Learning Regular Languages using queries

Thessaloniki, Greece
February 22, 2018
Learning Regular Languages
Automata Learning
Learning Automata over Large Alphabets
Outline

What is Learning?

Regular Languages

Automata Learning

Learning Automata over Large Alphabets
Outline

What is Learning?
   Definition
   Types of Learning
   Timeline of Automata Learning

Regular Languages

Automata Learning

Learning Automata over Large Alphabets
Machine Learning

Classification	Regression	Neural Network

- What do we know about the underlying model?
- What do we need the model for?
- How do we retrieve information?
Machine Learning

a small sample

\[ M = \{(x, y) : x \in X, \ y \in Y\} \]

Learning Regular Languages

- \( \Sigma \) an alphabet
- \( X = \Sigma^* \) set of words
- \( Y = \{+, -\} \)

The model is an automaton
Types of Learning

**Off-line vs Online**

The sample $M$ is known before the learning procedure starts.  
The sample $M$ is updated during learning.

**Passive vs Active**

The sample $M$ is given.  
The sample $M$ is chosen by the learning algorithm.

**Learning using Queries**

The learning algorithm can access queries e.g., membership queries, equivalence queries, etc.
A Short Prehistory and History of Automaton Learning

1956
Edward F Moore. *Gedanken-experiments on sequential machines.*
Defines the problem as a black box model inference.

1967
E. Mark Gold. *Language identification in the limit.*

1972
Learning finite automata is possible in finite time. He first uses the basic idea that underlies table-based methods.

1978
E. Mark Gold. *Complexity of automaton identification from given data.*
Finding the minimal automaton compatible with a given sample is NP-hard.

1987
Dana Angluin. *Learning regular sets from queries and counter-examples.*
The $L^*$ active learning algorithm with membership and equivalence queries. Polynomial in the automaton size.

1993
An improved version of the $L^*$ algorithm using the breakpoint method to treat counter-examples.
Outline

What is Learning?

Regular Languages
  Automata, Trees, and Tables
  Nerode’s Theorem and Canonical Representation

Automata Learning

Learning Automata over Large Alphabets
Regular Languages and Automata

$L \subseteq \Sigma^*$ is a language

Equivalence relation

$u \sim_L v$ iff $u \cdot w \in L \iff v \cdot w \in L$

Nerode’s Theorem

$L$ is a regular language iff $\sim_L$ has finitely many equivalence classes.

$Q = \Sigma^* / \sim$ (states in the minimal representation of $L$.)

$\varepsilon \sim b \sim aa \quad a \sim ba \sim abb \quad ab \sim aba$
Regular Languages and Automata

A sufficient sample that characterizes the language

<table>
<thead>
<tr>
<th>$S$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>$a$</td>
<td>$-$</td>
</tr>
<tr>
<td>$ab$</td>
<td>$+$</td>
</tr>
<tr>
<td>$b$</td>
<td>$-$</td>
</tr>
<tr>
<td>$aa$</td>
<td>$-$</td>
</tr>
<tr>
<td>$aba$</td>
<td>$+$</td>
</tr>
<tr>
<td>$abb$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

$S$  prefixes (states)
$R$  boundary ($R = S \cdot \Sigma \setminus S$)
$E$  suffixes (distinguishing strings)

$f : S \cup R \times E \to \{+, -\}$ classif. function
$f_s : E \to \{+, -\}$ residual functions

$\mathcal{A}_L = (\Sigma, Q, q_0, \delta, F)$
- $Q = S$
- $q_0 = [\varepsilon]$
- $\delta([u], a) = [u \cdot a]$
- $F = \{[u] : (u \cdot \varepsilon) \in L\}$

The minimal automaton for $L$
The Observation Table $T$

From a *closed* and *consistent* table $T$, one can construct a dfa that is compatible with it.

$$A_T = (\Sigma, S, \epsilon, F, \delta)$$

Every reduced table is consistent.

- $T$ *closed* if $\forall r \in R, \exists s \in S, f_r = f_s$
- $T$ *consistent* if $\forall s, s' \in S, \forall a \in \Sigma, f_s = f_{s'} \Rightarrow f_s \cdot a = f_{s'} \cdot a$
- $T$ *reduced* when $\forall s, s' \in S, f_s \neq f_{s'}$
Outline

What is Learning?

