Computational Learning Theory
Regular Expression vs. Monomilas

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

Contents

- What about a regular expressions?
- Learning in the Limit
- General Theory of Learning from Positive Data

What About Regular Expressions?

Regular Expressions (1)

- Regular expression was invented by S. Kleene, a mathematician, to represent sets in mathematics.
- Some interfaces of operating systems employ regular expressions in order to sets of files, etc.
  - Such an interface is sometimes called a “shell” in UNIX-orgined operating systems, e.g. Ubuntu.
  - Regular expressions can be used in the command window (prompt) of MS Window systems.

```
$ ls *.c
$ ls [abc]*.c
```

Regular Expressions (2)

- Some commands in UNIX-origned operating systems also employ regular expressions in order to represent patterns of strings.
  - Examples of such editors are `ed`, `sed`, `vi`, `more`, …
- Some programming languages based on manipulating characters and strings are employ regular expressions.
  - Examples of such languages are awk, perl, python,....

```
import re regex = r’ ab+’
text = "abbabbabaaabb"
pattern = re.compile(regex) matchObj =
pattern.match(text)
```

Regular Expressions (3)

- From the history of their usage, so many variations and modifications are invented and introduced into particular commands or languages.
- The simplest regular expressions are constructed of characters a, b, c and operations ･, |, and *, where a string w of characters represents the set $\{w\}$ and
  - $(R \mid S)$ represents the union $R \cup S$
  - (RS) represents the set of catenations

    $$\{wv \mid w \in R \text{ and } v \in S\}$$
  - (R^*) represents the set of Kleene Closure of R

Kleene Closure

- $L^0 = \{\varepsilon\}$

  $L^n = L\,L^n = \{\, uv \mid u \in L, v \in L^n \,\}\ (n \geq 1)$

  $L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup \ldots = \bigcup_{n=1}^{\infty} L^n$

  - Sometimes the set $L^* - \{\varepsilon\}$ is denoted by L^+.

Example

$L = \{$aa, ab$\}$

$L^2 = \{$aaaa, aaab, abaa, abab$\}$

$L^3 = \{$aaaaaa, aaaaab, aaabaa, aaabab, abaaaa,...$\}$

$\ldots$

$L^* = \{\varepsilon$, aa, ab, aaaa, aaab, abaa, abab, aaaaaa,...$\}$

Examples

Regular Expression R	Set of strings $L(R)$
aababb	$\{$aababb$\}$
(aab)|(abb)	$\{$aab, abb$\}$
a(ab)* b	$\{w \mid w =$ aub and $u \in \{$ab$\}^*\}$ $= \{$ab, aabb, aababb,$\ldots\}$
a(a | b)* b)	$\{w \mid w =$ aub and $u \in \{$a, b$\}^*\}$ $= \{$ab, aab, abb, aaab, aabb ,$\ldots\}$
a(aa | bb)* b)	$\{w \mid w=$ aub and $u \in \{\{$aa$\} \cup \{$ab$\}\}^*\}$ $= \{$aaab, aabb, aaaaab, aaaabb, aabaab, aababb, aaaaaaab,$\ldots\}$

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, baabb, baaaab, baaabb, baabab,...
bbaab, bbabb,bbaaab, bbaabb, bbabab,...
baaaab, baaabb,baaaaab, baaaabb,...
bbaaab, bbaabb,bbaaaab, bbaaabb,...}

RE vs. Monomials in Learning

- While both regular expressions and monomials represents data set of strings, they are different when we treat them in machine learning.
- Assume the case that an *unknown* (*hidden*) *representation R* is learned from training examples *in the limit*.
  - If we adopt a regular expression to represent *R*, we cannot learn *R* only from only positive examples, i.e. unsupervised learning.
  - If we adopt a monomial to represent *R*, we can learn *R* only from only positive examples.

Learning in the Limit

Examples on $L(R)$

- We assume that, for an unknown rule R_*,

C_* is a finite set of positive examples on $L(R_*)$ and
D_* is a finite set of negative examples on $L(R_*)$.

