
3. Related work
4. Identification in the limit
5. Our learning algorithms
6. Results
7. Summary & future work
Related work

- A* [Angluin, 1987]
- NP-hard [Gold, 1978]
- K-tails [Biermann & Feldman, 1972]
- Gold's algorithm [Gold, 1978]
- RPNI [Oncina & Garcia, 1992]
- Genetic algorithms
- Ant colony optimization
- Our work

- Active
- Passive
- Exact
- Heuristic
Concept introduced in [Gold, 1967], in the context of formal language learning

- Learning is seen as an infinite process
- Training set keeps growing: $S_0 \subseteq S_1 \subseteq S_2 \subseteq \cdots$
- Every input word is guaranteed to eventually appear in the training set
- For each $S_i$, the learner outputs machine $M_i$
- Identification in the limit := learner outputs the right machine after some $i$
Identification in the limit

Concept introduced in [Gold, 1967], in the context of formal language learning

- Learning is seen as an infinite process
- Training set keeps growing: $S_0 \subseteq S_1 \subseteq S_2 \subseteq \cdots$
- Every input word is guaranteed to eventually appear in the training set
- For each $S_i$, the learner outputs machine $M_i$
- Identification in the limit $\equiv$ learner outputs the right machine after some $i$

A good passive learning algorithm must identify in the limit.
To prove identification in the limit, we use the notion of the **Characteristic Sample** [C. de la Higuera, 2010]:

- Concept existing for DFAs (deterministic finite automata) – we adapt it to Moore machines
- Intuition: set of IO-traces that “covers” the machine (covers all states, all transitions)
- For a minimal Moore machine $M = (I, O, Q, q_0, δ, λ)$, there exists a CS of total length $O(|Q|^4|I|)$
Characteristic samples

To prove identification in the limit, we use the notion of the **Characteristic Sample** [C. de la Higuera, 2010]:

- Concept existing for DFAs (deterministic finite automata) – we adapt it to Moore machines
- Intuition: set of IO-traces that “covers” the machine (covers all states, all transitions)
- For a minimal Moore machine $M = (I, O, Q, q_0, \delta, \lambda)$, there exists a CS of total length $O(|Q|^4|I|)$

**Characteristic Sample Requirement** (CSR):

- A learning algorithm satisfies CSR if it satisfies the following:
  
  *If the training set $S$ is a characteristic sample of a minimal machine $M$, then the algorithm learns from $S$ a machine isomorphic to $M$.*

- CSR can be shown to imply identification in the limit
Outline

1 Background

2 Formal problem definition

3 Related work

4 Identification in the limit

5 Our learning algorithms

6 Results

7 Summary & future work
Three learning algorithms

- PTAP - Prefix Tree Acceptor Product
- PRPNI - Product RPNI
- MooreMI - Moore Machine Inference
This is the trivial solution we discussed earlier:

\[
\begin{align*}
  b & \mapsto 01 \\
  aa & \mapsto 020 \\
  ab & \mapsto 022
\end{align*}
\]

Drawbacks:

- Large number of states in learned machine
- Poor generalization / accuracy
Observations:
- A DFA is a special case of a Moore machine with binary output (accept/reject)
- A Moore machine can be encoded as a product of $\lceil \log_2 |O| \rceil$ DFAs

Based on these observations, PRPNI works as follows:
- Uses the RPNI algorithm [J. Oncina and P. Garcia, 1992], which learns DFAs
- Learns several DFAs that encode the learned Moore machine
- Computes product of the learned DFAs and completes it

Drawbacks:
- DFAs are learned separately, therefore do not have same state-transition structure $\Rightarrow$ state explosion during product computation
Observations:
- A DFA is a special case of a Moore machine with binary output (accept/reject)
- A Moore machine can be encoded as a product of $\lceil \log_2 |O| \rceil$ DFAs

Based on these observations, PRPNI works as follows:
- Uses the RPNI algorithm [J. Oncina and P. Garcia, 1992], which learns DFAs
- Learns several DFAs that encode the learned Moore machine
- Computes product of the learned DFAs and completes it

Drawbacks:
- DFAs are learned separately, therefore do not have same state-transition structure $\implies$ state explosion during product computation
- Invalid output codes
Invalid output codes

Output alphabet: $O = \{0, 1, 2\}$

Binary encoding of $O$: $f = \{0 \mapsto 00, 1 \mapsto 01, 2 \mapsto 10\}$

Invalid output code: **11** does not correspond to any output symbol
MooreMI - Moore Machine Inference

- Modified RPNI, tailored to Moore machine learning
- Like PRPNI, learns several DFAs that encode the learned Moore machine
- Unlike PRPNI, learned DFAs maintain same state-transition structure
- Therefore, no state explosion during product computation
- No invalid output codes either
Outline

1. Background
2. Formal problem definition
3. Related work
4. Identification in the limit
5. Our learning algorithms
6. Results
7. Summary & future work
Theorem 1
All three algorithms return Moore machines consistent with the IO-traces received as input.

Theorem 2
The MooreMI algorithm satisfies the characteristic sample requirement and identifies in the limit.

Experimental evaluation result:
MooreMI is better not just in theory, but also in practice
Outline

1 Background

2 Formal problem definition

3 Related work

4 Identification in the limit

5 Our learning algorithms

6 Results

7 Summary & future work
Summary

- Learning deterministic, complete Moore machines from input-output traces
- Characteristic sample for Moore machines
- Three algorithms to solve the problem
- MooreMI algorithm identifies in the limit
Future work

- Extend to Mealy machines
- Learning symbolic machines
- Learning from traces and formal requirements (e.g. LTL formulas)
- Industrial case studies
Future work

- Extend to Mealy machines
- Learning symbolic machines
- Learning from traces and formal requirements (e.g. LTL formulas)
- Industrial case studies

Thank you! Questions?


