Polynomial-Time Algorithms for Learning Typed Pattern Languages

Michael Geilke[1] and Sandra Zilles[2]

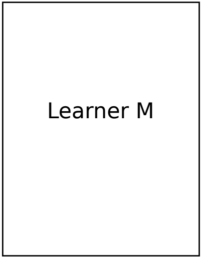
[1]Johannes Gutenberg University Mainz

[2]University of Regina

March 08, 2012

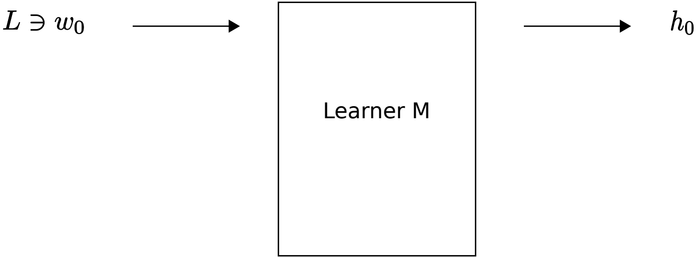
Learning from positive data

Let $\mathcal{L}$ be a set of languages, $L \in \mathcal{L}$ be the target language.



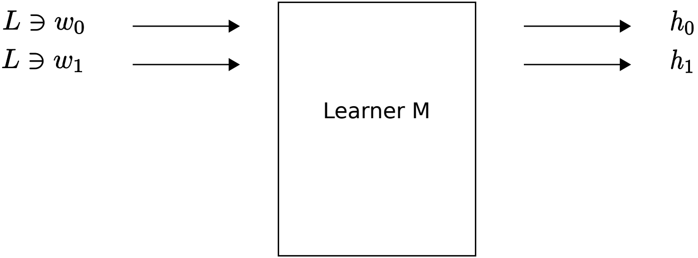
Learning from positive data

Let $\mathcal{L}$ be a set of languages, $L \in \mathcal{L}$ be the target language.



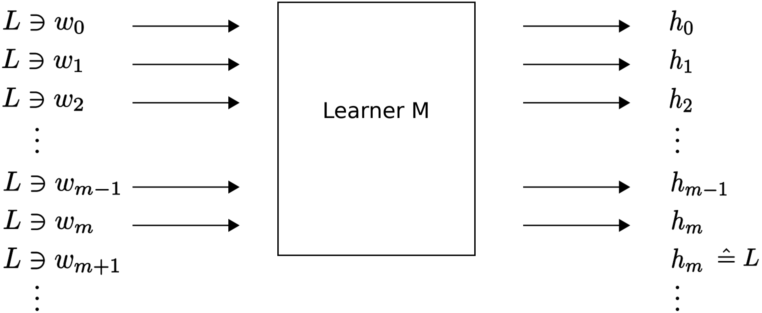
Learning from positive data

Let $\mathcal{L}$ be a set of languages, $L \in \mathcal{L}$ be the target language.



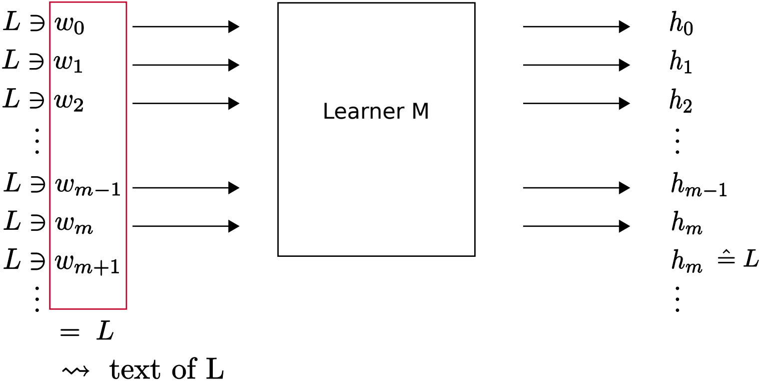
Learning from positive data

Let $\mathcal{L}$ be a set of languages, $L \in \mathcal{L}$ be the target language.