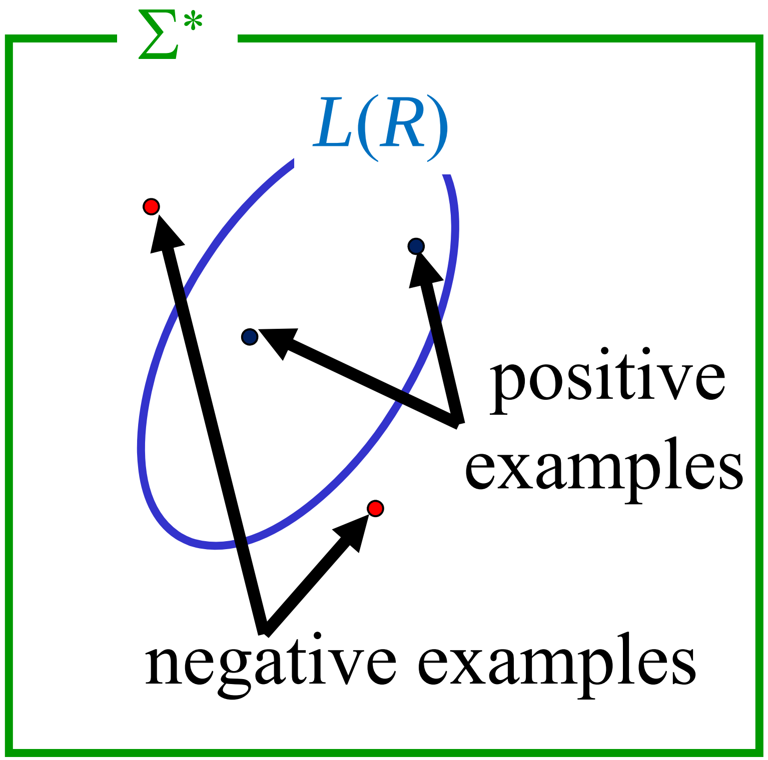
$L(R)$: the set represented by the representation R

- a positive example on $L(R)$:

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

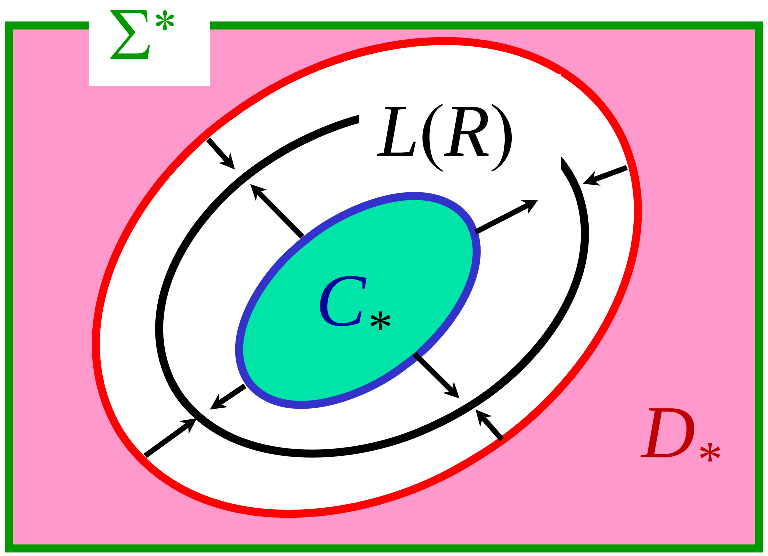
- a negative example on $L(R)$:

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



Question

- If we give more and more (negative and positive) examples on $L(R_*)$ to an learning algorithm, does it eventually conjecture the unknown R_*?
- We have to give mathematical definitions of
  - giving more and more examples, and
    - or giving examples many enough
  - conjecturing M eventually.



$R\wedge \rightarrow R$

Assumption

- Without loss of generality, we may assume that learning algorithm takes examples in C_* and D_* one by one.
- In the situation that both C_i and D_i grow, we assume that an infinite sequence σ of strings marked with either + or –, and some truncation of σ corresponds to C_i and D_i.

Example

σ : <ab,+>, <aab,+>, <bbb,–>, <aaab,+>, <abba,–>, …

C_i = {ab, aab, aaab},

D_i = {bbb, abba}.

Presentations

Definition A presentation of $L(R)$ is a infinite sequence

σ: $< s_0, p_0 >, < s_1, p_1 >, < s_2, p_2 >, \ldots$

where $s_i \in \Sigma^*$ and $p_i = +$ or $-$.

- $< s, +>$ is a positive example
- $< s, ->$ is a negative example

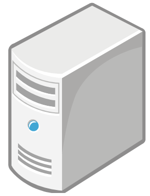
- $\sigma[n] = < s_0, p_0 >, < s_1, p_1 >, < s_2, p_2 >, \ldots, < s_{n-1}, p_{n-1} >$

Definition A presentation σ is complete if

any $x \in L(R)$ appears in σ as a positive example $< x, +>$ at least once and

any $x \in \overline{L(R)}$ appears in σ as a negative example $< x, ->$ at least once.