Regular Languages

Automata Learning
- The $L^*$ Algorithm
- Observation Tables
- Counter-examples
- An Example
- Other Automata Learning

Learning Automata over Large Alphabets
The $L^*$ Algorithmic Scheme*

Active learning using queries

---

Counter-Examples

Let $w = a_1 \cdot a_2 \cdots a_n$ be a counter-example

Angluin’s Counter-Example Treatment [Angluin’87]
- Add all prefixes $a_1 \cdots a_i$ to the set of prefixes $S$

Maler’s Counter-Example Treatment [Maler’95]
- Add all suffixes $a_i \cdots a_n$ to the set of suffixes $E$

Breakpoint Method [Rivest Shapire’93]
- Find suitable suffix $v_i$ to add to the set of suffixes $E$

What is the error?

All $w \in L \oplus L(H)$ are counter-examples
Example of $L^*$

$\Sigma = \{a, b\}$

**Observation Table**

<table>
<thead>
<tr>
<th>Observation</th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$b$</th>
<th>$aa$</th>
<th>$ab$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>$-$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a$</td>
<td>$+$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>$-$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$aa$</td>
<td>$+$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$ab$</td>
<td>$+$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Hypothesis Automaton**

Counterexample: $-ba$
Example of $L^*$

$\Sigma = \{a, b\}$
Counter-example Treatment (Breakpoint Method)

Let $w = a_1 \cdots a_i \cdots a_{|w|} = u_i \cdot a_i \cdot v_i$ be a counter-example.

$$f(s_{i-1} \cdot a_i \cdot v_i) \neq f(s_i \cdot v_i)$$

$s_i = \delta(\varepsilon, u_i \cdot a_i)$

vertical expansion

$s \cdot a_i$ is a new state
Counter-example Treatment (Breakpoint Method)

**Proposition**

If $w$ is a counter-example to $A_T$ then there exists an $i$-factorization of $w$, i.e., $w = u_i \cdot a_i \cdot v_i$, such that either

$$f(s_{i-1} \cdot a_i \cdot v_i) \neq f(s_i \cdot v_i)$$

(1)

- If (1), then $v_i$ is a new distinguishing word
  - Table not closed $\rightarrow$ new state
Example of $L^*$ breakpoint method

\[ \Sigma = \{a, b\} \]

- **Observation Table**

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$b$</th>
<th>$aa$</th>
<th>$ab$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>$-$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a$</td>
<td></td>
<td>$+$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td></td>
<td></td>
<td>$-$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$aa$</td>
<td></td>
<td></td>
<td></td>
<td>$-$</td>
<td></td>
</tr>
<tr>
<td>$ab$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$+$</td>
</tr>
</tbody>
</table>

- **Hypothesis Automaton**

Ask Equivalence Query:

- counterexample: $-ba$

$a \not\sim ba \rightarrow a$ is a new distinguishing string
Example of $L^*$ breakpoint method

$\Sigma = \{a, b\}$

### Observation Table

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>$-$</td>
<td>$+$</td>
</tr>
<tr>
<td>$a$</td>
<td>$+$</td>
<td>$-$</td>
</tr>
<tr>
<td>$b$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$aa$</td>
<td>$-$</td>
<td>$+$</td>
</tr>
<tr>
<td>$ab$</td>
<td>$+$</td>
<td>$-$</td>
</tr>
<tr>
<td>$ba$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>$bb$</td>
<td>$+$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

### Hypothesis Automaton

Ask Equivalence Query:

True
Other Automata Learning

- Mealy Machines (R Groz 2009)
- Register Automata (M Isberner, F Howar, B Steffen 2012)
- Timed Automata (Verver 2010, Grinchtein 2008)
- ω-languages (Maler 1995, D Angluin, D Fisman 2016)
- Non-deterministic Automata (P García - 2008)
- Probabilistic Automata (ALERGIA Algorithm for passive learning)
- Grammars (C.de la Higuera. Grammatical inference - Book)
- Large Alphabets (Mens, Maler, Steffen, Isberner)
Outline

What is Learning?