Learning from positive data

Let $\mathcal{L}$ be a set of languages, $L \in \mathcal{L}$ be the target language.



Pattern Languages

$\Sigma = \{a, b, ...\}$ be a finite set of **terminal symbols** with $|\Sigma| \geq 2$
$X = \{x_1, x_2, ...\}$ be a countable set of **variables** such that $\Sigma \cap X = \emptyset$

Informal definition (Angluin)

A **pattern** *is any finite string over terminal symbols and variables.*
The **language of a pattern** *p is the set of all words that result from substituting all variables in p by strings of terminal symbols.*

Example

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

$$p \quad = \quad (ab)^3 \quad x_1 \quad x_2 \quad b^2c^4 \quad x_3 \quad b^3$$

Pattern Languages

$\Sigma = \{a, b, ...\}$ be a finite set of **terminal symbols** with $|\Sigma| \geq 2$
$X = \{x_1, x_2, ...\}$ be a countable set of **variables** such that $\Sigma \cap X = \emptyset$

Informal definition (Angluin)

A **pattern** *is any finite string over terminal symbols and variables.*
The **language of a pattern** *p is the set of all words that result from substituting all variables in p by strings of terminal symbols.*

Example

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

$$p \quad = \quad (ab)^3 \quad \mathbf{x_1} \quad \mathbf{x_2} \quad b^2 c^4 \quad \mathbf{x_3} \quad b^3$$

Pattern Languages

$\Sigma = \{a, b, ...\}$ be a finite set of **terminal symbols** with $|\Sigma| \geq 2$
$X = \{x_1, x_2, ...\}$ be a countable set of **variables** such that $\Sigma \cap X = \emptyset$

Informal definition (Angluin)

A **pattern** *is any finite string over terminal symbols and variables.*
The **language of a pattern** *p is the set of all words that result from substituting all variables in p by strings of terminal symbols.*

Example

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

$$\theta(p) = \quad (ab)^3 \quad a^4 \quad ba \quad b^2c^4 \quad c^3 \quad b^3$$

Typed Pattern Languages

Bibliographic data entry system:

$$\text{Author: } x_1, \text{ Title: } x_2, \text{ Year: } x_3$$

Introduction of types

Each variable has exactly one type: $\mathcal{T} := \{t_1, t_2\}$

$L_{t_1} = \Sigma^+,$ $\quad X_{t_1} := \{x_1, x_2\},$
$L_{t_2} = \{1900, \ldots, 2100\} \cup \{\epsilon\}$ $\quad X_{t_2} := \{x_3\}$

Learning Pattern Languages Efficiently - Problems

The membership problem

Given:	pattern p, word w
Question:	does p generate w?

Theorem (Angluin)

The membership problem for the class of untyped pattern languages is NP-complete.

$\rightsquigarrow$ avoid membership tests during the learning process

Learning Untyped Pattern Languages (1)

Theorem (Lange and Wiehagen)

The class of untyped pattern languages as introduced by Angluin can be learned in polynomial time.

Idea: Only take words of shortest length to infer the pattern.

Learning Untyped Pattern Languages (1)

Theorem (Lange and Wiehagen)

The class of untyped pattern languages as introduced by Angluin can be learned in polynomial time.

Idea: Only take words of shortest length to infer the pattern.

Example

$\Sigma = \{a, b\}$, $L_t = \Sigma^+$

aaabbaab
aabaaabbbbbabaaaab
aabbbaab
ababbbab
abbbbbaaabbbbbbbaaab
abbbbbbbab

Learning Untyped Pattern Languages (1)

Theorem (Lange and Wiehagen)

The class of untyped pattern languages as introduced by Angluin can be learned in polynomial time.

Idea: Only take words of shortest length to infer the pattern.