Identification in the limit [Gold]

 $x_1, x_2, x_3, \ldots$  $R_1, R_2, R_3, \ldots$

- A learning algorithm A EX-identifies $L(R)$ in the limit from complete presentations if for any complete presentation $\sigma = x_1, x_2, x_3, \ldots$ of $L(R)$ and the output sequence $R_1, R_2, R_3, \ldots$ of A, there exists N such that for all $n \geq N$ $R_n = R'$ and $L(R') = L(R)$
- A learning algorithm A BC-identifies $L(R)$ in the limit from complete presentations if for any complete presentation $\sigma = x_1, x_2, x_3, \ldots$ of $L(R)$ and the output sequence $R_1, R_2, R_3, \ldots$ of A, there exists N such that for all $n \geq N$ $\cancel{R_n = R' \text{ and}}$ $L(R_n) = L(R)$

A Well-known Result on RE

Theorem For every set $L(R)$ represented by a regular expression R, there exists a unique minimal expression R' such that $L(R)=L(R')$.

Embedding the Modified Generate-and-Test Algorithm into the Framework

Assume a procedure of enumerating all RE so that the enumeration $R_0, R_1, R_2, \ldots, R_i, \ldots$ satisfies

$$|R_0| \leq |R_1| \leq |R_2| \leq \ldots \leq |R_i| \leq \ldots$$

Input $\sigma = x_1, x_2, \ldots$: presentation (an infinite sequence)

Initialize $k = 0$ /* R_0 is the simplest RE */

for $N = 1,2,\ldots$

  $\sigma[N] = x_1, x_2, \ldots, x_N$

  forever

   let $k' = k$

   for $n = 1,2,\ldots, N$,

    if $(x_n \in C$ and $x_n \notin L(R_{k'}))$ or $(x_n \in D$ and $x_n \in L(R_{k'}))$

     replace k with $k + 1$

   if $k' = k$

    terminate and output R_k

On the Generate-and-Test Algorithm

Theorem For any regular expression R_*,

the modified generate-and-test algorithm EX-identifies $L(R_*)$ in the limit from complete presentations.

Proof Let σ be an any complete presentation on $L(R_*)$.

Let R_N be the output of the algorithm for the input $\sigma[N]$.

If $L(R_*) \neq L(R_N)$, then there must be a string $x \in \Sigma^*$

$(x \in L(R_*)$ and $x \notin L(R_N))$ or $(x \notin L(R_*)$ and $x \in L(R_N))$.

Since σ is complete, x must be appears in the sequence with the sign + if $x \in L(R)$ or otherwise with – .

This means that R_N must be replaced with another expression, at latest, when x appears in σ.

Once the algorithm outputs R_N s.t. $L(R_*) = L(R_N)$, it never changes the output afterwards.

Revised version of *learn-patterns*

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

  $\cancel{\pi_1, \pi_2, \ldots,}$

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

for $n = 1$ forever

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

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

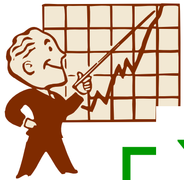
        $(e_j = \langle s_j, + \rangle$ and $s_j \notin L(\pi))$ ~~and~~

        $\cancel{(e_j = \langle s_j, - \rangle \text{ and } s_j \in L(\pi))}$

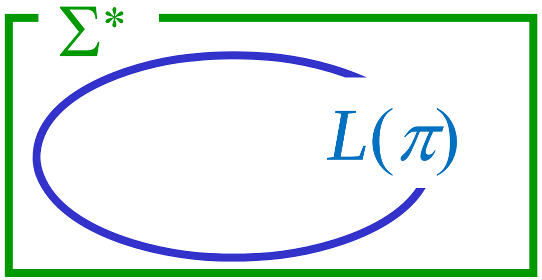
            $\pi = \pi'$ for an appropriate π'; k ++

    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.

Identification in the limit [Gold]

$s_1, s_2, s_3, \ldots$

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

- A learning algorithm A EX-identifies $L(\pi)$ in the limit from positive presentations if
  for any positive presentation $\sigma = s_1, s_2, s_3, \ldots$ of $L(g)$ 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 $\sigma = s_1, s_2, s_3, \ldots$ of $L(g)$ 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~~ $L(\pi_n) = L(\pi)$

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.

