Learning from an informant

Colin de la Higuera
University of Nantes
Acknowledgements

- Laurent Miclet, Jose Oncina and Tim Oates for previous versions of these slides.
- Rafael Carrasco, Paco Casacuberta, Rémi Eyraud, Philippe Ezequel, Henning Fernau, Thierry Murgue, Franck Thollard, Enrique Vidal, Frédéric Tantini,…
- List is necessarily incomplete. Excuses to those that have been forgotten.

http://pagesperso.lina.univ-nantes.fr/~cdlh/slides/

Chapter 12 (and part of 14)
Outline

1. The rules of the game
2. Basic elements for learning DFA
3. Gold’s algorithm
4. RPNI
5. Complexity discussion
6. Heuristics
7. Open questions and conclusion
1 The rules of the game
Motivation

- We are given a set of strings $S_+$ and a set of strings $S_-$
- Goal is to build a classifier
- This is a traditional (or typical) machine learning question
- How should we solve it?
Ideas

- Use a distance between strings and try k-NN
- Embed strings into vectors and use some off-the-shelf technique (decision trees, SVMs, other kernel methods)
Alternative

- Suppose the classifier is some grammatical formalism
- Thus we have $L$ and $\Sigma^* \setminus L$
Informed presentations

- An *informed* presentation (or an *informant*) of $\mathcal{L} \subseteq \Sigma^*$ is a function $f : \mathbb{N} \rightarrow \Sigma^* \times \{-,+\}$ such that $f(\mathbb{N}) = (\mathcal{L},+) \cup (\overline{\mathcal{L}},-)$
- $f$ is an infinite succession of all the elements of $\Sigma^*$ labelled to indicate if they belong or not to $\mathcal{L}$. 
Obviously many possible candidates

- Any Grammar $G$ such that
  - $S_+ \subseteq L(G)$
  - $S_- \cap L(G) = \emptyset$
- But there is an infinity of such grammars!
Structural completeness

- (of $S_+ \text{ re a DFA}$)
  each edge is used at least once
  each final state accepts at least one string

- Look only at DFA for which the sample is structurally complete!
Example

- $S_+=\{aab, b, aaaba, bbaba\}$
\[ S_+ = \{aab, b, aaaba, bbaba\} \ldots \]
Defining the search space by structural completeness

(Dupont, Miclet, Vidal 94)

- the basic operation: merging two states
- a bias on the concepts: structural completeness of the positive sample $S_+$
- a theorem: every biased solution can and can only be obtained by merging states in $CA(S_+)$
- the search space is a partition lattice.
$S_+ = \{aaa\}$

$S_- = \{a\}$
The partition lattice

- Let $E$ be a set with $n$ elements
- The number of partitions of $E$ is given by the Bell number

\[
\omega(0) = 1
\]

\[
\omega(n + 1) = \sum_{p=0}^{n} C_n^p \cdot \omega(n)
\]

$\omega(16) = 10\,480\,142\,147$
Regular inference as search

- another result: the smallest DFA fitting the examples is in the lattice constructed on PTA(S+)
- generally, algorithms would start from PTA(S+) and explore the corresponding lattice of solutions using the merging operation. $S_-$ is used to control the generalization.
$CA\ (S_+)$ or $PTA(S_+)$

Border Set

Universal Automaton

Zadar, August 2010
2 Basic structures
Two types of final states

\[ S_+ = \{ \lambda, \text{aaa} \} \]
\[ S_- = \{ \text{aa}, \text{aaaaa} \} \]

1 is accepting
3 is rejecting
What about state 2?
What is determinism about?

Merge 1 and 3?

