

November 25, 2015



Discretization and Learning

Data Mining Theory (データマイニング工学)

Mahito Sugiyama (杉山磨人)

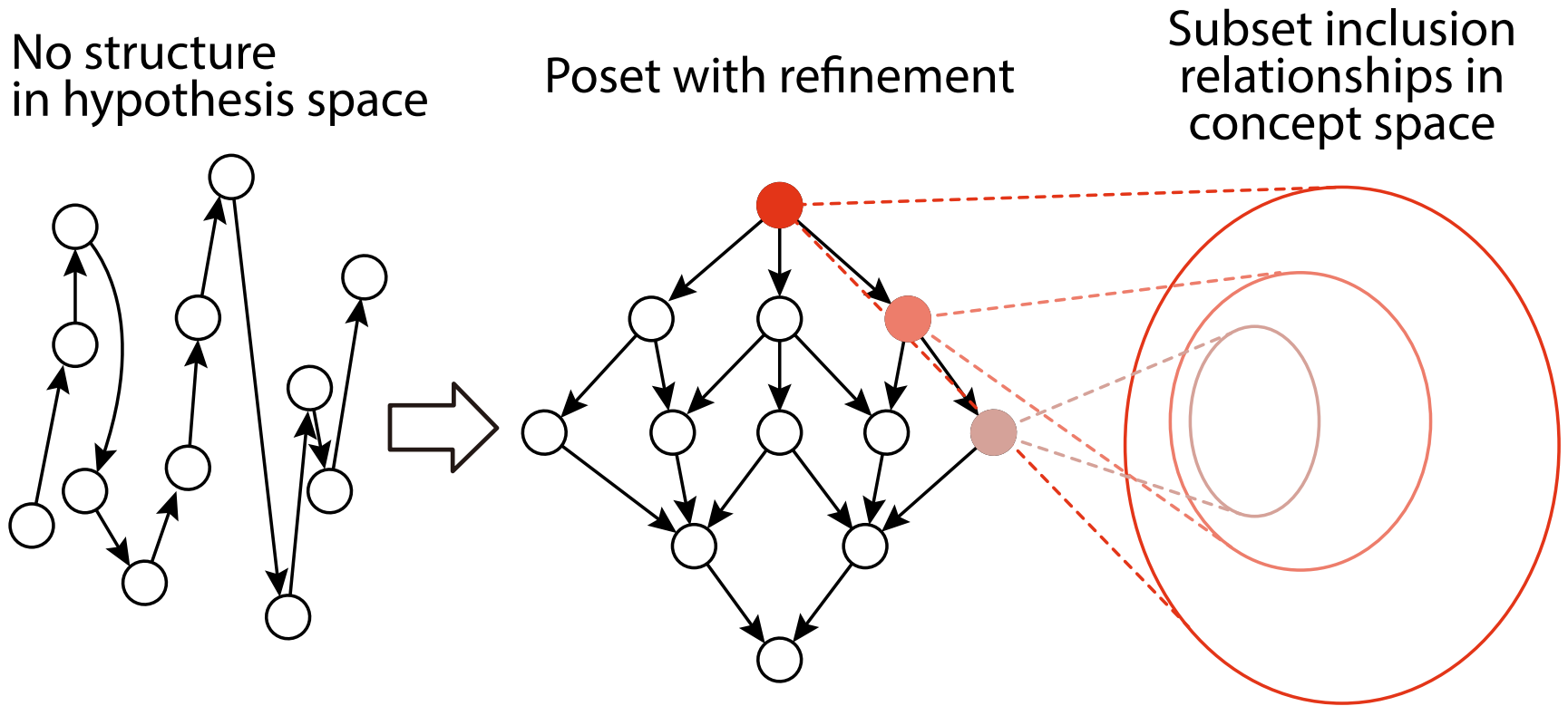
Today's Outline

- Recap the main points of last week's lecture
- Discretization on learning
- Real number computation (実数計算)
- Learning = computing = discretization?

Structurization of Hypothesis Space









- To search hypotheses,
 - (i) The **structure** of the hypothesis space \mathcal{H}
 - (ii) An **operator** that enables to traverse the spaceare indispensable
- The structured space is mathematically modeled as a **poset** (partially ordered set; 半順序集合)
- 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



Discretization and Learning

Analog data (reals)

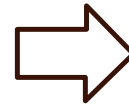
	Feature A	Feature B	Class
1			1
2			0
3			1
4			0

Goal: Learning of a classifier

Discretization and Learning

Analog data (reals)