Anti-Unifcation of Strings

- For a set C of stings of same length

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

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 $c_1c_2\ldots c_n$.

Identification of patterns

Theorem The revised algorithm of *Learn-pattern* with computing an anti-unification EX-identifies the class of all pattern languages in the limit from positive presentations.

A Negative Result

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((ab)*)in the following manner.
- Let e_1 be a string in L. Since the regular expression 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 the regular expression e_1."

$$\exists N_1 \ \forall \ n > N_1 \ h_n = e_1$$

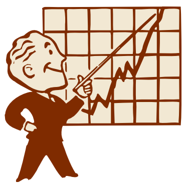
$\underbrace{e_1, e_1, \ldots, e}_{N_1+1} {}_2, \ldots$

$h_1, h_2, h_3, \ldots, e_1, e_1, \ldots$

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.

$$\exists N_1 \ \forall \ n > N_2 > N_1 \ R_n = e_1 | e_2$$



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

$h_1, h_2, \ldots, e_1 | e_2 , \ldots,$
$e_1 | e_2 , \ldots,$

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.

$$\exists N_3 \ \forall n > N_3 > N_2 > N_1 \ h_n = e_1 | e_2 | e_3$$

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

General Theory of Learning from Positive Data

GCD and Learning

A class of languages in $\mathbf{N}$:

$$\mathsf{L}(\mathbf{N}) = \{L(m) \mid m \in \mathbf{N}\}$$

$$L(m) = \{0\underbrace{1\ldots1}_{n}0 \mid n \bmod m = 0\}$$

$$L(m) = \{n \in \mathbf{N} \mid n \bmod m = 0\}$$

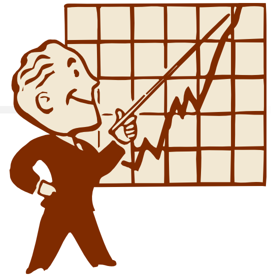
A class of languages in $\mathbf{Z}$:

$$\mathsf{L}(\mathbf{N}) = \{L(m) \mid m \in \mathbf{N}\}$$

$$L(m) = \{\underbrace{1\ldots1}_{n} \mid n \bmod m = 0\} \cup \{0\underbrace{1\ldots1}_{n} \mid n \bmod m = 0\}$$