But...
The prefix tree acceptor

- The smallest tree-like DFA consistent with the data
- Is a solution to the learning problem
- Corresponds to a rote learner
From the sample to the PTA

\[ PTA(S_+) \]

\[ S_+ = \{ \lambda, aaa, aaba, ababa, bb, bbaaa \} \]

\[ S_- = \{ aa, ab, aaaa, ba \} \]
From the sample to the PTA (full PTA)

PTA($S_+,$$S_-)$

$S_+=${$\lambda$, $aaa$, $aaba$, $ababa$, $bb$, $bbaaa$}$

$S_-=${$aa$, $ab$, $aaaa$, $ba$}$
Red, Blue and White states

- **Red** states are confirmed states
- **Blue** states are the (non Red) successors of the Red states
- **White** states are the others
Merge and fold

Suppose we want to merge state 3 with state 2
Merge and fold

First disconnect 3 and reconnect to 2
Merge and fold

Then fold subtree rooted in 3 into the DFA starting in 2
Merge and fold

Then fold subtree rooted in 3 into the DFA starting in 2
Other search spaces

an augmented $PTA$ can be constructed on both $S_+$ and $S_-$ (Coste 98, Oliveira 98)

- but not every merge is possible
- the search algorithms must run under a set of dynamic constraints
State splitting

Searching by splitting

- start from the one-state universal automaton, keep constructing DFA controlling the search with $\langle S_+, S_- \rangle$
That seems a good idea... but take $a^{5*}$. What 4 (or 3, 2, 1) state automaton is a decent approximation of $a^{5*}$?
3 Gold’s algorithm

Key ideas

- Use an observation table
- Represent the states of an automaton as strings, prefixes of the strings in the learning set
- Find some incompatibilities between these prefixes due to separating suffixes
- This leads to equivalent prefixes
- Invent the other equivalences
Strings as states
Incompatible prefixes

- $S_+={aab}$
- $S_-={bab}$
- Then clearly there are at least 2 states, one corresponding to $a$ and another to $b$. 
Observation table

- The information is organised in a table <STA,EXP,OT> where:
  - $RED \subseteq \Sigma^*$ is the set of states
  - $BLUE \subseteq \Sigma^*$ is the set of transitions
    \[ BLUE = (RED . \Sigma) \setminus RED \]
  - $EXP \subseteq \Sigma^*$ is the experiment set
  - $OT: (STA=RED \cup BLUE) \times EXP \to \{0,1,*\}$ such that:
    \[ OT[u][e] = \begin{cases} 
    1 & \text{if } ue \in S_+ \\
    0 & \text{if } ue \in S_- \\
    * & \text{otherwise}
  \end{cases} \]
### An observation table

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$aa$</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>$ab$</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

#### The states (RED)

#### The transitions (BLUE)

#### The experiments (EXP)
### Meaning

<table>
<thead>
<tr>
<th></th>
<th>λ</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

\[
\delta(q_0, \lambda \cdot \lambda) \in F \iff \lambda \in L
\]

<table>
<thead>
<tr>
<th></th>
<th>λ</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>aa</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>ab</td>
<td>*</td>
<td>0</td>
</tr>
</tbody>
</table>
\delta(q_0, ab \cdot a) \not\in F \iff \text{aba} \not\in L
Redundancy

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
<th>$aa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>*</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>*</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>$aa$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ab$</td>
<td>1</td>
<td>*</td>
<td>1</td>
</tr>
</tbody>
</table>

Must have identical label (redundancy)
DFA consistent with a table

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

Both DFA are consistent with this table
Closed table without holes

- Let $u \in \text{RED} \cup \text{BLUE}$ and let $OT[u][e]$ denote a cell.
  $OT[u]$ denotes the row indexed by $u$.

- We say that the table is **closed if**
  $\forall t \in \text{BLUE}, \exists s \in \text{RED} : OT[t] = OT[s]$.

- We say that the table has **no holes if**
  $\forall u \in \text{RED} \cup \text{BLUE}, \forall e \in E \; OT[u][e] \in \{0,1\}$. 
This table is closed

<table>
<thead>
<tr>
<th></th>
<th>(\lambda)</th>
<th>(a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(a)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(b)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(aa)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(ab)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
This table is not closed

<table>
<thead>
<tr>
<th></th>
<th>λ</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>0</td>
<td>1</td>
<td>Not closed</td>
</tr>
<tr>
<td>ab</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
An equivalence relation

