November 22, 2016

Refinement on Learning

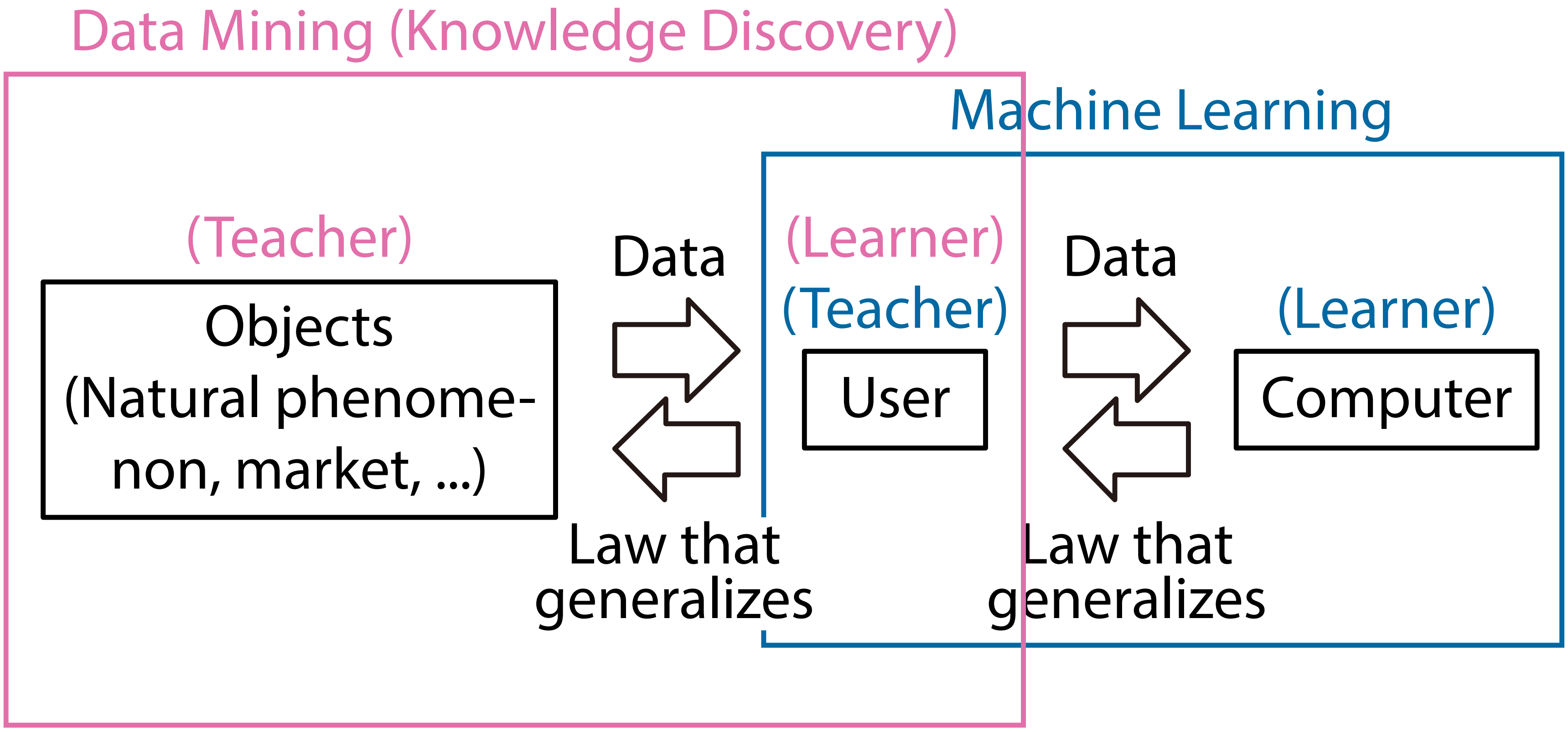
Data Mining Theory (データマイニング工学)

Mahito Sugiyama (杉山麿人)

Today's Outline

- Recap the main points of last week's lecture
- Consider the structure of a hypothesis space
  - Essential to efficiently search candidate hypotheses
- Understand the hypothesis space as a poset (半順序集合)
- Introduce the key concept of a refinement (精密化) operator to traverse the (structured) hypothesis space