$$L(m) = \{n \in \mathbf{Z} \mid |n| \bmod m = 0\}$$

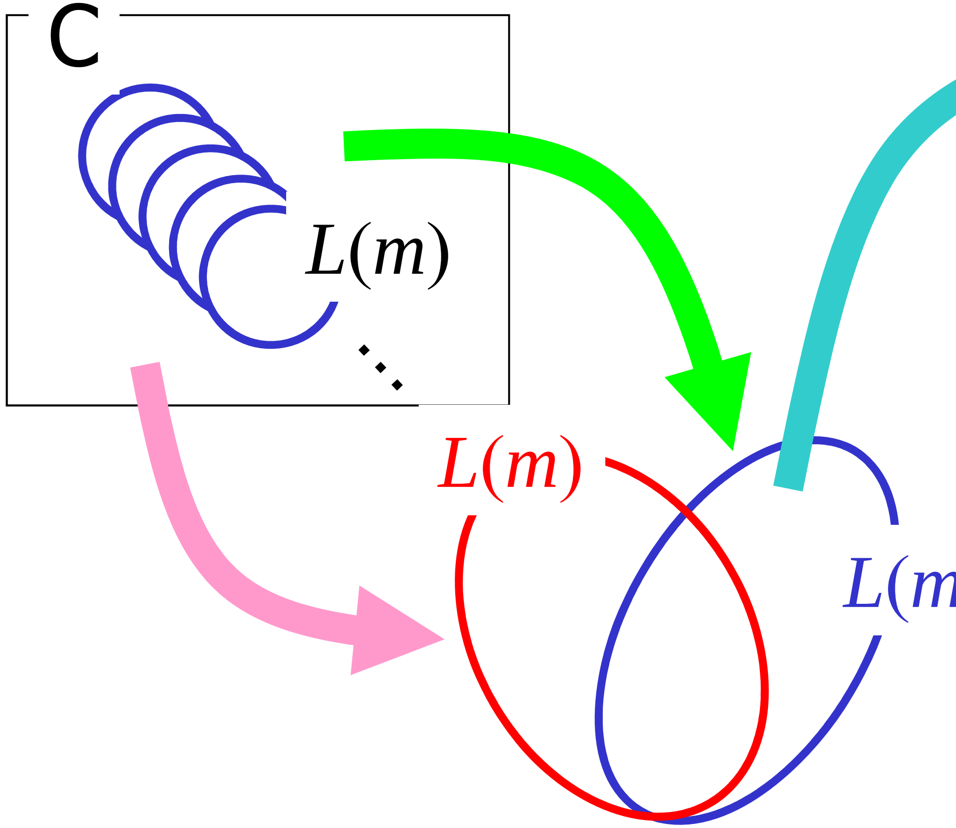
GCD and Learning



- L($\mathbf{N}$) = $\{L(m) \mid m \in \mathbf{N}\}$
  $L(m) = \{01\ldots10 \mid n \bmod m = 0\}$

Positive presentation

72, 48, 60, …,12,…



Compute the GCD of $s_1, s_2, \ldots, s_k$ with Euclidean Algorithm

Conjecture

72, 24, 12,…,12,…

Proving that L($\mathbf{N}$) is identifiable

- For every $n \in \mathbf{N}$, the characteristic set of $L(m)$ in L($\mathbf{N}$) is $\{ m \}$, that is, $\{ m \} \subseteq L(m')$ implies $L(m) \subseteq L(m')$.

- To see this, assume that $\{ m \} \subseteq L(m')$.

  This is equivalent to $m \in L(m')$ and from the definition of $L(m')$, $m = k' m'$ for some $k' \in \mathbf{N}$ ($\mathbf{Z}$).
  - $L(m) = \{n \in \mathbf{N} \mid n \bmod m = 0\}$ ($\{n \in \mathbf{Z} \mid |n| \bmod m = 0\}$).

  Let n be any element in $L(m)$. Then, from the definition, there exists $k \in \mathbf{N}$ ($\mathbf{Z}$) such that $n = k\, m$. For the k' and k, it holds that $n = k\, k'\, m'$. This means $n \in L(m')$, and therefore $L(m) \subseteq L(m')$.

Analysis of Patterns (1)

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 (2)

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

aaabbaaa, aaabbbaa, abbbbaaa, abbbbbaa

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

  - 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:

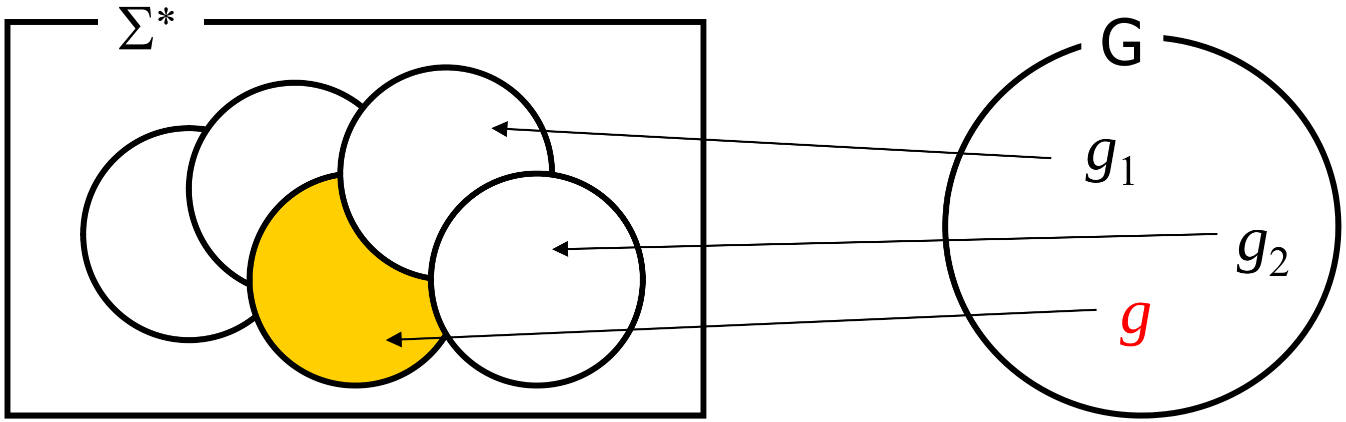
  $$\theta_{\mathrm{a}} = \{(x_1, \mathrm{a}), (x_2, \mathrm{a}), \ldots, (x_n, \mathrm{a})\},$$
  $$\theta_{\mathrm{b}} = \{(x_1, \mathrm{b}), (x_2, \mathrm{b}), \ldots, (x_n, \mathrm{b})\},$$
  $$\sigma_1 = \{(x_1, \mathrm{a}), (x_2, \mathrm{b}), \ldots, (x_n, \mathrm{b})\},$$
  $$\ldots$$
  $$\sigma_n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

