Computational Learning Theory
Learning Patterns (Monomials)

Akihiro Yamamoto 山本 章博
http://www.iip.ist.i.kyoto-u.ac.jp/member/akihiro/
akihiro@i.kyoto-u.ac.jp

Formal Languages

- Σ : a finite set of symbols and called an alphabet
- Σ^* : the set of all finite strings consisting of the symbols in Σ.
  - An empty string is denoted by ε.
  - $\Sigma^+ = \Sigma^* - \{\varepsilon\}$
- A formal language L on Σ is a subset of Σ^*.

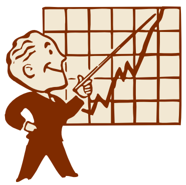
Example

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

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

$$L = \{aab, abb, aaab, aabb, abab, abbb, \ldots\}$$

Identification in the limit [Gold]

 $e_1, e_2, e_3, \ldots$ $\Rightarrow$  $\pi_1, \pi_2, \pi_3, \ldots$

- A learning algorithm A EX-identifies $L(\pi)$ in the limit from positive presentations if
  for any positive presentation $\sigma = e_1, e_2, e_3, \ldots$ of $L(\pi)$ and the output sequence $\pi_1, \pi_2, \pi_3, \ldots$ of A, there exists N such that for all $n > N$ $\pi_n = \pi'$ and $L(\pi') = L(\pi)$
- A learning algorithm A BC-identifies $L(\pi)$ in the limit from positive presentations if
  for any positive presentation $\sigma = e_1, e_2, e_3, \ldots$ of $L(\pi)$ and the output sequence $\pi_1, \pi_2, \pi_3, \ldots$ of A, there exists N such that for all $n > N$ $\cancel{\pi_n = \pi'}$ ~~and~~ $L(\pi_n) = L(\pi)$

Patterns (Monomials)

- Let X be a countable set of variables
  - Assuming $\Sigma \cap X = \varnothing$
- A pattern π is an element of $(\Sigma \cup X)^*$
  - That is, a pattern is a string consisting of symbols and variables.

Example

$\Sigma = \{\mathrm{a}, \mathrm{b}\}$, $X = \{x, y, \ldots\}$

axb, a$x$bb$y$a, aa$x$bybxa,...

- We sometime assume that every variable in a pattern is indexed, in the ordering of its first occurrence.

$\Sigma = \{\mathrm{a}, \mathrm{b}\}$, $X = \{x_1, x_2, x_3, \ldots\}$

ax_1b, a$x_1$bbx_2a, aax_1bx_2bx_1a,...

Defining languages with patterns

- A language defined with a pattern π is

$\{\sigma \mid \sigma = \pi\theta$ for some non-empty grounding substitution $\theta\}$

The language is denoted by $L(\pi)$.

Example

L(axb) = {aab, abb, aaab, aabb, abab, abbb,... }

L(ayb) = {aab, abb, aaab, aabb, abab, abbb,... }

L(bxaxb) = {baaab, bbabb,

baaaaab, babaabb,bbaabab bbbabbb,

baaaaaaab,...}

L(bxayb) = {baaab, baabb, baaaab, baaabb, baabab,...

bbaab, bbabb,bbaaab, bbaabb, bbabab,...

baaaab, baaabb,baaaaab, baaaabb,...

bbaaab, bbaabb,bbaaaab, bbaaabb,...}

Substitution (1)

- A substitution is a set of pairs

  $$\theta = \{\ (x_1, \tau_1),\ (x_2, \tau_2),\ \ldots,\ (x_n, \tau_n)\ \}$$

  where x_1, x_2, …, x_n are distinct variables and

  π_1, π_2, …, π_n are patterns.
- Applying a substitution θ to a pattern π is replacing every variable x_i in π with τ_i simultaneously.

  The result is denoted by $\pi\theta$.