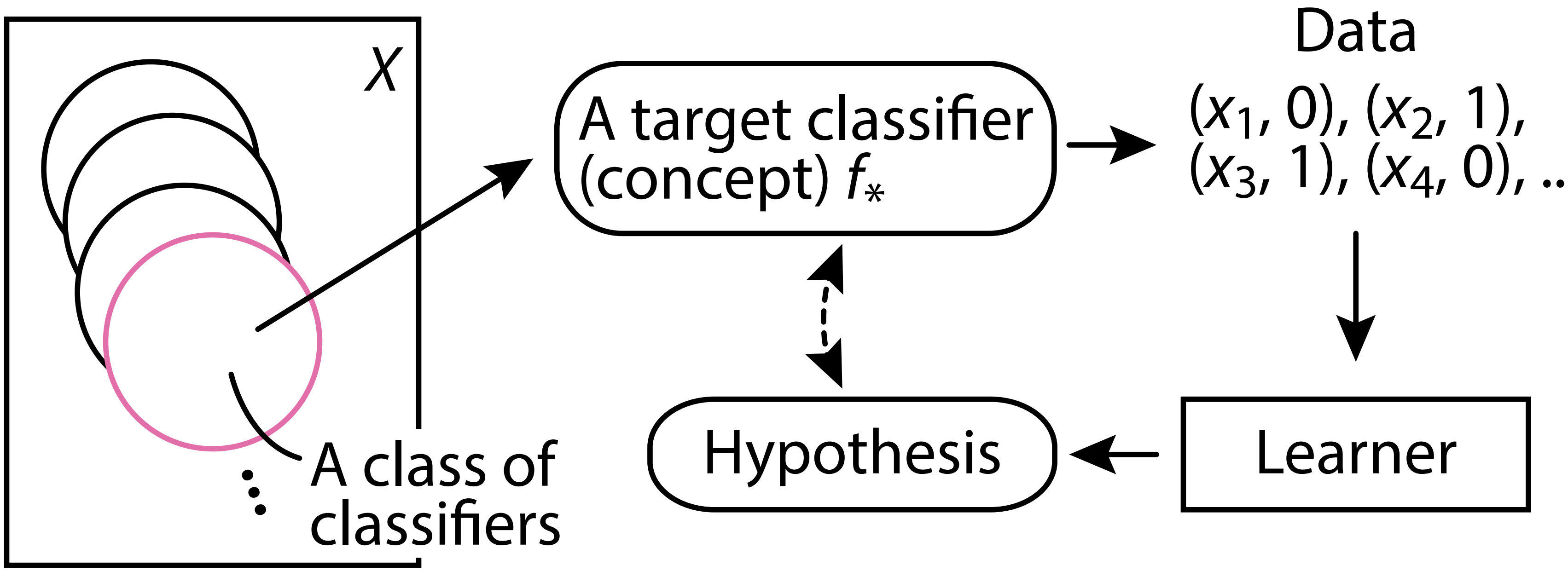
Framework of Learning (ML vs DM)



Formalization of Learning in Computational Manner

1. What are targets of learning? (学習対象)
   - Each target (concept) C is a subset of the domain X ($C \subseteq X$)
   - A concept space $\mathcal{C}$ is a collection of concepts ($\mathcal{C} \subseteq \mathcal{P}(X)$)
2. How to represent targets and hypotheses? （表現言語）
   - We use a hypothesis space $\mathcal{H}$
   - Each hypothesis $H \in \mathcal{H}$ represents a concept $\upsilon(H) \subseteq X$
3. How are data provided to a learner? （データ）
4. How does the learner work? (学習手順，アルゴリズム)
5. When can we say that the learner correctly learns the target? (学習の正当性)

