On Learning Regular Expressions and Patterns via Membership and Correction Queries

Efim Kinber
Sacred Heart University
Fairfield, CT, USA

Query Learning Model (D. Angluin):

L – target language, *w* – a string

- **Membership queries:** "*w* in L?"

  Answer: "YES" OR "NO"

- **Correction queries:** "*w* in L?"

  Answer: "YES" or a *correction*:

  a string in L

  "closest" to *w.*

A learner asks a finite number of queries and returns a description for the target language (e.g., a regular expression, a dfa, a pattern)

Corrections:

- Not seen in the learning process so far
- At the shortest Levenshtein edit distance (preference given to corrections of the same length as the queried string)
- Smallest in the lexicographic order among those satisfying the above conditions

(Originally, corrections – ***shortest extensions*** of the queried string:

- Introduced by L. Beccera-Bonache, A. Dediu, C. Tîrnăucă : learning DFA
- C. Tîrnăucă , T. Knuutila: learning *k*- reversible languages, patterns
- Corrections – at the ***shortest edit distance:***

  L. Beccera-Bonache, C. de la Higuera, J. Janodet, F. Tantini: learning balls of strings)

A Class of Regular Expressions:

$$u_1(v_1)^+u_2(v_2)^+\ldots u_n(v_n)^+u_{n+1}$$

u_i , v_i – strings over an alphabet Σ.

Example:

$$a(aa)^+bccd(cd)^+ddabb^+b(abb)^+$$

Two more conditions:

- Left-aligned:

$$(aa)^+abc(cd)^+cddab^+bb(abb)^+$$

- No subexpressions of the type $((ab)^3)^+((ab)^5)^+$

Learning:

Using

- **One correction query** “empty string in L?”

  (to find the *shortest string* in L)

- **Membership queries**

  (another way to find the shortest string: given an arbitrary string in L, use membership queries)

Example:

Target expression: $a^+a(ab)^+ab^+(ab)^+ab^+a(bb)^+$

Shortest string: *aaababababababb*

First step: **finding loops of length 1**

- Query "***a***aaaababababababb in L?" (one extra *a* in the first block of *a*-s)
- Query "aaa***b***bababababababb in L?" (answer is "no")
- Etc.

RESULT: $a^+aabab^+abab^+abb$

Example (continued):

Target expression: $a^+a(ab)^+ab^+(ab)^+ab^+a(bb)^+$

Result of Step 1: $a^+aabab^+abab^+abb$

Step 2: **finding loops of length 2:**

- Query "*aa**ab**aba**b**babababb* in L?" ("yes")
  ("*aa**ab**abababababb* in L?" does not work!)
  New conjecture: $a^+a(ab)^+ab^+abab^+abb$
- Query "*aaaba**b**b**ab**aba**b**babb* in L?" ("yes")
  New conjecture: $a^+a(ab)^+ab^+(ab)^+ab^+abb$
- Etc.

Key point: *(under certain conditions,) the neighboring loops on the left and/or on the right in the current conjecture must be expanded for the next queried string*

Complexity:

- Running time: $O(n^5)$
- Number of queries: $O(n^3)$

n - the length of the target expression (equivalently, the length of the shortest example)

Discussion

A similar algorithm for the following modification:

- * instead of +
- one-letter loops only
- nonempty strings between any two loops

(example: *a*aab*ac*ca*aa*)

Similar to a class considered by H. Fernau (learnable in the limit from positive data)

Most of the classes of regular expressions considered in literature are deterministic (or one-unambiguous), whereas our class is not

Patterns (introduced by D. Angluin):

Σ – an alphabet (of constants)

X - a countable (infinite) set of variables

A pattern π – a string over $(\Sigma \cup X)^*$

Example: 0xx10yxy0z

A substitution: 01 → x, 1→ y, 10 → z

(all variables are substituted by nonempty strings over Σ)

Get the string: 00101101011010

The language $L(\pi)$: all strings obtained by such substitutions

Learning Patterns

A hard problem:

- Membership problem: NP-complete
- Inclusion problem: undecidable

(Equivalence problem: decidable in linear time, but does not help much)

LEARNABILITY (under various protocols) extensively studied

Recently: C. Tîrnăucă , T. Knuutila: learning from *correction queries,* where a correction is the shortest extension of the queried string

Learnability can be applied for various practical problems, including learning DNA structures (a survey by T. Shinohara, S. Arikawa)

Our Learning Algorithm (using correction queries):

On example: x01yzuuuuzzzzvvv

- To find a shortest string, query "empty string in L?" Get the correction string: 0010^{13}
- Query "1^{16} in L?" Get the correction string: 101^{14}

Constants 0 and 1 have been found. To find variables, let U01UUUUUUUUUUUU be the current conjecture, where U is a placeholder for a variable

Learning Algorithm (continued):

Current conjecture: U01UUUUUUUUUUUU

- Query "1010^{13} in L?" (trying 1 for the first position). Answer is "yes". The new conjecture is x01UUUUUUUUUUUU
- Query "00110^{12} in L?" (trying 1 for the 4^{th} postion). Answer is "yes". The new conjecture is x01yUUUUUUUUUUU
- Query "001010^{12} in L?" Answer is "no". Correction string: 10110^{12} (where x, y are substituted by 1)
- Query "101110^{11} in L?" Correction string: $10110^{9}111$. The new conjecture: x01yUUUUUUUUUvvv

Learning Algorithm (continued)

Target pattern: x01yzuuuuzzzzvvv

Current conjecture: x01yUUUUUUUUUUvvv

- Query "$101110^{8}111$ in L?" (trying 1 for the 5^{th} position one more time). Correction string: 101110000111111. New conjecture: x01yzUUUUzzzzvvv
- Query "0010010^{10} in L?" Correction: 0010011110^{7} (as closer correction strings have been seen). The final result is: x01yzuuuuzzzzvvv

One more example:

Target pattern: 10xx

- Query empty string. Get correction: 1000.
- Query 1111. Get correction 1011. Conjecture 10UU (U - placeholder)
- Query 1010. The answer is “no”. Correction string must be longer than 4 (as both 1000 and 1011 have already been used). The algorithm terminates and returns 10xx (replacing all placeholders by the same variable)

Complexity

- Running time: $O(n^3)$
- Number of queries: $O(n^2)$

n - the length of the target pattern

Discussion

- The algorithm can be extended for an arbitrary alphabet Σ
- Why corrections of the same length? Strings of arbitrary length are of little help (Lange, Wiehagen)
- Requirement of using the lexicographically smallest shorted example can be slightly relaxed
- Conjecture: patterns cannot be learned in poly-time using corrections at the shortest edit distance with no constraints