Let \( <\text{RED}, \text{EXP}, \text{OT}> \) be a closed table and with no holes. Consider the equivalence relation over \( \text{STA} \):

\[
\begin{align*}
{s_1} \equiv {s_2} \\
\text{if} \\
\text{OT}[s_1] = \text{OT}[s_2] \land \forall a \in \Sigma \; \text{OT}[s_1a] = \text{OT}[s_2a]
\end{align*}
\]

Class of \( s \) is denoted by \([s]\) (also!)
Equivalent prefixes

<table>
<thead>
<tr>
<th></th>
<th>λ</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Prefixes λ and b are equivalent since \( OT[λ] = OT[b] \)
Building an automaton from a table

We define \( A(<\text{STA,EXP,OT}>) = (\Sigma, Q, \delta, q_0, F) \) as follows:

- \( Q = \{ [s]: s \in \text{RED} \} \)
- \( q_0 = \lambda \)
- \( F = \{ [ue]: \text{OT}[u][e] = 1 \} \)
- \( \delta([s_1], a) = \)
  - \( [s_2a] \text{ if } \exists s_2 \in [s_1]: s_2a \in \text{RED} \)
  - any \( [s]: s \in \text{RED} \land \text{OT}[s] = \text{OT}[s_1a] \)
<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c|cc}
 b & 1 & 0 \\
 aa & 1 & 0 \\
 ab & 0 & 1 \\
\end{array}
\]
Compatibility Theorem:

- Let $<STA, EXP, OT>$ be an observation table closed and with no holes
- If $STA$ is prefix-complete and $EXP$ is suffix-complete then $A(<STA, EXP, OT>)$ is compatible with the data in $<STA, EXP, OT>$
Example

- \( RED = [\lambda] \)
- \( Q = \{[\lambda]\} \)
- \( q_0 = [\lambda] \)
- \( F = \{[\lambda]\} \)

<table>
<thead>
<tr>
<th></th>
<th>( \lambda )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( b )</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Exercise

- **Build a DFA from either of these two tables**

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$b$</th>
<th>$bbb$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
</tr>
<tr>
<td>$aaa$</td>
<td>0</td>
</tr>
<tr>
<td>$aaaa$</td>
<td>0</td>
</tr>
</tbody>
</table>
Building the initial table from a sample

- Given a sample $S$ and a set of strings (RED) prefix-complete, it is always possible to select a set of experiments $E$ such that the table $\langle STA, E, OT \rangle$ contains all the information in $S$
- But usually this table is going to have holes
Obviously different rows

Let $s_1, s_2 \in RED \cup BLUE$
we say that $OT[s_1]$ is obviously different from $OT[s_2]$ if

$$\exists e \in E:$$

$$OT[s_1][e], OT[s_2][e] \in \{0, 1\} \text{ and } OT[s_1][e] \neq OT[s_2][e]$$

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$aa$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$ab$</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
If \( \exists t \in \text{BLUE} \) such that \( OT[t] \) is obviously different from any \( OT[s] \), \( (s \in \text{RED}) \) then no filling of holes in \( \langle \text{RED}, \text{EXP}, OT \rangle \) can produce a closed table.

\[
\begin{array}{c|cc}
\lambda & \lambda & a \\
\hline
\lambda & 1 & 0 \\
a & 0 & 0 \\
\hline
b & 1 & * \\
aa & 0 & * \\
ab & * & 1 \\
\end{array}
\]

\( ab \) is OD with each \( s \)
Algorithm

\( \text{RED} \leftarrow \{\lambda\} \)

build \(<\text{RED},E,OT>\) with \(E\) suffix-complete

while \(\exists x \in \text{BLUE}: \text{OT}[x] \text{ is OD}\) do

add \(x\) to \(\text{RED}\)

update \(\text{BLUE}\)

update \(<\text{STA},E,OT>\)
There can be several such $t'$

$$Q \leftarrow \text{RED}$$

$$q_0 \leftarrow \lambda$$

$$F \leftarrow \{ t \in \text{RED} : OT[t][\lambda] = 1 \}$$

$$\delta(t,a) \leftarrow \begin{cases} ta & \text{if } ta \in \text{RED} \\ t' & \text{if } t' \in \text{RED} \text{ and not } \text{OD} \end{cases}$$