Learning Model



Gold's Learning Model on Languages

- A concept space $\mathcal{C} \subseteq \{ A \mid A \subseteq \Sigma^* \}$ is chosen
- For a language $C \in \mathcal{C}$, an infinite sequence $\sigma = (x_1, y_1), (x_2, y_2), \dots$ is a complete presentation (完全提示) of C if
  (i) $\{x_1, x_2, \dots\} = \Sigma^*$
  (ii) $y_i = 1 \iff x_i \in C$ for all i
- A learner is a procedure M that receives σ and generates an infinite sequence of hypotheses $\gamma = H_1, H_2, \dots$
- If γ converges to some hypothesis H and $\upsilon(H) = C$, we say that *M* identifies *C* in the limit (極限学習する)
  - If M identifies any $C \in \mathcal{C}$ in the limit, *M* identifies $\mathcal{C}$ in the limit

Consistency of Hypotheses

- A language C is inconsistent with (x, y) (矛盾する) if $(y = 1 \text{ and } x \notin C)$ or $(y = 0 \text{ and } x \in C)$
- C is consistent with (x, y) if C is not inconsistent with (x, y)
- For a set of examples $S = \{ (x_1, y_1), \dots, (x_n, y_n) \}$,
  C is consistent with S (C は S に無矛盾)
  if C is consistent with every $(x, y) \in S$

Basic Strategy: Generate and Test

- Input: a complete presentation σ of a language $C \in \mathcal{C}$
- Output: $\gamma = H_1, H_2, \ldots$

```
i ← 1, S ← ∅
repeat
    S ← S ∪ {(x_i, y_i)}
    while υ(H) is not consistent with S do
        H ← the next hypothesis in the hypothesis space ℋ
    end while
    H_i ← H and output H_i
    i ← i + 1
until forever
```

Power of Generate and Test Strategy and Its Problem

- For any class $\mathcal{C}$ of languages, Generate and Test strategy identifies $\mathcal{C}$ in the limit
  - That is, Generate and Test strategy identifies every language $C \in \mathcal{C}$ in the limit
- Unfortunately, this strategy is not realistic

Power of Generate and Test Strategy and Its Problem

- For any class $\mathcal{C}$ of languages, Generate and Test strategy identifies $\mathcal{C}$ in the limit
  - That is, Generate and Test strategy identifies every language $C \in \mathcal{C}$ in the limit
- Unfortunately, this strategy is not realistic
- What is needed for more efficient learning?

→ An efficient search of candidate hypotheses is essential!

Structurization of Hypothesis Space

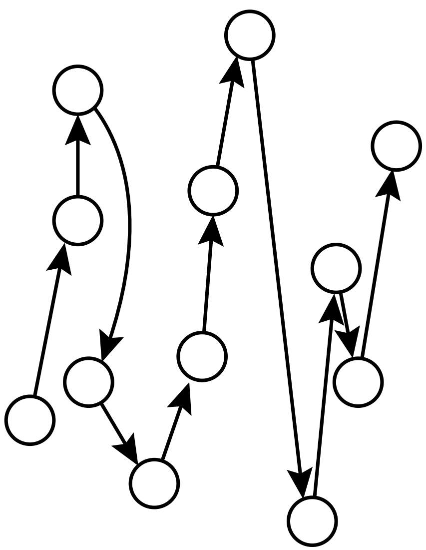
- To search hypotheses,
  (i) The structure of the hypothesis space $\mathcal{H}$
  (ii) An operator that enables to traverse the space

  are indispensable

1. The structured space is mathematically modeled as a poset (partially ordered set; 半順序集合)
2. As an operator, we use refinement (精密化)
   – For each hypothesis, a learner can “refine” it and derive a set of one level specific hypotheses