Example

$$\theta_1 = \{\ (x, \mathrm{bba}),\ (y, \mathrm{ba})\ \}$$

$$\theta_2 = \{\ (x, \mathrm{b}y\mathrm{a}),\ (y, \mathrm{a}y\mathrm{b})\ \}$$

$$\mathrm{b}x\mathrm{a}x\mathrm{b}\,\theta_1 = \mathrm{bbbaabbab},\quad \mathrm{b}x\mathrm{a}x\mathrm{b}\,\theta_2 = \mathrm{bb}y\mathrm{aab}y\mathrm{ab},$$

$$\mathrm{a}x\mathrm{bb}y\mathrm{a}\,\theta_1 = \mathrm{abbabbbaa},\quad \mathrm{a}x\mathrm{bb}y\mathrm{a}\,\theta_2 = \mathrm{ab}y\mathrm{abba}y\mathrm{ba}$$

Substitution (2)

- A substitution θ={ $(x_1, \tau_1), (x_2, \tau_2), \ldots, (x_n, \tau_n)$ } is non-empty if all of $\tau_1, \tau_2, \ldots, \tau_n$ are in $(\Sigma \cup X)^+$.
- A substitution θ grounds a pattern π if $\pi\theta \in \Sigma^*$. Such θ is called a grounding substitution for π.
- A substitution θ={ $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ } is variable renaming if $y_1, y_2, \ldots, y_n$ are distinct varaibles.
  - We regard two patterns equivalent when each one is obtained from the other by renaming variables.

Examples

Two patterns axb and ayb are equivalent, and they are also equivalent to ax_1b.

Two patterns aa$x$bxybxa and aa$y$bxbya are equivalent, and they are also equivalent to aazb$w$bza and aax_1bx_2bx_1a.

Learning pattern languages

Example 1

C = {aab, abb, aaab, aabb,abab, abbb }

D = {a, b, bbbb, abba, baaaaba, babbb}

Example 2

C = {baaab, bbabb, baaaaab,babaabb, bbaabab}

D = {a, b, bbbb, abb, baaaaba, babbb}

The learning algorithm *learn-patterns*

- Fix an effective enumeration of patterns on $\Sigma \cup X$:

  $\pi_1, \pi_2, \ldots,$

$k = 1,\ \pi = \pi_1$

for $n = 1$ forever

  receive $e_n = \langle s_n, b_n \rangle$

  while ($0 \le \exists j \le n$

    $(e_j = \langle s_j, + \rangle$ and $s_j \notin L(\pi))$ and

    $(e_j = \langle s_j, - \rangle$ and $s_j \in L(\pi))$

      $\pi = \pi_k\ ;\ k{+}{+}$

  output π_k

The learning algorithm *learn-patterns*

- Fix an effective enumeration of patterns on $\Sigma \cup X$:

  $\pi_1, \pi_2, \ldots,$

$k = 1,\ \pi = \pi_1$

for $n = 1$ forever

  receive $e_n = \langle s_n, b_n \rangle$

  while ($0 \leq \exists j \leq n$

    $(e_j = \langle s_j, + \rangle$ and $s_j \notin L(\pi))$ and

    $(e_j = \langle s_j, - \rangle$ and $s_j \in L(\pi))$

      $\pi = \pi'$ for an appropriate π'; k++

  output π_k

Patterns v.s. Finite state automata

Patterns and FAs

- There does not always exist a FA M for a pattern π such that $L(M) = L(\pi)$.
- There does not always exist a pattern π for a FA M such that $L(M) = L(\pi)$.

A pattern π is regular if each variable in π occurs only once in π.

Example A pattern bxayb is regular, but bxaxb is not.

- For a regular pattern π there is a FA M such that $L(M) = L(\pi)$.

Regular Expressions (1)

- Mathematically, a regular expression is defined as a expression constructed of

  constants: **ε**, ∅, and **c** for every *c* in Σ

  operators : ▪, +, *

Examples Let Σ = {a, b}. Some examples of RE are:

abaa, **a** + **b**, **a***, (**ab**)*,

ε + **abaa** + **babb**, (**ab** + **ba**)*,

a((**a** + **b**)*)**b**, (**a** + **b**)* (**a** + **b**)

Regular Expressions (2)

- The language $L(E)$ represented by E is defined as

$$L(\boldsymbol{\varepsilon}) = \{\varepsilon\},\ \ L(\varnothing) = \varnothing, \text{ and } L(\mathbf{c}) = \{c\},$$

$$L(E\ F) = \{\ wv \mid w \in E \text{ and } v \in F\},$$

$$L(E + F) = L(\mathrm{E}) \cup L(\mathrm{F}),$$

$$L(E^*) = \{\ w^n \mid w \in E \text{ and } n \geq 0\}.$$

Examples Let $\Sigma = \{$a, b$\}$. Some examples of RE are:

$L(\boldsymbol{\varepsilon} + \mathbf{abaa} + \mathbf{babb}) = \{\varepsilon$, abaa, babb$\}$