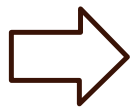






	Feature A	Feature B	Class
1			1
2			0
3			1
4			0



Discretization by measurement

Goal: Learning of a classifier










Discretization and Learning

Analog data (reals)			Digital data (rationals)		
	Feature A	Feature B Class		Feature A Feature B Class	
1		 1		1.5	0.6 1
2		 0		0.6	0.2 0
3		 1		1.1	0.4 1
4		 0		1.8	0.7 0

Discretization by measurement

Goal: Learning of a classifier

Discretization and Learning









Analog data (reals)				Digital data (rationals)			
	Feature A	Feature B	Class		Feature A	Feature B	Class
1	 1.539582...	 0.6069...	1		1.5	0.6	1
2	 0.676711...	 0.2655...	0		0.6	0.2	0
3	 1.111577...	 0.4998...	1		1.1	0.4	1
4	 1.871501...	 0.7569...	0		1.8	0.7	0

Discretization by measurement

Goal: Learning of a classifier

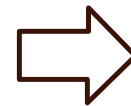
Discretization and Learning

Analog data (reals)

	Feature A	Feature B	Class
1	 1.539582...	 0.6069...	1
2	 0.67	 ...	0
3	 1.111577...	 0.4998...	1
4	 1.871501...	 0.7569...	0

Digital data (rationals)

	Feature A	Feature B	Class
1	1.5	0.6	1
2	0	0	0
3	1.1	0.4	1
4	1.8	0.7	0



Data in theory

Data on computer

Discretization by measurement

Goal: Learning of a classifier

Fatal Error Caused by Discretization

- Solve the system of equations [Schröder, 2003]

$$40157959.0 x + 67108865.0 y = 1$$

$$67108864.5 x + 112147127.0 y = 0$$

- We can solve by the well-known formula:

$$x = \frac{b_1 a_{22} - b_2 a_{12}}{a_{11} a_{22} - a_{21} a_{12}}, \quad y = \frac{b_2 a_{11} - b_1 a_{21}}{a_{11} a_{22} - a_{21} a_{12}}$$

- Computation by floating point arithmetic with double precision variables (IEEE 754):









$$x = 112147127, \quad y = -67108864.5$$

- Correct solution:

$$x = 224294254, \quad y = -134217729$$

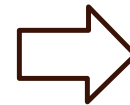
Treat Data as Intervals

Analog data (reals)

	Feature A	Feature B	Class
1	 1.539582...	 0.6069...	1
2	 0.676711...	 0.2655...	0
3	 1.111577...	 0.4998...	1
4	 1.871501...	 0.7569...	0

Digital data (rationals)

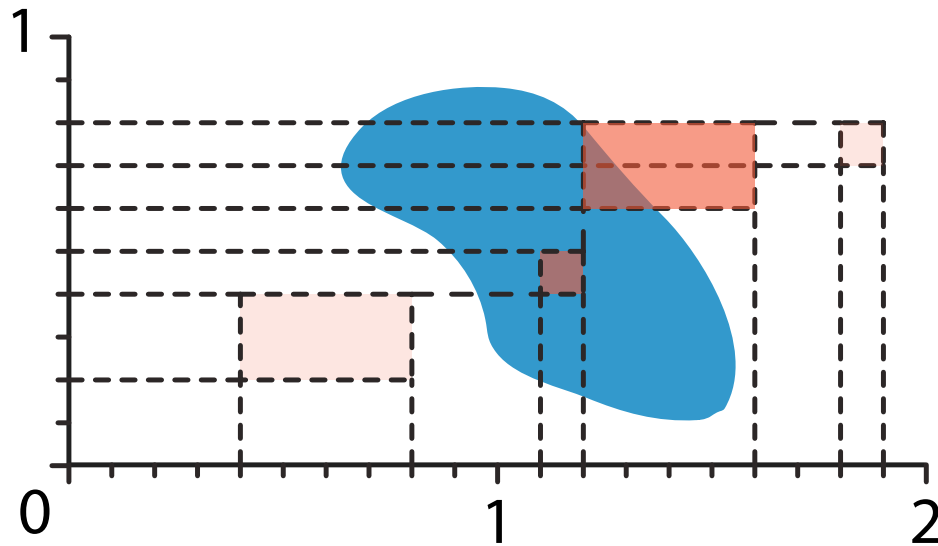
Feature A	Feature B	Class
1.2~1.6	0.6~0.8	1
0.4~0.8	0.2~0.4	0
1.1~1.2	0.4~0.5	1
1.8~1.9	0.7~0.8	0



Discretization by measurement

Goal: Learning of a classifier

Geometric Point of View



Digital data (rationals)