Example

$\Sigma = \{a, b\},\ L_t = \Sigma^+$

aaabbaab
~~aabaaabbbbbabaaaab~~
aabbbaab
ababbbab
~~abbbbbaaabbbbbbaaab~~
~~abbbbbbbbab~~

Learning Untyped Pattern Languages (2)

Why is the set of shortest words sufficient?

There is a subset S_p of $L(p)$ with $|S_p| \leq 2|p|$ such that S_p is a characteristic set with respect to the set of untyped pattern languages.

a	x_1	x_2	bb	x_1	ab
a	**a**	a	bb	**a**	ab
a	**b**	a	bb	**b**	ab
a	a	**a**	bb	a	ab
a	a	**b**	bb	a	ab

$\rightsquigarrow$ de la Higuera's **characteristic sets**

Learning Typed Pattern Languages (1)

For typed pattern languages this does no longer work in general!

Example

- $\Sigma := \{a, b\}$, $t_1 := \{a, b\}$ and $t_2 := \{aa, ab, ba, bb, aaa, bbb\}$
- $p := x_{(t_1,1)}\ x_{(t_2,1)}$
- $q := x_{(t_1,1)}\ x_{(t_1,2)}\ x_{(t_1,3)}$

$L(p)$ and $L(q)$ have the **same set of shortest words**, $S := \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$, **but**

$$L(p) = S \cup \{aaaa, baaa, abbb, bbbb\} \neq S = L(q).$$

Learning Typed Pattern Languages (2)

Type Witnesses

Let $\mathcal{T}$ be a set of subsets of Σ^+. (ω_1, ω_2) is a **type witness** for $\mathcal{T}$ if

- $\omega_1, \omega_2 : \mathcal{T} \to \Sigma^+$ are mappings
- $\omega_1(t) \neq \omega_2(t)$ and $\{\omega_1(t), \omega_2(t)\} \subseteq t \setminus \bigcup_{t' \in \mathcal{T} \setminus \{t\}} t'$ for all $t \in \mathcal{T}$
- and some technical conditions are fulfilled

Example (details omitted)

$\mathcal{T} = \{t_1, t_2, t_3\}$ with

- t_1: positive integers
- t_2: floats
- t_3: text

$$
\begin{aligned}
(\omega_1(t_1), \omega_2(t_1)) &= (1, 2) \\
(\omega_1(t_2), \omega_2(t_2)) &= (3.0, 4.0) \\
(\omega_1(t_3), \omega_2(t_3)) &= (a, b)
\end{aligned}
$$

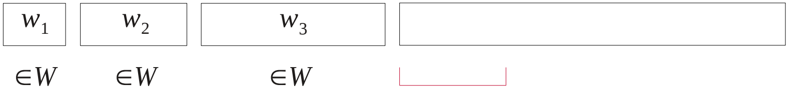
Learning Typed Pattern Languages (3)

To infer the pattern, use words that result from substitutions that replace all variables by their type witnesses.

Terminal-free patterns

The properties of a type witness allow us to **decompose** words into type witnesses in **polynomial time**:

- words are processed from left to right
- a prefix of the remainder of the word is matched to a type witness



Results (1)

Theorem

Let $\mathcal{T}$ be a finite set of decidable subsets of Σ^+ that has **a type witness**. Then the class of all non-erasing $\mathcal{T}$-typed pattern languages that are generated by **terminal-free patterns** is polynomially learnable from positive data.

Sketch of Algorithm

1. decompose words into type witnesses
2. select words with shortest decomposition
3. use decompositions to infer the pattern

Results (2)

Theorem

Let $\mathcal{T}$ be a finite set of decidable subsets of Σ^+ that has **a short type witness**. Then the class of all non-erasing $\mathcal{T}$-**typed pattern languages** is polynomially learnable from positive data.

More results

- some classes with **infinite** sets of types are polynomially learnable from positive data
- some classes of typed pattern languages are also polynomially learnable from positive data by a **consistent** learning algorithm