Structurization of Hypothesis Space

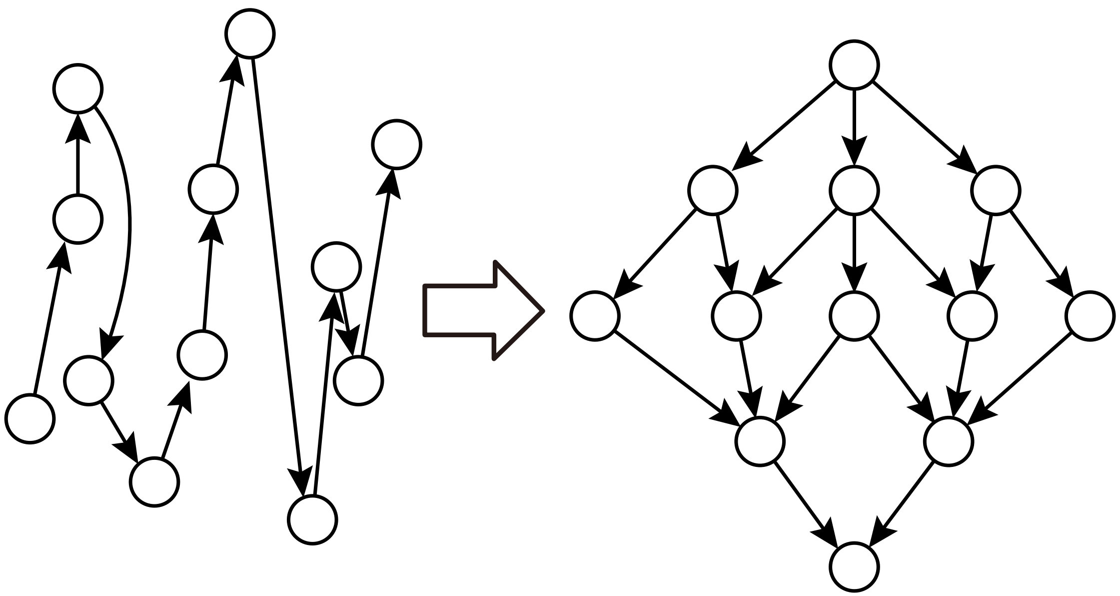
No structure
in hypothesis space



Structurization of Hypothesis Space

No structure
in hypothesis space

Poset with refinement

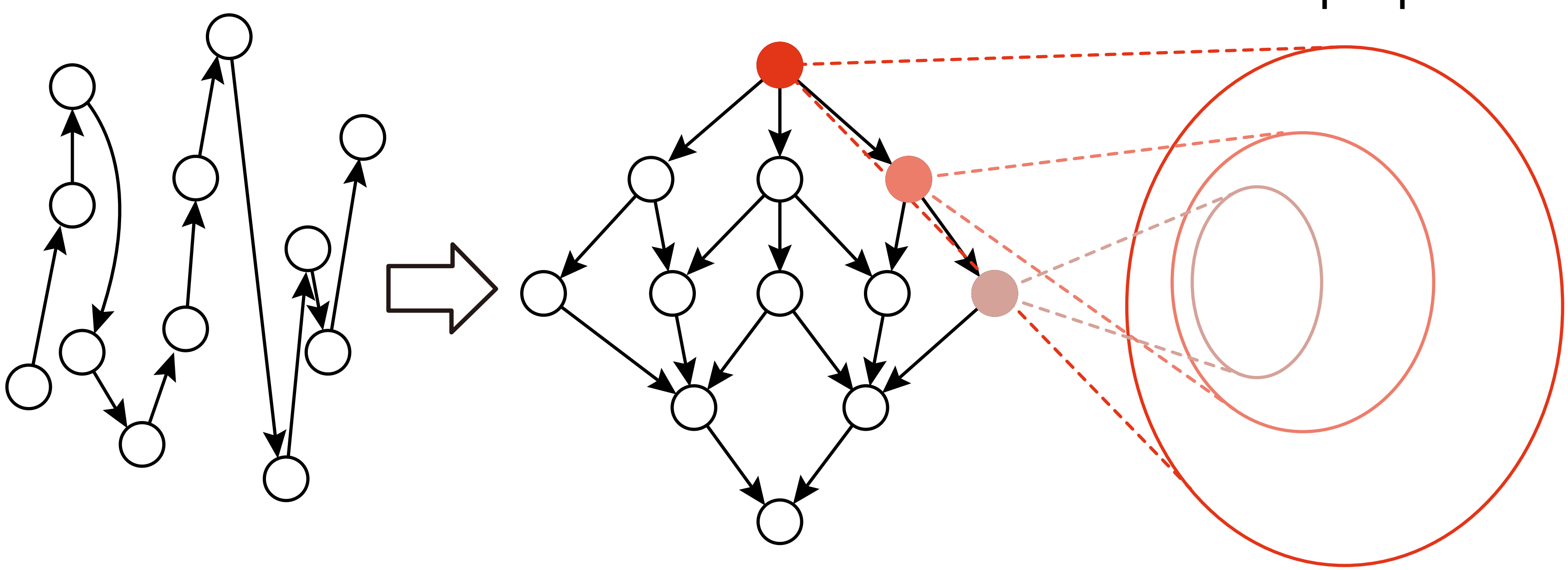


Structurization of Hypothesis Space

No structure
in hypothesis space

Poset with refinement

Subset inclusion
relationships in
concept space



Poset

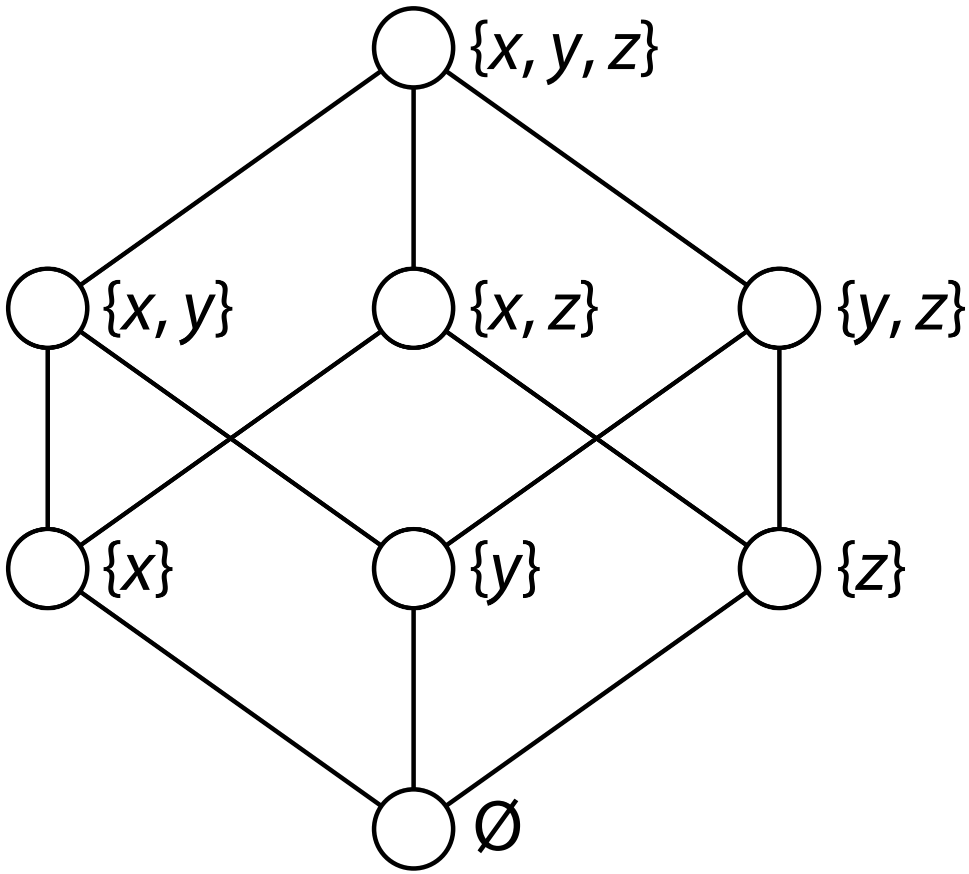
- A partial order (半順序) is a binary relation $\leq$ s.t.
  1. $x \leq x$ (reflexivity; 反射律)
  2. $(x \leq y \text{ and } y \leq x) \Rightarrow x = y$ (antisymmetry; 反対称律)
  3. $(x \leq y \text{ and } y \leq z) \Rightarrow x \leq z$ (transitivity; 推移律)

- A set X with a partial order $\leq$, denoted as $(X, \leq)$, is called a partially ordered set (poset; 半順序集合)
  - The least upper bound (supremum; 最小上界) of $S \subseteq X$ is the least $x \in X$ s.t. $\forall s \in S, s \leq x$
  - The greatest lower bound (infimum; 最大下界) of $S \subseteq X$ is the greatest $x \in X$ s.t. $\forall s \in S, x \leq s$