if $<\text{STA,EXP,OT}>$ is incompatible with $S$, return the PTA.
Example run

- $S_+ = \{bb, abb, bba, bbb\}$
- $S_- = \{a, b, aa, ba, bab\}$

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
<th>$b$</th>
<th>$aa$</th>
<th>$ab$</th>
<th>$ba$</th>
<th>$bb$</th>
<th>$abb$</th>
<th>$bab$</th>
<th>$bba$</th>
<th>$bbb$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$a$</td>
<td>0 0 * * * 1 * * * * *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>0 0 1 * 0 1 1 * * * *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\lambda$ and $b$ are OD
<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$a$</th>
<th>$b$</th>
<th>$aa$</th>
<th>$ab$</th>
<th>$ba$</th>
<th>$bb$</th>
<th>$abb$</th>
<th>$bab$</th>
<th>$bba$</th>
<th>$bbb$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>$a$</td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>$ba$</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>$bb$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

1) We promote line $b$

2) We expand the table, adding rows $ba$ and $bb$

3) $bb$ is OD
<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>$a$</th>
<th>$b$</th>
<th>$aa$</th>
<th>$ab$</th>
<th>$ba$</th>
<th>$bb$</th>
<th>$abb$</th>
<th>$bab$</th>
<th>$bba$</th>
<th>$bbb$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>$*$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$*$</td>
<td>$0$</td>
<td>$1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$b$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
<td>$*$</td>
<td>$0$</td>
<td>$1$</td>
<td>$1$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
</tr>
<tr>
<td>$bb$</td>
<td>$1$</td>
<td>$1$</td>
<td>$1$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
</tr>
<tr>
<td>$a$</td>
<td>$0$</td>
<td>$0$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$1$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
</tr>
<tr>
<td>$ba$</td>
<td>$0$</td>
<td>$*$</td>
<td>$0$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
</tr>
<tr>
<td>$bba$</td>
<td>$1$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
</tr>
<tr>
<td>$bbb$</td>
<td>$1$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
<td>$*$</td>
</tr>
</tbody>
</table>

1) We promote line $bb$

2) We expand the table, adding rows $bba$ and $bbb$

3) We construct the automaton as no line is OD
Now the * have to be replaced
<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>( b )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( bb )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( a )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( ba )</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>( bba )</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>( bbb )</td>
<td>1</td>
<td>*</td>
</tr>
</tbody>
</table>

Wild guess!
\[ S_+ = \{bb, abb, bba, bbb\} \]
\[ S_- = \{a, b, aa, ba, bab\} \]

The automaton is inconsistent. We shall have to return the PTA instead.
But!

- $a_{Gold}$ is deterministic: it takes deterministic decisions in order to solve the « filling holes » question
- In practice it will very often return the PTA
Equivalence of problems

Let $RED$ be a state test set prefix-complete, and $S$ be a sample. Let $<STA, EXP, OT>$ be an observation table consistent with all the data in $S$, with $EXP$ suffix-complete.

The question:

Does there exist a DFA with the states of $RED$ and compatible with $S$?

is equivalent to:

Can we fill the holes such that $<STA, EXP, OT>$ is closed?
Complexity

The problem:

is there a DFA with states in RED and compatible with $S$?

is NP-Complete
Corollary

Given $S$ and a positive integer $n$, the question:

Is there a DFA with $n$ states compatible with $S$?

is NP-Complete
Properties of \( a_{\text{Gold}} \)

1) the output is consistent with a sample \( S \)
2) \( a_{\text{Gold}} \) identifies in the limit any regular language
3) \( a_{\text{Gold}} \) works in time polynomial in \(|S|\)
4) if the size of the target is \( n \), then there is a characteristic sample \( CS \) with \(|CS| = 2n^2(|\Sigma|+1)\), such that \( a_{\text{Gold}}(S) \) produces the canonical acceptor for all \( S \supseteq CS \)
Open questions

- Can one fill the holes in a more “intelligent” way?
- How fast can we detect that a choice (for a filling) is good or bad?
Exercise