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

A General Framework of Learning

- A class of formal languages L(G) indexed with G
- G: A set of expressions such that each expression in G represents one language in L(G), and every language in L(G) is represented by at least one expression in G.
  - We assume that There is an algorithm which determines whether or not $w \in L(g)$ for every string $w \in \Sigma^*$ and g.

Examples of G : a set of finite state automata, a set of CFGs, a set of patterns,…



C2: The Characteristic Set Property

- A subset $C(g)$ of a language of $L(g)$ is a characteristic set of $L(g)$ in $\mathsf{L}(\mathsf{G})$ if

  (1) $C(g)$ is a finite set and

  (2) for every $L(g') \in \mathsf{L}(\mathsf{G})$ $C(g) \subseteq L(g')$ implies

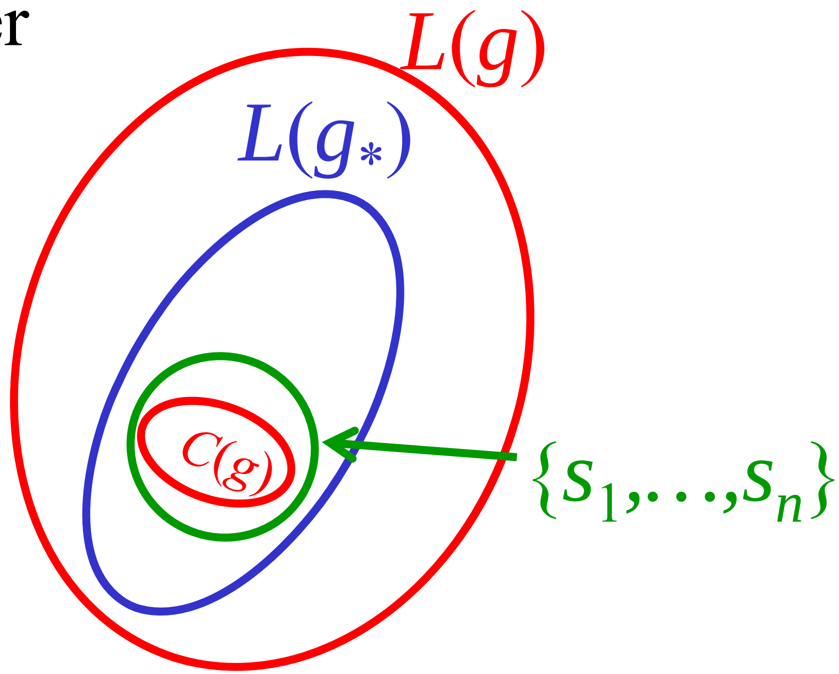
  $$L(g) \subseteq L(g')$$

Theorem [Kobayashi] A class $\mathsf{L}(\mathsf{G})$ of languages is identifiable in the limit from positive presentation if every language $L(g)$ in $\mathsf{L}(\mathsf{G})$ has a characteristic set $C(g)$ in $\mathsf{L}(\mathsf{G})$.

Which grammar should be chosen?

- Choose g such that $C(g) \subseteq \{s_1, \ldots, s_n\}$
  - The examples are from $L(g_*)$, that is, $\{s_1, \ldots, s_n\} \subseteq L(g_*)$. and therefore $C(g) \subseteq L(g_*)$. From the definition of characteristic sets, this implies $L(g) \subseteq L(g_*)$.

    So over generalization never happens.