$L((\mathbf{ab})^*) = \{\varepsilon$, ab, abab, ababab,…$\}$,

$L((\mathbf{ab} + \mathbf{ba})^*) = \{\varepsilon$, ab, ba, abab, abba, baab,baba,…$\}$,

$L(\mathbf{a}((\mathbf{a} + \mathbf{b})^*)\mathbf{b}) = \{$ab, aab, abb, aaab, aabb, …$\}$

$L((\mathbf{a} + \mathbf{b})^*\ (\mathbf{a} + \mathbf{b})) = \{$a, b, aa, ab, ba, bb, …$\}$

Regular Expressions and Patterns

- It can be proved that

  for every RE E there is a FA M s.t. $L(M)=L(E)$, and

  for every FA M there is a RE E s.t. $L(E)=L(M)$.

  - There does not always exist a RE E for a pattern π such that $L(E) = L(\pi)$.
  - There does not always exist a pattern π for a FA M such that $L(E) = L(\pi)$.

- For a regular pattern π, we can construct a RE E such that $L(E) = L(\pi)$, by replacing

  every symbol c in π with **c**, and

  every variable in π with $(\mathbf{c}_1+\ldots+\mathbf{c}_n)\,(\mathbf{c}_1+\ldots+\mathbf{c}_n)^*$.

Example $L(\mathbf{a}((\mathbf{a}+\mathbf{b})^*(\mathbf{a}+\mathbf{b})^*)\mathbf{b}) = L(\mathsf{a}x\mathsf{b})$

Learning from Positive Data

Learning from Positive Data

Example

C = {aab, abb, aaab, aabb,abab, abbb }

- In discussing learning from positive data, we have to define it mathematically, or some simple (trivial) solutions may always exist.
  - The learning algorithm which always return prefix tree automata.
  - The learning algorithm which always return the automaton accepting any strings.

Learning pattern languages

Example 1

C = {aab, abb, aaab, aabb,abab, abbb }

Example 2

C = {baaab, bbabb, baaaaab,babaabb,
bbaabab}

Learning Patterns from Positive Data

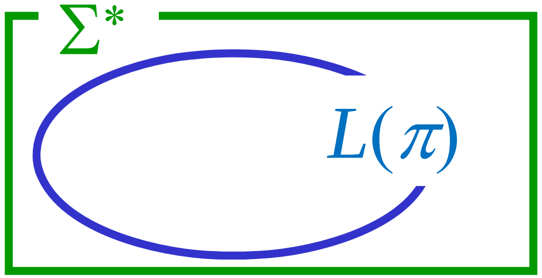
- ~~Fix an effective enumeration of patterns on $\Sigma \cup X$:~~
  ~~$\pi_1, \pi_2, \ldots,$~~

$k = 1,\ \pi = \pi_1$
for $n = 1$ forever
  receive $e_n = \langle\, s_n\,, b_n \,\rangle$
  while ($0 \le \exists\, j \le n$
    ($e_j = \langle\, s_j\,, + \,\rangle$ and $s_j \notin L(\pi)$) ~~and~~
    ~~($e_j = \langle\, s_j\,, - \,\rangle$ and $s_j \in L(\pi)$)~~
      $\pi = \pi'$ for an appropriate π'; k ++
  output π