- Run Gold’s algorithm for the following data:
  - \( S_+ = \{ a, abb, bab, babbb \} \)
  - \( S_- = \{ ab, bb, aab, b, aaaa, babb \} \)
4 RPNI

Regular Positive and Negative Grammatical Inference

- RPNI is a state merging algorithm
- RPNI identifies any regular language in the limit
- RPNI works in polynomial time
- RPNI admits polynomial characteristic sets
\( A = \text{PTA}(S^+); \ Blue = \{ \delta(q_I, a) \colon a \in \Sigma \}; \)
\( Red = \{ q_I \} \)

While \( Blue \neq \emptyset \) do
  
  choose \( q \) from \( Blue \)
  
  if \( \exists p \in Red : \text{L}(\text{merge\_and\_fold}(A, p, q)) \cap S^- = \emptyset \) then \( A = \text{merge\_and\_fold}(A, p, q) \)
  
  else \( Red = Red \cup \{ q \} \)

\( Blue = \{ \delta(q, a) : q \in Red \} - \{ Red \} \)
\[ S_+ = \{ \lambda, \text{aaa, aaba, ababa, bb, bbaaa} \} \]

\[ S_- = \{ \text{aa, ab, aaaa, ba} \} \]
Try to merge 2 and 1

$S_\omega = \{aa, \ ab, \ aaaa, \ ba\}$
First merge, then fold

\[ S_\gamma = \{aa, ab, aaaa, ba\} \]
But now string \texttt{aaaa} is accepted, so the merge must be rejected, and state 2 is promoted.

\[ S_\text{-} = \{aa, \ ab, \ aaaa, \ ba\} \]
Try to merge 3 and 1

\[ S_\ominus = \{aa, ab, aaaa, ba\} \]
First merge, then fold

\[ S_+ = \{aa, ab, aaaa, ba\} \]
No counter example is accepted so the merge is kept

\[ S_\ast = \{ aa, ab, aaaa, ba \} \]
Next possible merge to be checked is \{4,13\} with \{1,3,6\}

\[ S_\text{-} = \{aa, ab, aaaa, ba\} \]
Merged. Needs folding subtree in \{4,13\} with \{1,3,6\}

\[ S_\downarrow = \{aa, ab, aaaa, ba\} \]
But now \textit{aa} is accepted

\[S_\epsilon = \{ aa, \ ab, \ aaaa, \ ba \}\]
So we try \{4,13\} with \{2,10\}

\[ S_\equiv \{aa, ab, aaaa, ba\} \]
Negative string $aa$ is again accepted. Since we have tried all Red for merging, state 4 is promoted.

$$S_- = \{aa, ab, aaaa, ba\}$$
So we try 5 with \{1,3,6\}

\[ S_\_ = \{aa, ab, aaaa, ba\} \]
But again we accept \( ab \)

\[
S_- = \{ aa, \ ab, \ aaaa, \ ba \}
\]
So we try 5 with \{2,10\}

\[ S_\ast = \{aa, \ ab, \ aaaa, \ ba\} \]
Which is OK. So next possible merge is \{7,15\} with \{1,3,6\}

\[S_-=\{aa, ab, aaaa, ba\}\]
Which is OK. Now try to merge \{8,12\} with \{1,3,6,7,15\}

\(S_\ast=\{aa, ab, aaaa, ba\}\)
And $ab$ is accepted

$S_-\{aa, \ ab, \ aaaa, \ ba\}$
Now try to merge \{8,12\} with \{4,9,13\}

\[ S_\pm = \{aa, ab, aaaa, ba\} \]
This is OK and no more merge is possible so the algorithm halts

\[ S_- = \{ aa, ab, aaaa, ba \} \]
Properties

- RPNI identifies any regular language in the limit
- RPNI works in polynomial time. Complexity is in $O(\|S_+\|^3 \|S_-\|)$
- There are many significant variants of RPNI
- RPNI can be extended to other classes of grammars
Exercises