Lattice

- We use join "$\vee$" (結び) and meet "$\wedge$" (交わり)
  - $x \vee y = \sup\{x, y\}$ (join of x and y)
    - For $S \subseteq X$, $\vee S = \sup S$
  - $x \wedge y = \inf\{x, y\}$ (meet of x and y)
    - For $S \subseteq X$, $\wedge S = \inf S$
- A poset $(X, \leq)$ is a lattice (束) if $x \vee y$ and $x \wedge y$ exist for all $x, y \in X$
- Examples:
  - The power set $\mathcal{P}(X)$ of any set X (we translate "$\subseteq$" as $\leq$)
  - The set of natural numbers $\mathbb{N}$ w.r.t "$\leq$"
  - The Cartesian product $\mathbb{N} \times \mathbb{N} = \{(a, b) \mid a, b \in \mathbb{N}\}$, $(a, b) \leq (a', b')$ if $a \leq a'$ and $b \leq b'$

The Power Set Is a Lattice



Definition of Refinement

- Assume that our hypothesis space $(\mathcal{H}, \preceq)$ is a poset and

  $G \preceq H \Rightarrow \upsilon(G) \subseteq \upsilon(H)$

  $G \equiv H \Rightarrow \upsilon(G) = \upsilon(H)$

  – υ should be a homomorphism (準同型写像) that preserves structure between $\mathcal{C}$ and $\mathcal{H}$

- A refinement (精密化) is a mapping $\rho : \mathcal{H} \to 2^{\mathcal{H}}$ s.t.

  1. $\forall H \in \mathcal{H}, \rho(H)$ is finite
  2. $G \in \rho(H) \Rightarrow G \preceq H$
  3. $\forall H \in \mathcal{H},$ there is no infinite sequence $H_1, H_2, \ldots$ s.t. $H = H_1$ and $H_i \in \rho(H_{i+1})$

Semantically Complete Refinement

- We write $X \xrightarrow{\rho} Y$ if $Y \in \rho(X)$
  - $\xrightarrow{*}$ is zero or more applications of $\xrightarrow{\rho}$
- A refinement ρ is semantically complete (意味的に完全) if
  $\left\{ \upsilon(G) \;\middle|\; H \xrightarrow{*} G \right\} = \{ C \in \mathcal{C} \mid C \subset \upsilon(H) \}$
  - Start from H, we can find any $C \subset \upsilon(H)$ by applying $\xrightarrow{\rho}$
  - If this condition is not satisfied, we will miss some concepts

Pioneers of Refinement

- Refinement is (implicitly) used in various contexts
  - It can be viewed as an online construction of search space with tree-like structure
- It has been explicitly introduced in Model Inference System by Shapiro in 1981
  - E. Y. Shapiro, **An Algorithm That Infers Theories from Facts**, *IJCAI*, 1981
- Plotkin considered the opposite direction (from specific to general)
  - G. D. Plotkin, **A further note on inductive generalization**, *Machine Intelligence*, 1970

Examples of Refinement

- Let us consider concrete examples of refinement and learning
- We use two simple examples:
  - Regular language (正則言語)
  - The set of pairs of natural numbers
    $\mathbb{N}^2 = \mathbb{N} \times \mathbb{N} = \{ (a, b) \mid a, b \in \mathbb{N} \}$

Regular Language (1/2)

- Given an alphabet Σ
  - For $a \in \Sigma$, $a^2 = aa, a^3 = aaa, \ldots$
  - $X^0 = \varnothing$, $X^n = \{\, au \mid a \in X, u \in X^{n-1} \,\}$ $(n \geq 1)$
- For a regular expression (正則表現, RE) H, $\upsilon(H)$ is a regular language (正則言語)
  - $\varnothing$ is an RE; $\upsilon(\varnothing) = \varnothing$
  - $\forall a \in \Sigma$, a is an RE; $\upsilon(a) = \{a\}$
  - If X and Y are REs,
    - $X + Y$ is an RE; $\upsilon(X + Y) = X \cup Y$ (union)
    - XY is an RE; $\upsilon(XY) = \{\, ab \mid a \in X, b \in Y \,\}$ (concatenation)
    - X^* is an RE; $\upsilon(X^*) = \bigcup \{\, X^n \mid n \geq 0 \,\}$ (Kleene closure; クリーネ閉包)