Regular Languages

Automata Learning

Learning Automata over Large Alphabets
  Why Large Alphabets?
  Symbolic Automata
  Learning Algorithm for Symbolic Automata
  Experimental Results
Languages over Large Alphabets

Input:

\[ x_1 : 10101010000100 \cdots \]
\[ x_2 : 10100100100100 \cdots \]
\[ x_3 : 1010100010001 \cdots \]
\[ x_4 : 10101000100100 \cdots \]

Boolean Vectors \((\mathbb{B}^n)\)

Time Series \(\subseteq \mathbb{R}\)
Symbolic Automata

\[ \mathcal{A} = (\Sigma, \Sigma, \psi, Q, \delta, q_0, F) \]

- \( Q \) finite set of states,
- \( q_0 \) initial state,
- \( F \) accepting states,
- \( \Sigma \) large concrete alphabet,
- \( \delta \subseteq Q \times \Sigma \times Q \)
- \( \Sigma \) finite alphabet (symbols)
- \( \psi_q : \Sigma \rightarrow \Sigma_q, q \in Q \)
- \( \llbracket a \rrbracket = \{ a \in \Sigma \mid \psi(a) = a \} \)

\( \Sigma \subseteq \mathbb{R} \)

\[ \llbracket a_{01} \rrbracket = \{ x \in \Sigma : x < 50 \} \]

\( w = 20 \cdot 40 \cdot 60, + \)

\( w = a_{01} \cdot a_{12} \cdot a_{41} \)

\( \mathcal{A} \) is complete and deterministic if \( \forall q \in Q \)

\( \{ \llbracket a \rrbracket \mid a \in \Sigma_q \} \) forms a partition of \( \Sigma \).
Learning over Large Alphabets

Why $L^*$ cannot be applied?

- The learner asks MQ’s for all continuations of a state ($\forall a \in \Sigma$, ask MQ($u \cdot a$))
- Inefficient for large finite alphabets
- Not applicable to infinite alphabets

Our solution:

- Use a finite sample of evidences to learn the transitions
- Form evidence compatible partitions
- Associate a symbol to each partition block
- Each symbol has one representative evidence
- The prefixes are symbolic

Evidences:

- $\mu(a) = \{a^1, a^2\}$
- $\hat{\mu}(a) = a^1$
Symbolic Learning Algorithm

Learner

- Initialize
- Fill in Table partially
- Make Hypothesis $H$
- Treat cex

$\Sigma_\varepsilon = \{ a_1, a_2 \}$

Repeat for each new state $q$:

- Sample evidences
- Ask MQ’s
- Learn partitions
- Define the symbolic alphabet $\Sigma_q$
- Select representative $\hat{\mu}(a), \forall a \in \Sigma_q$
Evidence Compatibility

A state $u$ is evidence compatible when

$$f_u \cdot a = f_u \cdot \hat{\mu}(a)$$

for every evidence $a \in [a]$

Evidence incompatibility at state $u$

<table>
<thead>
<tr>
<th>$u \cdot \hat{\mu}(a)$</th>
<th>$u \cdot a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\cdot \cdot \cdot + \cdot \cdot \cdot$</td>
<td>$\cdot \cdot \cdot - \cdot \cdot \cdot$</td>
</tr>
</tbody>
</table>
Counter-example Treatment (Symbolic Breakpoint)

Let \( w = a_1 \cdots a_i \cdots a_{|w|} = u_i \cdot a_i \cdot v_i \) be a counter-example.

\[
f(\hat{\mu}(s_{i-1} \cdot a_i) \cdot v_i) \neq f(\hat{\mu}(s_i) \cdot v_i) = f(\hat{\mu}(s_{i-1}) \cdot a_i \cdot v_i) \neq f(\hat{\mu}(s_{i-1}) \cdot \hat{\mu}(a_i) \cdot v_i)
\]