- Run RPNI on
  - $S_+ = \{a, bba, bab, aabb\}$
  - $S_- = \{b, ab, baa, baabb\}$
- Find a characteristic sample for:

```
0 1 2 3
0---a---1
|    |   |
|    b|   |
|    |   |
|    | b a|
|    |   |
|    | b |

Zadar, August 2010
5 Complexity issues
A characteristic sample

- A sample is characteristic (for some algorithm) whenever, when included in the learning sample, the algorithm returns the correct DFA
- The characteristic sample should be of polynomial size
- There is an algorithm which given a DFA builds a characteristic sample
Definition: polynomial characteristic sample

\( \mathcal{G} \) has polynomial characteristic samples for identification algorithm \( a \) if there exists a polynomial \( p() \) such that: given any \( G \) in \( \mathcal{G} \),

\[ \exists CS \text{ correct sample for } G, \text{ such that when } CS \subseteq f_n, a(f_n) \equiv G \text{ and } \| CS \| \leq p(\| G \|) \]
About characteristic samples

- If you add more strings to a characteristic sample it still is characteristic.
- There can be many different characteristic samples (EDSM, tree version,...)
- Change the ordering (or the exploring function in RPNI) and the characteristic sample will change.
Open problems

- RPNI’s complexity is not a tight upper bound. Find the correct complexity.
- The definition of the characteristic sample is not tight either. Find a better definition.
- Can there be a linear time DFA learner?
Collusion

- Collusion consists in having the learner and the teacher agree on some specific encoding system. Then, the teacher can just pass one string which is the encoding of the target.

- Is that cheating?
- Is that learning?
6 Heuristics
6.1 Genetic Algorithms

- The principle: via evolutionary mechanisms nature increases the quality of its population.

- Allow a population of solutions to interact and evolve.
Mechanisms (gene level):

- Mutation
- Crossing-over

(a solution is just a string)
Mutation

TTAGCCTTC

↓

TTTGCCCTTC
Crossing-over

- TTATCCGT

  TTATC CGT
  TTATC CTTC
  TTATCCTTC

  TAGGCTTTC
  TAGG CTTC
  TAGG CGT
  TAGGCGGT
Idea: define the solutions as sequences

- Be able to measure the quality of a solution
- Conceive a first generation
- Define the genetic operations (mutation, crossing over)
- Keep the best elements of the second generation
- Iterate
Genetic algorithms in Grammatical Inference

- (Dupont 94)
  - code the automata (the partition of states of $PTA(S_\dagger)$) into partitions
  - define genetic operators
  - define an optimum as an automaton with as few states as possible and rejecting $S_\dagger$
  - run the genetic algorithm
Structural Mutation

- Select a state from a block and move it to another block
- Example:
  \[
  \begin{align*}
  \{1, 3\} & \quad \{2\} & \quad \{4, 5\} \\
  \{1\} & \quad \{2\} & \quad \{3, 4, 5\} \\
  \{1\} & \quad \{2\} & \quad \{3\} & \quad \{4, 5\}
  \end{align*}
  \]
Structural crossover

\{1,2\}\{3,4,5\} \quad \{1,3\}\{4\}\{2,5\}

\{1,3\}\{3,4,5\} \quad \{1,2\}\{4\}\{2,5\}

\{1,2,3\}\{4,5\} \quad \{1,2,3\}\{4\}\{5\}
Group number encoding

Partition

\[ \{\{1,2,6\}\{3,7,9,10\}\{4,8,12\}\{5\}\{11\}\} \]

is encoded by

\[ (112341232253) \]
6.2 Tabu search

- (Giordano 96, based on Glover 89)

- General idea: search a space by choosing a point, and going to its best neighbor that is not in the tabu list.
\( R \leftarrow \) the set of rules of the grammars in the search space
\( G \leftarrow \) an initial grammar
\( G^* \leftarrow G \) the best solution reached so far
\( T \leftarrow \emptyset \) the Tabu list that cannot occur
\( k \leftarrow 0 \) the iterations counter
While $k \neq k_{\text{max}}$ do

