Computational Learning Theory
Linear Patterns and Dynamic Programming

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

Patterns(Monomials) and Machine Leaning

Alphabets and Stings

- Σ : a finite set of symbols and called an alphabet
- Σ^* : the set of all finite strings (sequences) consisting of the symbols in Σ.
  - An empty string is denoted by ε.
  - $\Sigma^+ = \Sigma^* - \{\varepsilon\}$
  - The size of a string w, denoted by $| w |$, is the total number of symbols occurring in w.

Examples

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

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

  $|aaa| = |aab| = 3$, $|a| = |b| = 1$, $|\varepsilon| = 0$
- $\Sigma = \{A, T, C, G\}$

  $\Sigma^* = \{\varepsilon, A, T, C, G, AA, \ldots, AG, TA, \ldots, AAA, \ldots\}$

Question

- Assume that we have provided

  $C \subset \Sigma^*$: a finite set of positive examples, and

  $D \subset \Sigma^*$: a finite set of negative examples

  such that $C \cap D = \emptyset$.

- Develop a computer program to find a rule which accepts all positive examples and rejects all negative examples.

The First and Second Problem

- What are rules?
- From where do the rules come?
  - How can we generate the rules mechanically?

Examples

Example 1

$C_1 = \{$ab, aab, abaab, aaab, aaaabbbb,abab$\}$

$D_1 = \{$a, b, bbbb, abba, baaaaba, babb$\}$

- It could hold that *every string in* C_1 *starts with* a *and end with* b.

Example 2

$C_2 = \{$ba, bababa, babababa, bababababa$\}$

$D_2 = \{$a, b, bbbb, abb, baaaaba, babbb$\}$

- It might hold that *every string in* C_2 *is made of some repetition of* ba.

Examples

Example 1

C_1 = {ab, aab, abaab, aaab, aaaabbbb,abab}

D_1 = {a, b, bbbb, abba, baaaaba, babb}

- The rule which is output by a learning machine would represent a set

L_1 = {ab, aab, abb, aaab, aabb, abab, abbb,
aaaab, aaabb, ..., abaab, ..., abbbb,
..., aaaabbbb, ...}

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,...

Substitution (1)

- A substitution is a set of pairs

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

where $x_1, x_2, \ldots, x_n$ are distinct variables and

$\pi_1, \pi_2, \ldots, \pi_n$ 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

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

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

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

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

Substitution (2)

- A substitution $\theta = \{ (x_1, \tau_1), (x_2, \tau_2), \ldots, (x_n, \tau_n) \}$ 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 $\theta = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \}$ 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.

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,...}

Learnning Linear Patterns

Learning pattern languages

Example 1

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

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

π = axb

Example 5

C ={aaabbaaa, aaabbbaa, abbbbaaa, abbbbbaa}

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

π = axxbb$y$aa

Linear Patterns

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

Example

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

- Examples of linear patterns are axb, and aa$x$bybxa is not linear.

- 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.

Finding Linear Patterns(1)

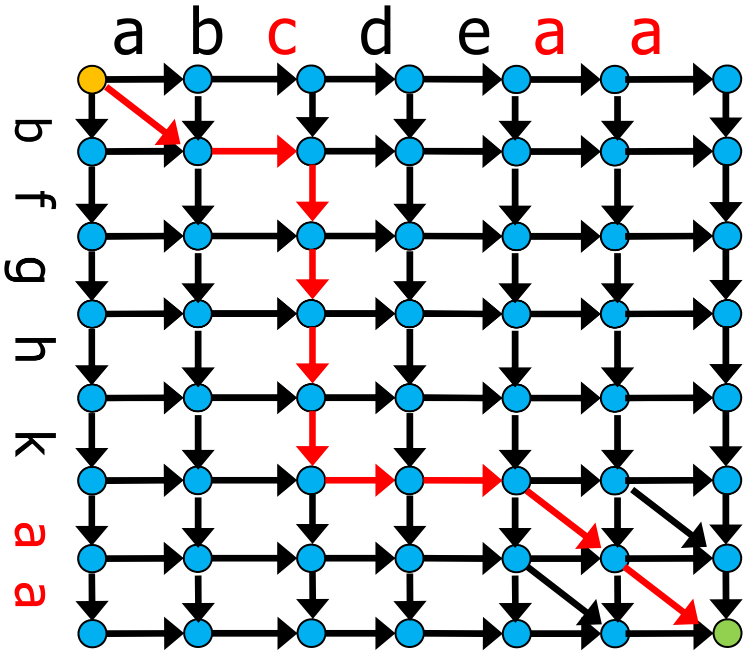
- Fill the cells from the top-left to the bottom-right.

  min(left+1, up+1 , up-left + (3 – 2δ(*c*, *d*))

  δ(*c*, *d*) = 1 if *c* = *d*, δ(*c*, *d*) = 0 o.w.

		a	b	c	d	e	a	a
	0	1	2	3	4	5	6	7
b	1	2	2	3	4	5	6	7
f	2							
g	3							
h	4							
k	5							
a	6							
a	7							

Edit Graph



Finding Linear Patterns(2)

- Follow the arrows from the bottom-right to the top-left.