Positive Presentations

$e_1, e_2, e_3, \ldots$

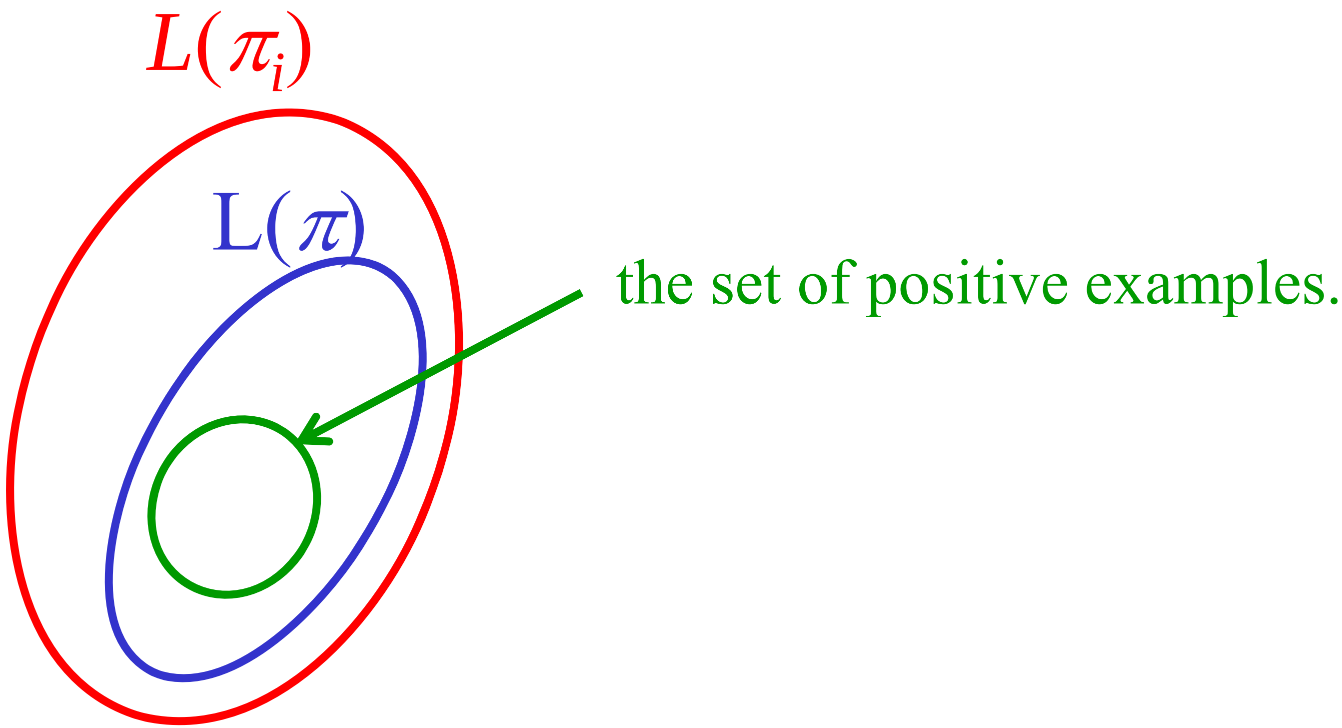
$\pi_1, \pi_2, \pi_3, \ldots$



- A presentation of $L(\pi)$ is a infinite sequence consisting of positive and negative example.
- A presentation σ is positive if σ consists only of positive example $< s, + >$ and any positive example occurs at least once in σ.

Which patterns should be chosen?

- Intuitively, choose a minimal language which contains all of the positive examples at the moment.
  - That is, avoid over-generalization!



Analysis of Patterns (1)

Lemma 1 For every string s, there are only finite number of pattern languages containing s.

Proof. If $s \in L(\pi)$, then $|s| \geq |\pi|$.