select $r$ in $R \setminus T$, such that the addition or deletion of $r$ from $G$ realizes the maximum of $\text{val}$ on $X$

add or delete $r$ from $G$

if $\text{val}(G) > \text{val}(G^*)$ then $G^* \leftarrow G$

Update $T$

$k \leftarrow k + 1$

Return $G^*$
- Procedure Update\((T, r)\)
  
  \[
  \text{if } \text{card}\(T\) = \text{\_}n\text{\_ then delete its last element}
  \]
  
  Add \(r\) as the first element of \(T\)

- Tricks
  
  - \textbf{If} \(\text{\_blocked\_ then delete oldest rule}\)
  
  - \(\text{\_blocked\_} \leftarrow 6\) \text{\_iterations\_}
  
  - \text{if new} \(G^*\) \text{\_then empty}(T)\)
6.3 Heuristic greedy State Merging

- RPNI chooses to merge the first 2 states that can be merged
- This is an optimistic view
- There may be another...
- But remember: RPNI identifies in the limit!
How do greedy state merging algorithms work?

- choose two states
  - perform a cascade of forced merges to get a deterministic automaton
  - if it accepts sentences of $S_-$, backtrack and choose another couple
  - if not, loop until no merging is still possible
The blue fringe (Lang 98)
What moves are allowed?

- Merging a ◼ with a ◼
- Promoting a ◼ to ◼ and all its successors that are not ◼ to ◼

- Promotion:
  - when a ◼ can be merged with no ◼
What if there are many merges possible?

- Heuristics
- compute a score
- choose highest score
The blue fringe (Lang 98)
Evidence driven (Lang 98)

for each possible pair (●, ○) do
    parse $S_+$ and $S_-$ on $A$ resulting from the merge
    assign a score to each state of $A$ according to the sentences that they accept
    if there is a conflict: -∞
    else the number of sentences accepted
    sum over all states ⇒ the score of the merge
    select the merge with the highest score
Data driven (cdlh, Oncina & Vidal 96)

For every ⬤ or ⬦ state in $A$ count

$$v_+(q) = \sum_{w \in S_+} \left| \left\{ u \in \text{Pref}(w) : \delta(q_0, u) = q \right\} \right|$$

$$v_-(q) = \sum_{w \in S_-} \left| \left\{ u \in \text{Pref}(w) : \delta(q_0, u) = q \right\} \right|$$

Choose the pair (⬤, ⬦) such that

$$\min(v_+(⬤), v_+(⦦)) + \min(v_-(⬤), v_-(⦦))$$

is maximal

Zadar, August 2010
Careful

- Count first...

  ...then try to merge

- Keep track of all tries

- if some is not mergeable, promote it!
Main differences

- data driven is cheaper
- evidence driven won Abbadingo competition

- In the stochastic case, it seems that data driven is a good option...
6.4 Constraint Satisfaction

**PTA**

\[(ababc, +) (c, +)\]
\[(aac, -) (ab, -) (abac, -) (a, -)\]
Inconsistent

(ababc, +) (c, +)

(aac, -) (ab, -) (abac, -) (a, -)
Consider \((Q, \text{incompatible})\)

- All you have to do is find a maximum clique...
- Another NP-hard problem, but for which good heuristics exist.
- Careful: the maximum clique only gives you a lower bound...
Alternatively

- You have \( \mathcal{Q} \) variables \( S_1..S_{\mathcal{Q}} \), and \( n \) values 1..\( n \).
- You have constraints
  \[
  S_i \neq S_j \\
  \text{or} \quad S_i = S_j \implies S_k = S_l
  \]
- Solve.

Biermann 72, Oliveira & Silva 98, Coste & Nicolas 98
7 Open questions and conclusions
Other versions

- A Matlab version of RPNI
  http://www.sec.in.tum.de/~hasan/matlab/gi_toolbox/1.0-Beta/

- A JAVA version
  http://pagesperso.lina.univ-nantes.fr/~cdlh/Downloads/RPNIP.tar.gz

- A parallel version exists, and an OCAML, C, C++...
Some open questions

- Do better than EDSM (still some unsolved Abbadingo task out there...)
- Write a $O(\| f(n) \|)$ algorithm which identifies DFA in the limit (Jose Oncina and cdlh have a log factor still in the way)
- Identify and study the collusion issues
- Deal with noise.