Feature A	Feature B	Class
-----------	-----------	-------

1.2~1.6	0.6~0.8	1
---------	---------	---

0.4~0.8	0.2~0.4	0
---------	---------	---

1.1~1.2	0.4~0.5	1
---------	---------	---

1.8~1.9	0.7~0.8	0
---------	---------	---

- Discretized data are intervals in \mathbb{R}^d
 - The width of an interval corresponds to the error of a data point
- A learner finds a set intersecting intervals of class 1

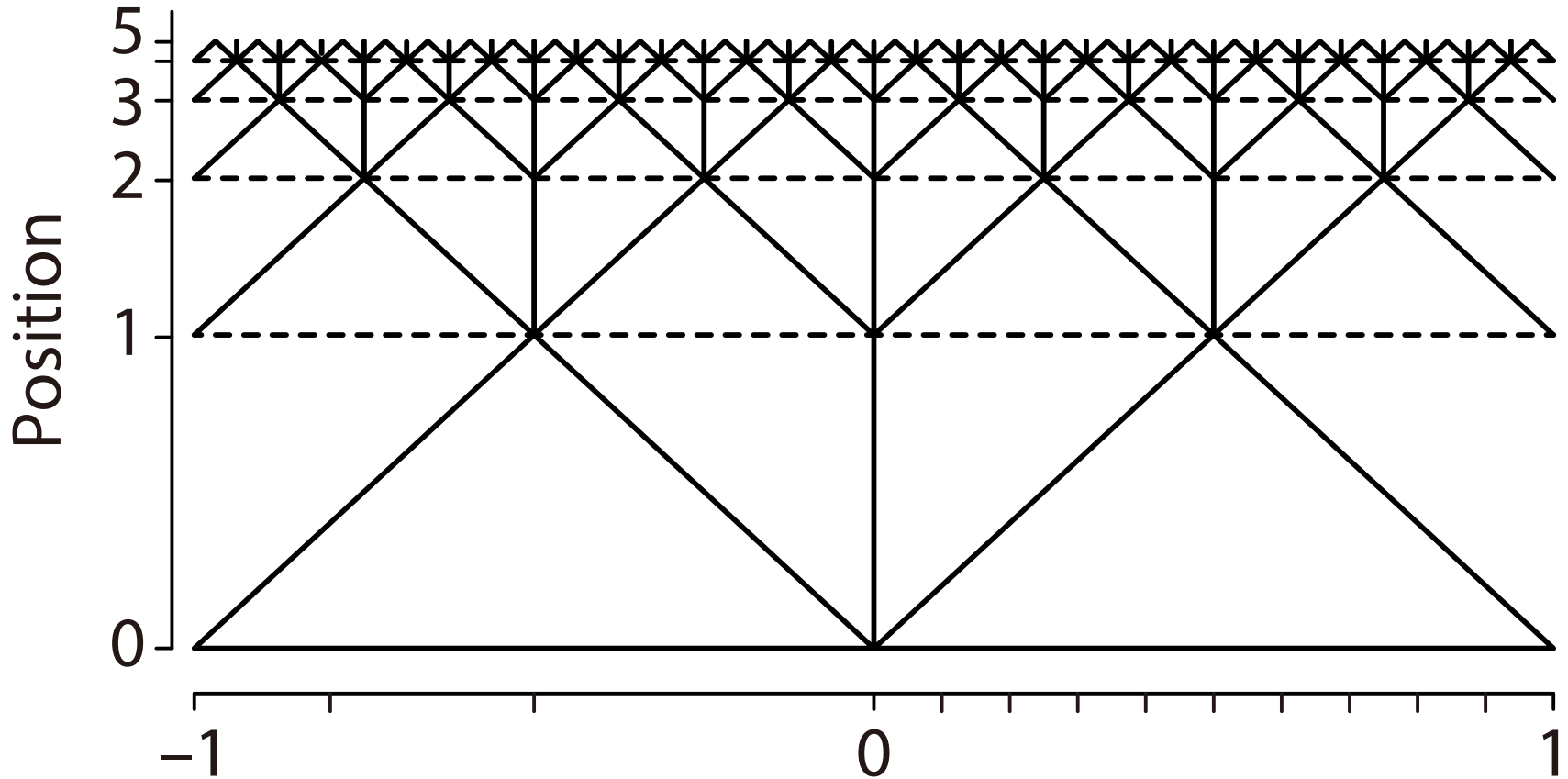
How to Compute Real Numbers?