Example The languages containing s = aab are

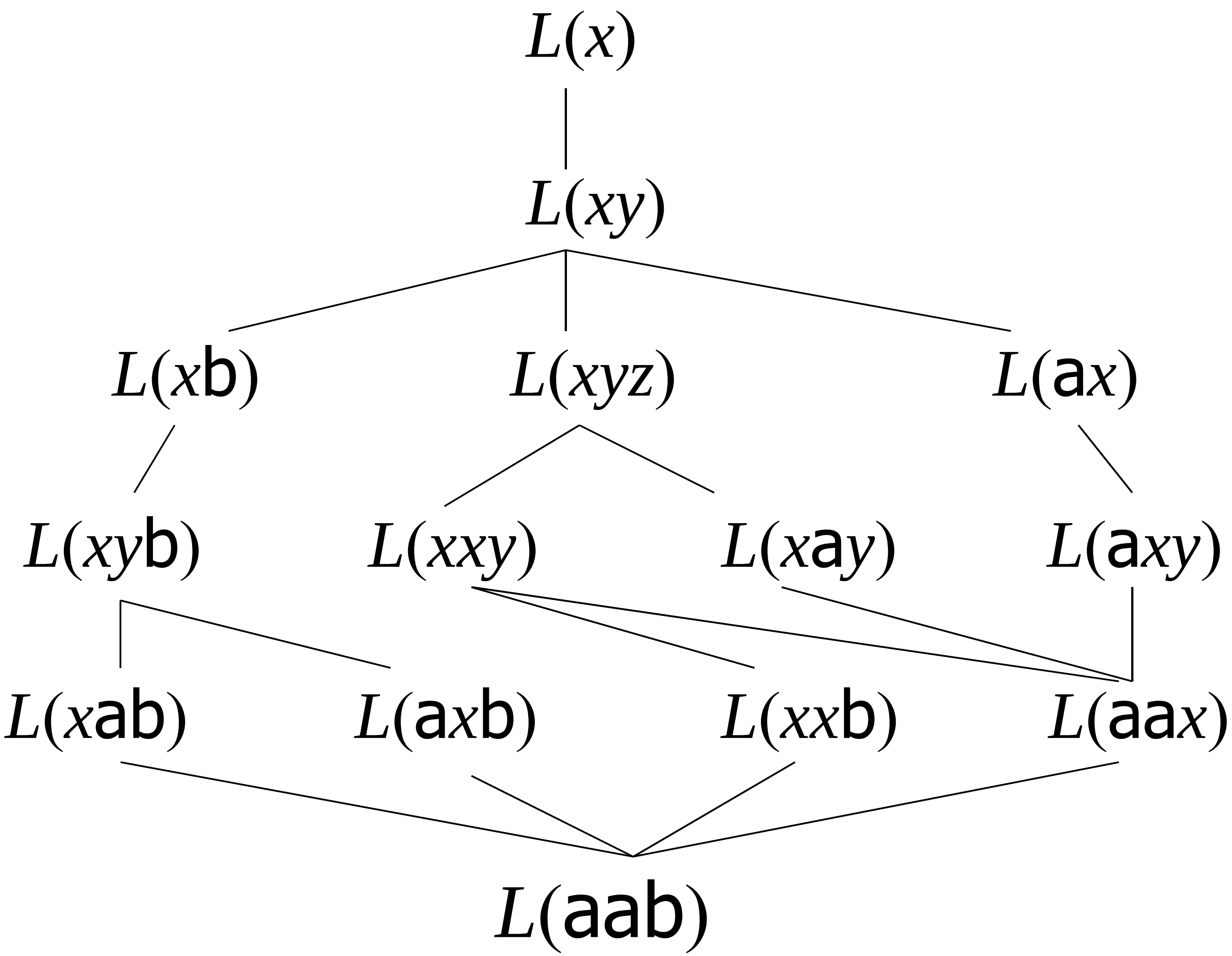
L(aab),

$L(x\text{ab})$, $L(\text{a}x\text{b})$, $L(\text{aa}x)$, $L(xx\text{b})$, $L(x\text{b})$, $L(\text{a}x)$, $L(x)$,

$L(xy\text{b})$, $L(x\text{a}y)$, $L(\text{a}xy)$, $L(xxy)$, $\mathrm{L}(xy)$,

$L(xyz)$,

Hasse Diagram



Analysis of Patterns (2)

Example π = axxbb$y$aa

L(axxbb$y$aa)

={aaabbaaa, aaabbbaa, abbbbaaa, abbbbbaa,

aaaaabbaaa, aaaaabbbaa, aababbbaaa,

aababbbbaa,…, aabaaabaabbbbbababaa,…}

- Using examples as long as π :

  aaabbaaa, aaabbbaa, abbbbaaa, abbbbbaa

θ_1={(x,a), (y,a)} θ_2={(x,a), (y,b)} θ_3={(x,b), (y,a)} θ_4={(x,a), (y,b)}

We can know that the 2nd, 3rd, and 6th positions must be variables.

The variable at the 6th position is different from those at the 2nd and 3rd.

Analysis of Patterns (3)

- Any language $L(\pi')$ containing the four strings must be a superset of $L(\pi)$.

aaabbaaa, aaabbbaa, abbbbaaa, abbbbbaa

$\theta_1=\{(x,\mathrm{a}), (y,\mathrm{a})\}$ $\theta_2=\{(x,\mathrm{a}), (y,\mathrm{b})\}$ $\theta_3=\{(x,\mathrm{b}), (y,\mathrm{a})\}$ $\theta_4=\{(x,\mathrm{a}), (y,\mathrm{b})\}$

  - If π' and π are of same length, π' has more variables than π.
  - If π' is shorter than π, π' has at least one variable with which some substring of π longer than 2 must be replaced.

Characteristic Set of $L(\pi)$

- Let π be a pattern which contains variables $x_1, x_2, \ldots, x_n$. Consider the following substitutions:

$$\theta_{\mathrm{a}} = \{(x_1, \mathrm{a}), (x_2, \mathrm{a}), \ldots, (x_n, \mathrm{a})\},$$
$$\theta_{\mathrm{b}} = \{(x_1, \mathrm{b}), (x_2, \mathrm{b}), \ldots, (x_n, \mathrm{b})\},$$
$$\sigma_1 = \{(x_1, \mathrm{a}), (x_2, \mathrm{b}), \ldots, (x_n, \mathrm{b})\},$$
$$\ldots$$
$$\sigma_n = \{(x_1, \mathrm{b}), (x_2, \mathrm{b}), \ldots, (x_n, \mathrm{a})\}$$

- The set $\{p\theta_{\mathrm{a}}, p\theta_{\mathrm{b}}, p\sigma_1, \ldots, p\sigma_n\}$ is a characteristic set of $L(\pi)$.