\( s_i = \delta(\epsilon, u_i \cdot a_i) \)
Example over the alphabet $\Sigma = [1, 100)$

observation table

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>$-$</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_1$</td>
<td>$+$</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_2$</td>
<td>$-$</td>
<td></td>
</tr>
<tr>
<td>13 18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_1a_3$</td>
<td>$-$</td>
<td></td>
</tr>
</tbody>
</table>

semantics

- $\Sigma_{\varepsilon} = \{a_1, a_2\}$
- $\hat{\mu}(a_1)$
- $\hat{\mu}(a_2)$
- $\Sigma_{a_1} = \{a_3\}$
- $\hat{\mu}(a_3)$

hypothesis automaton

- $\varepsilon$
- $x < 27$
- $x \geq 27$
- $w = 35 \cdot 52 \cdot 11$, $-$

Ask Equivalence Query:

counter-example:

add distinguishing string 11

discover new state (vertical expansion)
**Example over the alphabet $\Sigma = [1, 100]$**

**Observation table**

<table>
<thead>
<tr>
<th></th>
<th>$\varepsilon$</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>13 $a_1$</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>68 $a_2$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13 18 $a_1a_3$</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>13 73 $a_1a_6$</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>68 17 $a_2a_4$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>68 75 $a_2a_5$</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

**Semantics**

- $\Sigma_{\varepsilon} = \{a_1, a_2\}$
- $\Sigma_{a_1} = \{a_3, a_6\}$
- $\Sigma_{a_2} = \{a_4, a_5\}$

**Hypothesis automaton**

- $\varepsilon$
- $x < 27$ to $a_1$
- $x \geq 43$ to $a_2$
- $x < 43$ to $a_2$

Ask Equivalence Query:

**Counter-example:**

$w = 12 \cdot 73 \cdot 4, -$ 

- Add 73 as evidence of $a_1$
- Add new transition (horizontal expansion)
Example over the alphabet $\Sigma = [1, 100)$

<table>
<thead>
<tr>
<th>observation table</th>
<th>semantics</th>
<th>hypothesis automaton</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>$\Sigma_\varepsilon = {a_1, a_2}$</td>
<td>$\varepsilon$</td>
</tr>
<tr>
<td>$a_1$</td>
<td>$\Sigma_{a_1} = {a_3, a_6}$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$\Sigma_{a_2} = {a_4, a_5}$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$a_1a_3$</td>
<td>$\mu(a_1)$</td>
<td>$x &lt; 27$</td>
</tr>
<tr>
<td>$a_1a_6$</td>
<td>$\mu(a_2)$</td>
<td>$x \geq 63$</td>
</tr>
<tr>
<td>$a_2a_4$</td>
<td>$\mu(a_3)$</td>
<td>$x \geq 52$</td>
</tr>
<tr>
<td>$a_2a_5$</td>
<td>$\mu(a_4)$</td>
<td>$x &lt; 52$</td>
</tr>
<tr>
<td>$a_2a_5$</td>
<td>$\mu(a_5)$</td>
<td>$x \geq 27$</td>
</tr>
</tbody>
</table>