EC1: The Finite Tell-tale Property

- A subset $T(g)$ of a language of $L(g)$ is a finite tell-tale of $L(g)$ in **L(G)** if

  (1) $T(g)$ is a finite set and

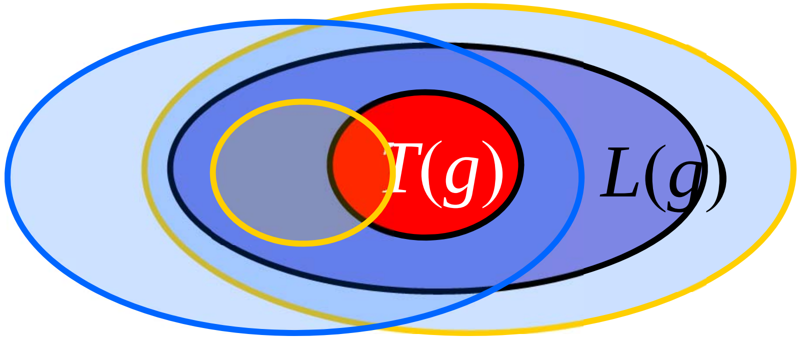
  (2) $T(g) \subseteq L(g') \subsetneq L(g)$ for no $L(g') \in$ **L(G)** other than $L(g)$

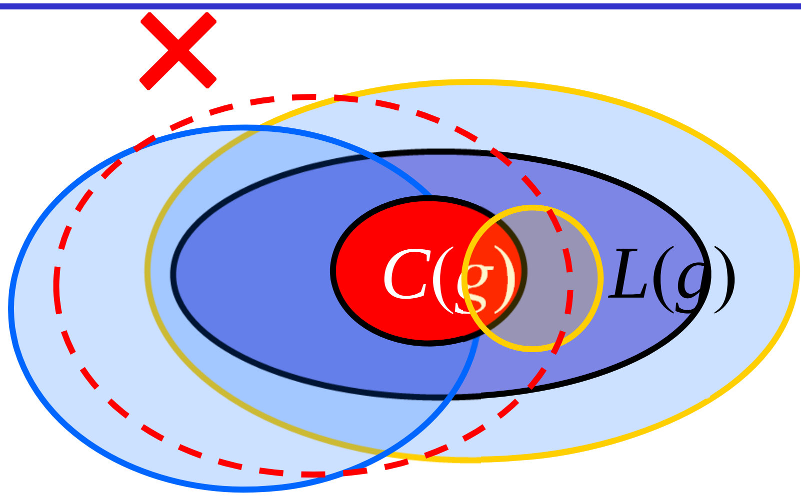
Theorem [Angluin] A class **L(G)** of languages is identifiable in the limit from positive presentation if and only if every language $L(g)$ in **L(G)** has a finite tell-tail $T(g)$ in **L(G)** and there is a procedure which generates elements of $T(g)$ when the grammar g is given as an input.

Tell-tales and Characteristic Sets

Finite Tell-tale $T(g)$ of $L(g)$:

- $T(g) \subseteq L(g)$ (T is a finite set)
- For no $L(g')\in$ L(G) other than $L(g')$, $T(g)\subseteq L(g')\subsetneq L(g)$





Characteristic set $C(g)$ of $L(g)$:

- $T(g) \subseteq L(g)$ (T is a finite set)
- For every $L(g')\in$ L(G)

$C(g)\subseteq L(g')$ implies $L(g) \subseteq L(g')$

Analysis of Patterns (3)