Anti-Unifcation of Strings

- For a set C of stings of same length

$$\begin{aligned} s_1 &= c_{11}\,c_{12}\ldots \boxed{c_{1i}} \ldots c_{1k} \\ s_2 &= c_{21}\,c_{22}\ldots c_{2i} \ldots c_{2k} \\ &\ldots \\ s_n &= c_{n1}\,c_{n2}\ldots c_{nj} \ldots c_{nk} \end{aligned}$$

the anti-unification of C is a pattern

$$\pi = \gamma(c_{11}c_{21}\ldots c_{n1})\gamma(c_{12}c_{22}\ldots c_{n2}) \ldots \gamma(c_{1k}c_{2k}\ldots c_{nk})$$

where

$$\gamma(c_1c_2\ldots c_n) = \begin{cases} c & \text{if } c_1 = c_2 = \ldots = c_n = c \\ x_{\iota(c1c2\ldots cn)} & \text{otherwise.} \end{cases}$$

and $\iota(c_1c_2\ldots c_n)$ is the "index" of $c_1c_2\ldots c_n$.

In Theoretical Form

Lemma 2 Let $\pi_1, \pi_2, \ldots, \pi_n$ be patterns. If the language $L(\pi_k)$ is minimal in $\{L(\pi_1), L(\pi_2), \ldots, L(\pi_n)\}$, then π_k is one of the longest patterns in the list.

Lemma 3 Let π_1 and π_2 be patterns of same length. Then $L(\pi_1) \subseteq L(\pi_2)$ if and only if $\pi_2\theta = \pi_1$.

Note If we do not assume π_1 and π_2 be patterns of same length, then it is not decidable whether or not

$L(\pi_1) \subseteq L(\pi_2)$.

Which pattern should be chosen?

- Let C be a set of (positive) examples

1. Select all shortest examples.
2. Look for one of the minimal patterns between x (a singleton variable) and the anti-unifier of the shortest examples, and return it.

Note: If we only follow the identification-in-the-limit criterion, the second can be simplified as

2'. Return the anti-unifier of the shortest examples but this might not seem "learning".

Positive and Negative examples

 $e_1, e_2, e_3, \ldots$ 

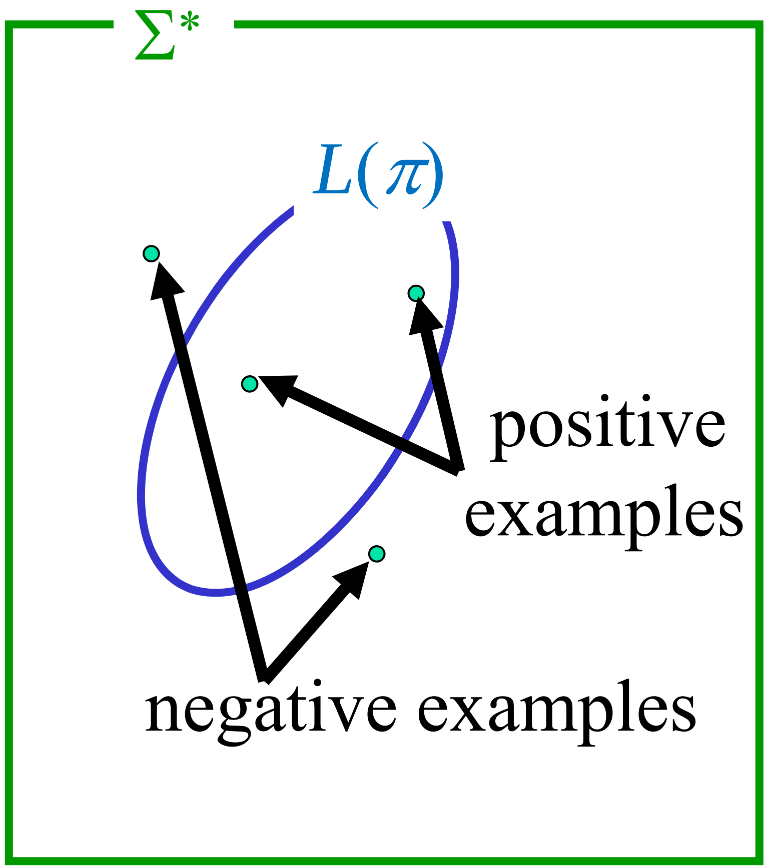
$L(\pi)$: a language represented with a pattern π