Regular Language (2/2)

- Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$
- We denote by $\top$ the language $(a_1 + a_2 + \cdots + a_n)^*$
  - $\upsilon(\top) = \Sigma^*$
  - The largest language over Σ
- Examples:
  - Assume that $\Sigma = \{\mathtt{a}, \mathtt{t}, \mathtt{g}, \mathtt{c}\}$
  - $\upsilon(\mathtt{at} + \mathtt{g}^*) = \{\varepsilon, \mathtt{at}, \mathtt{g}, \mathtt{gg}, \mathtt{ggg}, \ldots\}$
  - $\upsilon((\mathtt{a} + \mathtt{c})^*) = \{\varepsilon, \mathtt{a}, \mathtt{c}, \mathtt{aa}, \mathtt{ac}, \mathtt{ca}, \mathtt{cc}, \mathtt{aaa}, \ldots\}$
  - $\upsilon(\top) = \{\varepsilon, \mathtt{a}, \mathtt{t}, \mathtt{g}, \mathtt{c}, \mathtt{aa}, \mathtt{at}, \ldots\}$

Refinement on Regular Languages

(from P. D. Laird, Learning from Good and Bad Data, 1988)

1. $X \xrightarrow{\rho} X + X$
2. $X^* \xrightarrow{\rho} X^* X^*$
3. $X^* \xrightarrow{\rho} (X^*)^*$
4. $a \xrightarrow{\rho} \varnothing \quad (a \in \Sigma)$
5. $X^* \xrightarrow{\rho} X$
6. $X \xrightarrow{\rho} Y \Rightarrow X + Z \xrightarrow{\rho} Y + Z$
7. $X \xrightarrow{\rho} Y \Rightarrow Z + X \xrightarrow{\rho} Z + Y$
8. $X \xrightarrow{\rho} Y \Rightarrow X^* \xrightarrow{\rho} Y^*$
9. $X \xrightarrow{\rho} Y \Rightarrow XZ \xrightarrow{\rho} YZ$
10. $X \xrightarrow{\rho} Y \Rightarrow ZX \xrightarrow{\rho} ZY$

Examples of Refinement on Regular Languages

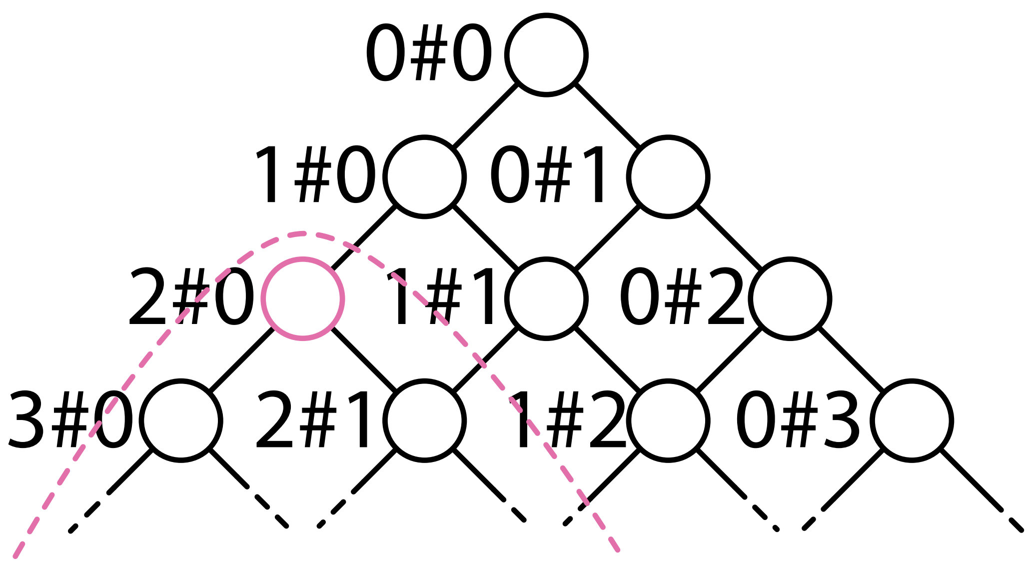
- Let $\Sigma = \{0, 1\}$
- $\top = (0 + 1)^* \xrightarrow{\rho} 0 + 1 \xrightarrow{\rho} \varnothing + 1 \xrightarrow{\rho} \varnothing + \varnothing$
- $\top = (0+1)^* \xrightarrow{\rho} (0+1)^*(0+1)^* \xrightarrow{\rho} (0+1)^*(0+1) \xrightarrow{\rho} (0+1)(0+1)$
  - $\upsilon((0 + 1)(0 + 1)) = \{00, 01, 10, 11\}$