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

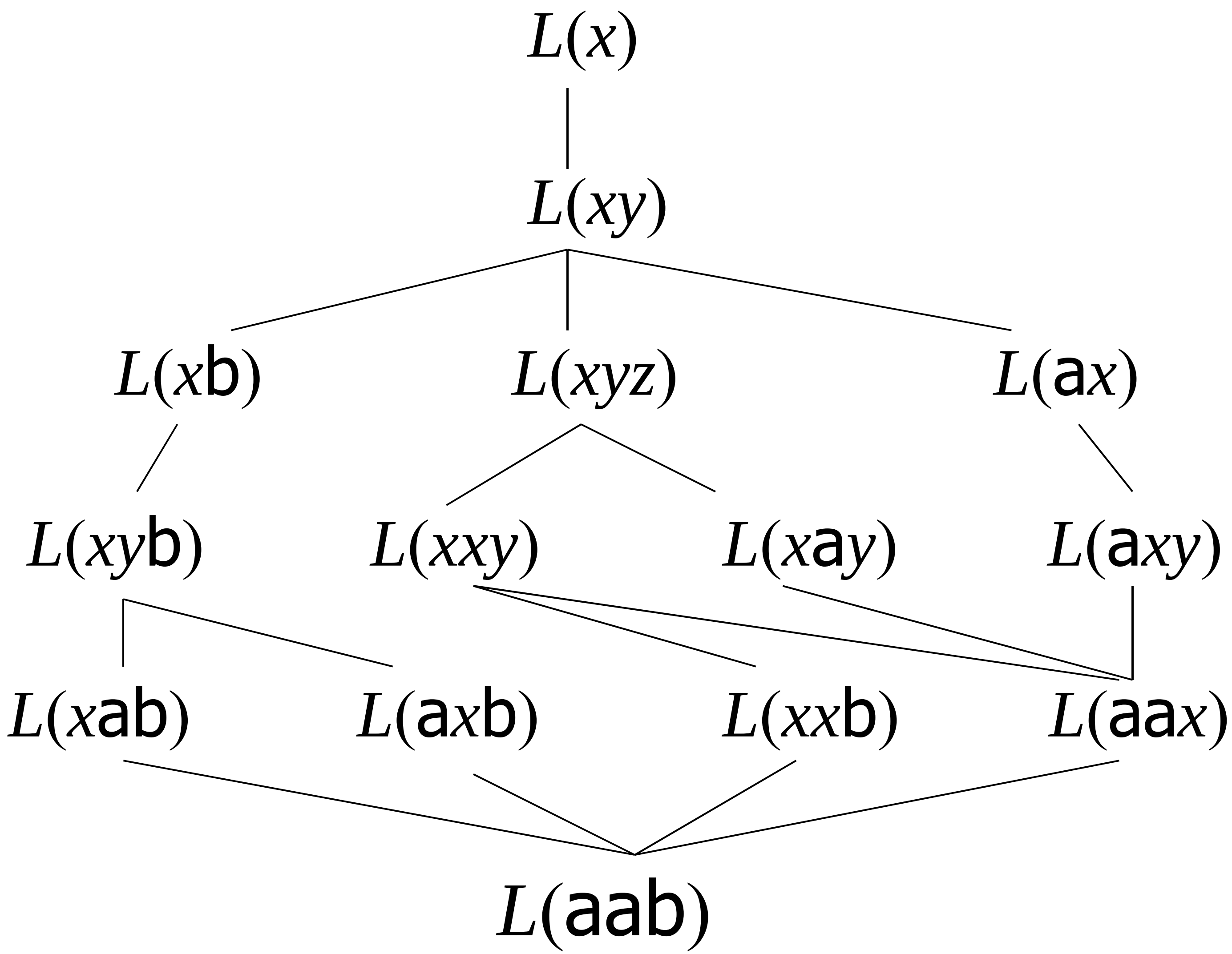
$L(\mathsf{aab})$,

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

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

$L(xyz)$,

Hasse Diagram



C4: Finite thickness

- A class L(G) of languages has the finite thickness if for all $w \in \Sigma^*$ there are only a finite number of languages in L(G) which contain w.

Theorem [Angluin] A class L(G) of languages is identifiable in the limit from positive presentation if if L(G) of languages has the finite thickness.

L(**N**) has the Finite Thickness

- From the finite thickness condition:

L(**N**) = $\{L(m) \mid m \in \mathbf{N}\}$ has the finite thickness property.

- From the fact

$$\mathrm{GCD}(e_1, e_2, \ldots, e_k) \geq \mathrm{GCD}(e_1, e_2, \ldots, e_k, e_{k+1})$$

and the following property:

Let $a_1, a_2, \ldots, a_n, \ldots$ be a infinite sequence of natural numbers satisfying that

$$a_n \geq a_{n+1} \text{ for all } n \geq 1.$$

Then there is $N \geq 1$ such that $a_n = a_{n+1}$ for all $n \geq N$.

C3:Finite Elasticity

- A class L(G) of languages has the infinite elasticity if there is an infinite sequence of strings $w_0, w_1, w_2, \ldots,$ and an infinite sequence languages in L(G) $L(g_0), L(g_1),$ $L(g_2)$ such that

  $\{w_0, w_1, \ldots, w_{n-1}\} \subseteq L(g_n)$ and $w_n \notin L(g_n)$ for every $n \geq 1$.

  A class L(G) of languages has the finite elasticity if it does not have the infinite elasticity.

Th. [Wright] A class L(G) of languages is identifiable in the limit from positive presentation if L(G) has the finite elasticity.

Relation among the conditions

U : a class of languages

- EC1（necessary and sufficient） [Angluin]

$\Uparrow \quad \not\Downarrow$

- C2: [Kobayashi]

$\Uparrow \quad \not\Downarrow$

- C3: [Wright]

$\Uparrow \quad \not\Downarrow$

- C4: [Angluin]

Announcement

- The lectures on 28th November follow the Schedule for Monday.
- The next lecture of this course is on 5th December.