- a positive example on $L(\pi)$:

  $< s, +>$ for $x \in L(\pi)$

  a negative example on $L(\pi)$:

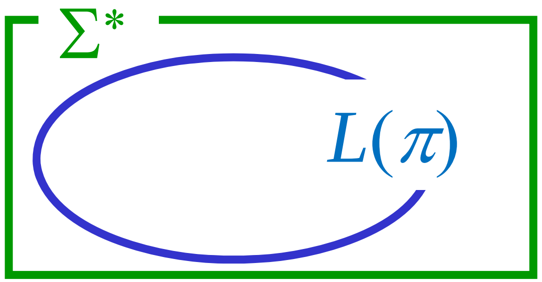
  $< s, ->$ for $x \in \overline{L(\pi)}$



Positive presentations

$e_1, e_2, e_3, \ldots$ $\Rightarrow$

$\pi_1, \pi_2, \pi_3, \ldots$



- A presentation of $L(\pi)$ is a infinite sequence consisting of positive and negative example.
- A presentation σ is positive if σ consists only of positive example $< s, +>$ and any positive example occurs at least once in σ.

Identification in the limit [Gold]

 $e_1, e_2, e_3, \ldots$ →  $\pi_1, \pi_2, \pi_3, \ldots$

- A learning algorithm A EX-identifies $L(\pi)$ in the limit from positive presentations if
  for any positive presentation $\sigma = e_1, e_2, e_3, \ldots$ of $L(\pi)$ and the output sequence $\pi_1, \pi_2, \pi_3, \ldots$ of A, there exists N such that for all $n > N$ $\pi_n = \pi'$ and $L(\pi') = L(\pi)$
- A learning algorithm A BC-identifies $L(\pi)$ in the limit from positive presentations if
  for any positive presentation $\sigma = e_1, e_2, e_3, \ldots$ of $L(\pi)$ and the output sequence $\pi_1, \pi_2, \pi_3, \ldots$ of A, there exists N such that for all $n > N$ $\cancel{\pi_n = \pi'}$ ~~and~~ $L(\pi_n) = L(\pi)$

Identification in the limit [Gold]

- A learning algorithm *A* EX-identifies a class C of languages in the limit from psoitive presentations if *A* EX-identifies every language in C in the limit from positive presentations.

- A learning algorithm *A* BC-identifies a class C of languages in the limit from positive presentations if *A* BC-identifies every language in C in the limit from positive presentations.