		a	b	c	d	e	a	a
	0	1	2	3	4	5	6	7
b	1	2	2	3	4	5	6	7
f	2	3	3	4	5	6	7	8
g	3	4	4	5	6	7	8	9
h	4	5	5	6	7	8	9	10
k	5	6	6	7	8	9	10	11
a	6	6	7	8	9	10	10	11
a	7	7	8	9	10	11	11	11

Finding Linear Patterns(3)

- Follow the arrows from the bottom-right to the top-left.

		a	b	c	d	e	a	a
	0	1	2	3	4	5	6	7
b	1	2	2	3	4	5	6	7
f	2	3	3	4	5	6	7	8
g	3	4	4	5	6	7	8	9
h	4	5	5	6	7	8	9	10
k	5	6	6	7	8	9	10	11
a	6	6	7	8	9	10	10	11
a	7	7	8	9	10	11	11	11

Preference of Patterns

The Third Problem

- Which rule is prefer?
  - What is the loss function Loss(f, $\mathbf{x}$) and the penalty function P(f)?

$$\text{argmin}_{f\in \mathrm{H}} \left(\Sigma_{\mathbf{x} \in Data} \text{Loss}(f, \mathbf{x}) + \lambda \, \mathrm{P}(f)\right)$$

Example 1

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

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

π = axb or π = ax

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

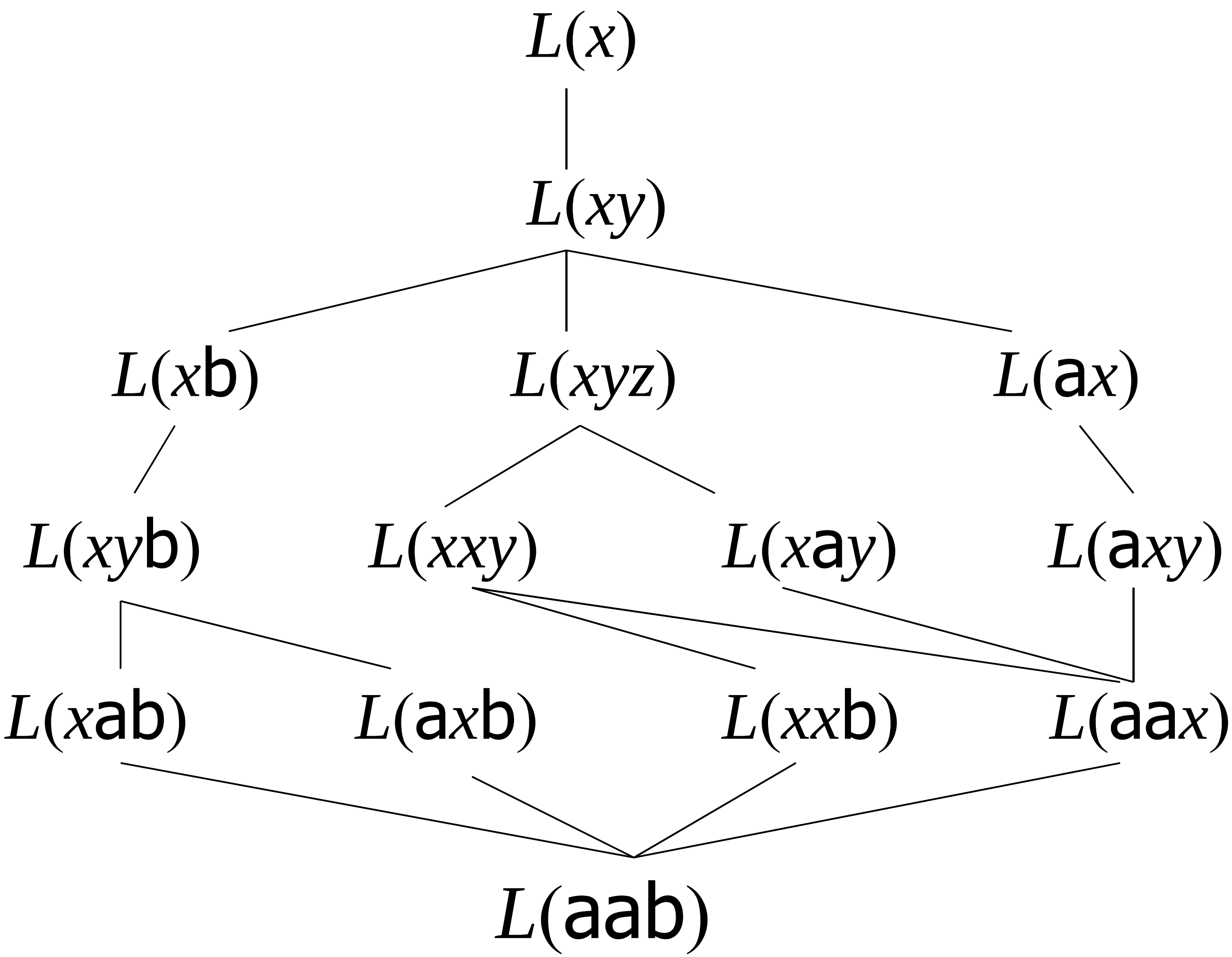
$L(\text{aab})$,

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

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

$L(xyz)$,

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,b), (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, abbbbbaaa

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

  - 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 $L(\pi)$.

Anti-Unifcation of Strings

- For a set C of stings of same length

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

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 $c_1c_2\ldots c_n$.