Efficient Learning with Refinement

$i \leftarrow 1$, $S \leftarrow \varnothing$, $H \leftarrow \top$, $Q \leftarrow \varnothing$ // Q is a list of candidate hypotheses
repeat
    $S \leftarrow S \cup \{(x_i, y_i)\}$
    while H is not consistent with S
        if $x \in \upsilon(H)$ for some $(x, 0) \in S$ **then**
            Append all $\rho(H)$ to the tail of Q
        end if
        $H \leftarrow$ the first hypothesis in Q, and remove it from Q
    end while
    $H_i \leftarrow H$ and output H_i
    $i \leftarrow i + 1$
until forever

Hypothesis Space on $\mathbb{N}^2$

- $\mathcal{H} = \{\, a\#b \mid a, b \in \mathbb{N} \,\}$
- $a\#b \leq c\#d$ if $a \geq c$ and $b \geq d$
  - Note that we invert $\leq$ for mathematical convenience



Refinement on $\mathbb{N}^2$

- We consider the following concept space $\mathcal{C}$:

  $\mathcal{C} = \{ \uparrow(a, b) \mid a, b \in \mathbb{N} \}$,

  where

  $\uparrow(a, b) = \{ (c, d) \in \mathbb{N}^2 \mid a \leq c, b \leq d \}$

  – A subset $O \subseteq \mathbb{N}^2$ s.t. $(a, b) \in O \Rightarrow \uparrow(a, b) \subseteq O$ is known to be open on the Alexandroff topology

- Define $\upsilon(a\#b) = \uparrow(a, b)$

- Refinement is given as follows:

  1. $a\#b \xrightarrow{\rho} (a + 1)\#b$
  2. $a\#b \xrightarrow{\rho} a\#(b + 1)$

Refinement on Sets of $\mathbb{N}^2$

- We can further treat a (finite) set of $\uparrow(a, b)$ as a concept
- $\mathcal{H}_S = \{\, a_1\#b_1 + a_2\#b_2 + \cdots + a_n\#b_n \mid a_i, b_i, n \in \mathbb{N} \,\}$
- $\mathcal{C}_S = \{\, C \mid C \subseteq \mathcal{C} = \{\uparrow(a, b) \mid a, b \in \mathbb{N}\},\ C \text{ is finite} \,\}$
- $\upsilon(a_1\#b_1 + \cdots + a_n\#b_n) = \uparrow(a_1, b_1) \cup \cdots \cup \uparrow(a_n, b_n)$
- Refinement is given as follows:
  1. $a\#b \overset{\rho}{\to} (a+1)\#b$
  2. $a\#b \overset{\rho}{\to} a\#(b+1)$
  3. $X \overset{\rho}{\to} Y \Rightarrow X + Z \overset{\rho}{\to} Y + Z$ and $Z + X \overset{\rho}{\to} Z + Y$
  4. $X \overset{\rho}{\to} X + X$

How about $\mathbb{R}$?

- Let us consider the set of real numbers $\mathbb{R}$
  - One of the most important objects in machine learning
- Each real number $x \in \mathbb{R}$ is represented as an infinite sequence
  - e.g., use infinite decimal expansions with $\Sigma = \{0, 1, \ldots, 9\}$
  - Let $\overline{x}$ be a representation of x
- Obviously, we cannot treat all elements in $\mathbb{R}$ as we cannot determine $x \in \mathbb{R}$ from $\overline{x}$ in finite time
- We can just treat prefixes of infinite sequences, and $\upsilon(w) = \{\, x \in \mathbb{R} \mid w \sqsubseteq \overline{x} \,\}$, which forms an open set on $\mathbb{R}$