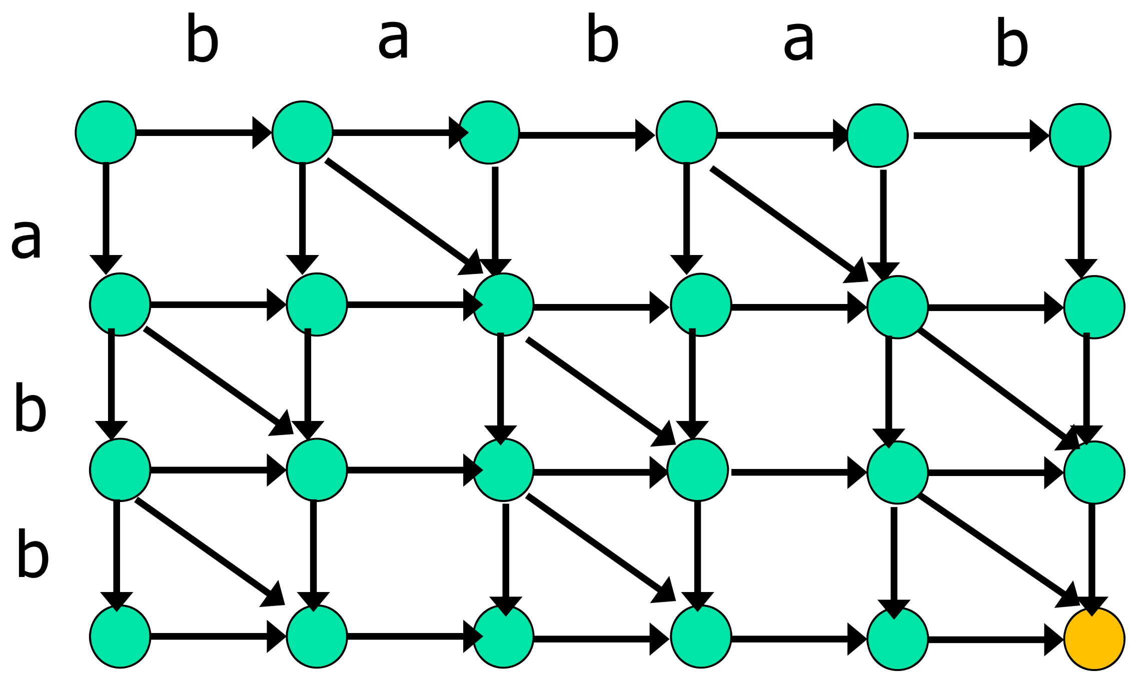
Identification of patterns

Theorem The revised algorithm of *Learn-pattern* with the minimal language strategy EX-identifies the class of all pattern languages in the limit from positive presentations.

- The minimal language strategy means that when revising conjecture π a pattern generating a minimal language for positive data is chosen as the "appropriate" pattern.

Linear Patterns

- When we are learning only linear patterns, the shortest linear patterns can be found by using the dynamic programming.
  - The algorithm is a modification of that for finding LCS "longest common subsequences" or edit distance.



A Negative Result

.Theorem [Gold] There is no learning algorithm which identifies any regular language from positive data.

- Note that a regular language is a formal language which is accepted by a finite state automaton. It is also represented in a regular expression.

Theorem [Gold] There is no learning algorithm which identifies any regular expression from positive data.

A Negative Result (2)

- We construct a positive presentation σ of L in the following manner.
- Let e_1 be a string in L. Since the set $\{e_1\}$ is also in C and A must identify $\{e_1\}$. So the first N_1 examples of σ are all E_1 , until "A identifies $\{e_1\}$."

$$\exists N_1 \ \forall\ n > N_1 \ h_n = g_1 \text{ and } L(g_1) = \{e_1\}$$

$$\underbrace{e_1, e_1, \ldots, e}_{N_1+1}{}_2, \ldots$$

$$h_1, h_2, h_3, \ldots, g_1, g_1, \ldots$$

A Negative Result (3)

- Let the (N_1+1)-th example be e_2 which is different from e_1 .
- Since C contains $\{e_1, e_2\}$, the learning algorithm A identifies $\{e_1, e_2\}$ in the limit.

$$\exists N_1 \ \forall \ n > N_2 > N_1 \ g_n = g_2 \text{ and } \{e_1, e_2\}$$

$$\underbrace{\underbrace{e_1, e_1, \ldots}_{N_1+1} \quad e_2, \ldots,}_{N_2+1} \quad e_3, \ldots$$

$h_1, h_2, \ldots, g_1, \ldots, g_2, \ldots,$

A Negative Result (4)

- Let the (N_2+1)-th example be e_3 which is different from both of e_1 or e_2.
- Since C contains $\{e_1, e_2, e_3\}$, A identifies $\{e_1, e_2, e_3\}$ in the limit.

$$\exists N_3 \ \forall n > N_3 > N_2 > N_1 \ h_n = g_3 \text{ and } L(g_3)=\{E_1, E_2, E_3\}$$

- The language $L = \{e_1, e_2, e_3, e_4, \ldots\}$ is a infinite and A cannot identify L.

Reference

- M. Gold : Language Identification in the Limit, Information and Control, 10, 447-474 (1967).

- D. Angulin : Inductive Inference of Formal Languages from Positive Data, Information and Control, 45, 117-135 (1980).

Defining languages with patterns

- A language defined with a pattern π is

$\{\sigma \mid \sigma = \pi\theta$ for some non-empty grounding substitution $\theta\}$

The language is denoted by L(π).

Example

L(axb) = {aab, abb, aaab, aabb, abab, abbb,... }

L(ayb) = {aab, abb, aaab, aabb, abab, abbb,... }

L(bxaxb) = {baaab, bbabb,
baaaaab, babaabb,bbaabab bbbabbb,
baaaaaaab,...}

L(bxayb) = {baaab, bbabb, baabb, bbaab,
baaaab, baaabb,baabab, babbbb,
bbaaab, bbaabb,bbabab, bbbbbb,...}