Ask Equivalence Query: True

return current hypothesis
Empirical Results

Valid passwords over the ASCII characters

<table>
<thead>
<tr>
<th></th>
<th>NUL</th>
<th>DLE</th>
<th>SPC</th>
<th>0</th>
<th>@</th>
<th>96</th>
<th>`</th>
<th>112</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOH</td>
<td>DC1</td>
<td>33</td>
<td>49</td>
<td>65</td>
<td>A</td>
<td>97</td>
<td>a</td>
<td>113</td>
</tr>
<tr>
<td>2</td>
<td>STX</td>
<td>DC2</td>
<td>34</td>
<td>50</td>
<td>66</td>
<td>B</td>
<td>98</td>
<td>b</td>
<td>114</td>
</tr>
<tr>
<td>3</td>
<td>ETX</td>
<td>DC3</td>
<td>35</td>
<td>51</td>
<td>67</td>
<td>C</td>
<td>99</td>
<td>c</td>
<td>115</td>
</tr>
<tr>
<td>4</td>
<td>EOT</td>
<td>DC4</td>
<td>36</td>
<td>52</td>
<td>68</td>
<td>D</td>
<td>100</td>
<td>d</td>
<td>116</td>
</tr>
<tr>
<td>5</td>
<td>ENQ</td>
<td>NAK</td>
<td>37</td>
<td>53</td>
<td>69</td>
<td>E</td>
<td>101</td>
<td>e</td>
<td>117</td>
</tr>
<tr>
<td>6</td>
<td>ACK</td>
<td>SYN</td>
<td>38</td>
<td>54</td>
<td>70</td>
<td>F</td>
<td>102</td>
<td>f</td>
<td>118</td>
</tr>
<tr>
<td>7</td>
<td>BEL</td>
<td>ETB</td>
<td>39</td>
<td>55</td>
<td>71</td>
<td>G</td>
<td>103</td>
<td>g</td>
<td>119</td>
</tr>
<tr>
<td>8</td>
<td>BS</td>
<td>CAN</td>
<td>40</td>
<td>56</td>
<td>72</td>
<td>H</td>
<td>104</td>
<td>h</td>
<td>120</td>
</tr>
<tr>
<td>9</td>
<td>HT</td>
<td>EM</td>
<td>41</td>
<td>57</td>
<td>73</td>
<td>I</td>
<td>105</td>
<td>i</td>
<td>121</td>
</tr>
<tr>
<td>10</td>
<td>LF</td>
<td>SUB</td>
<td>42</td>
<td>58</td>
<td>74</td>
<td>J</td>
<td>106</td>
<td>j</td>
<td>122</td>
</tr>
<tr>
<td>11</td>
<td>VT</td>
<td>ESC</td>
<td>43</td>
<td>59</td>
<td>75</td>
<td>K</td>
<td>107</td>
<td>k</td>
<td>123</td>
</tr>
<tr>
<td>12</td>
<td>FF</td>
<td>FS</td>
<td>44</td>
<td>60</td>
<td>76</td>
<td>L</td>
<td>108</td>
<td>l</td>
<td>124</td>
</tr>
<tr>
<td>13</td>
<td>CR</td>
<td>GS</td>
<td>45</td>
<td>61</td>
<td>77</td>
<td>M</td>
<td>109</td>
<td>m</td>
<td>125</td>
</tr>
<tr>
<td>14</td>
<td>SO</td>
<td>RS</td>
<td>46</td>
<td>62</td>
<td>78</td>
<td>N</td>
<td>110</td>
<td>n</td>
<td>126</td>
</tr>
<tr>
<td>15</td>
<td>SI</td>
<td>US</td>
<td>47</td>
<td>63</td>
<td>79</td>
<td>O</td>
<td>111</td>
<td>o</td>
<td>127</td>
</tr>
</tbody>
</table>

Control Characters  Numerals  Lower-Case Letters
Punctuation Symbols  Upper-Case Letters
Empirical Results

Valid passwords over the ASCII characters

The Symbolic Algorithm, $L^∗ − Reduced$: [RS93]

<table>
<thead>
<tr>
<th>Password Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (pin)</td>
<td>Length: 4 to 8. Contains only numbers.</td>
</tr>
<tr>
<td>B (easy)</td>
<td>Length: 4 to 8. It contains any printable character.</td>
</tr>
<tr>
<td>D (medium-strong)</td>
<td>Length: 6 to 14. Contains at least 1 number and 1 lower-case letter. Punctuation characters are allowed.</td>
</tr>
<tr>
<td>E (strong)</td>
<td>Length: 6 to 14. Contains at least 1 character from each group.</td>
</tr>
</tbody>
</table>
Empirical Results

Valid passwords over the ASCII characters

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Length</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (pin)</td>
<td>Length: 4 to 8. Contains only numbers.</td>
<td>4-8</td>
<td>C1-100</td>
</tr>
<tr>
<td>B (easy)</td>
<td>Length: 4 to 8. It contains any printable character but punctuation characters.</td>
<td>4-8</td>
<td>C90-200</td>
</tr>
<tr>
<td>C (medium)</td>
<td>Length: 6 to 14. Contains any printable character but punctuation characters.</td>
<td>6-14</td>
<td>C50-150</td>
</tr>
<tr>
<td>D (medium-strong)</td>
<td>Length: 6 to 14. Contains at least 1 number and 1 lower-case letter. Punctuation characters are allowed.</td>
<td>6-14</td>
<td>C150-300</td>
</tr>
<tr>
<td>E (strong)</td>
<td>Length: 6 to 14. Contains at least 1 character from each group.</td>
<td>6-14</td>
<td>C200-350</td>
</tr>
</tbody>
</table>
Thank you!