- Consider computation of $f(x) = 3 \cdot x$
- For example: $f(1/3) = 3 \cdot 1/3$
- Since $1/3 = 0.33333 \dots$, a computer should output $0.99999 \dots$ (or $1.00000 \dots$)
- However, it cannot output any digit since:
 - If an input is $0.333 \dots$ forever, the output is $0.999 \dots$
 - If an input is $0.333 \dots 34$ at some point, the output is $1.000 \dots 02$
- Thus the computer cannot determine even the first digit at any moment

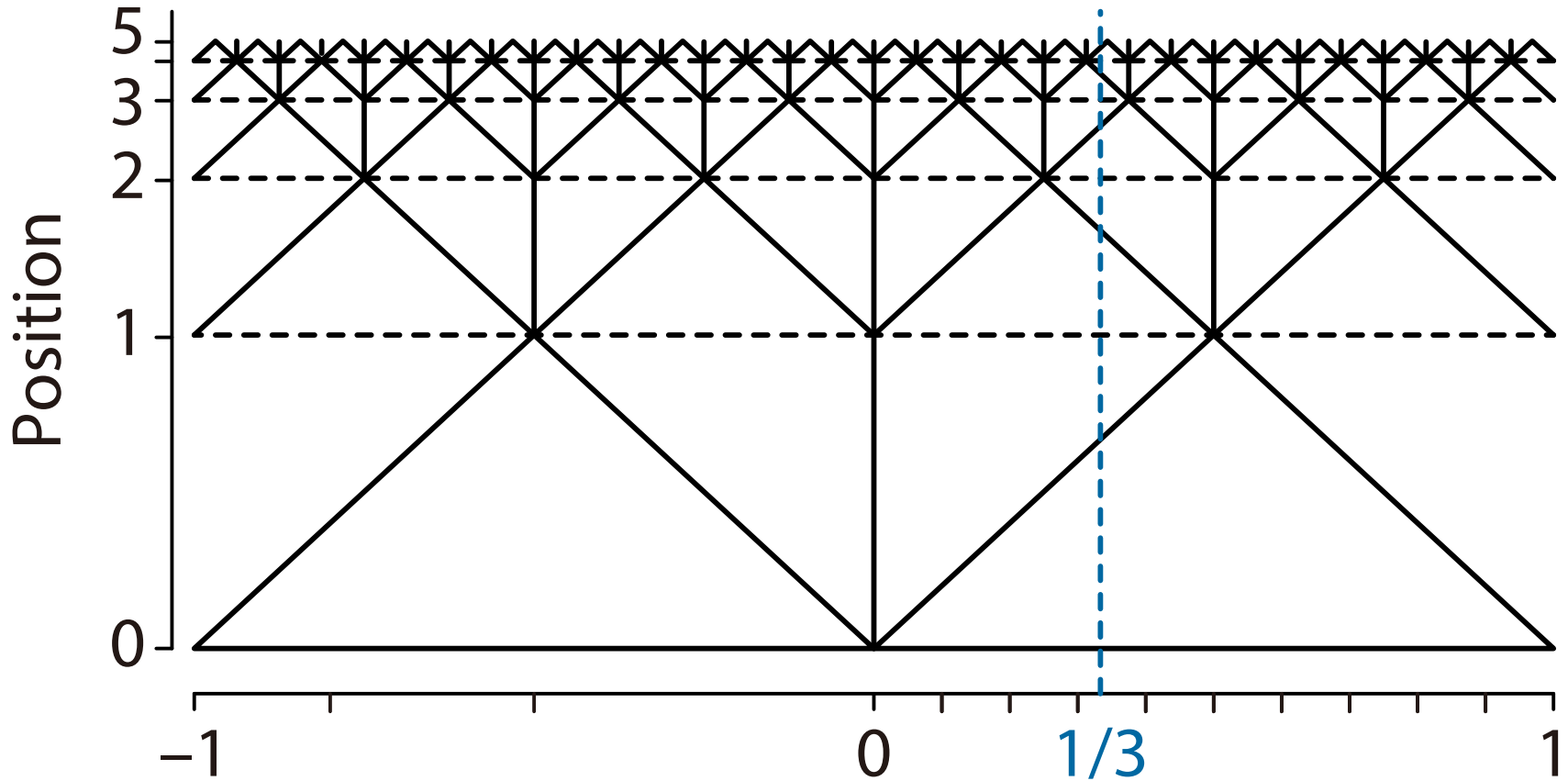
What is Problem in Real Number Computation?

- The problem arises from the **representation** of real numbers (実数表現)
- **Decimal representation** (10進表現) is a lack of **redundancy** (冗長性)
 - We need more sequences that represent the same number
- **Solution: signed digit representation** (符号付き 2進数)
 - Use three symbols: **1**, **0**, and **$\bar{1}$** (means -1) and defined as:
$$\rho(a_1 a_2 \dots) = \sum_{i=1}^{\infty} a_i \cdot 2^{-i}$$
 - Same as the binary representation if we use only 0 and 1

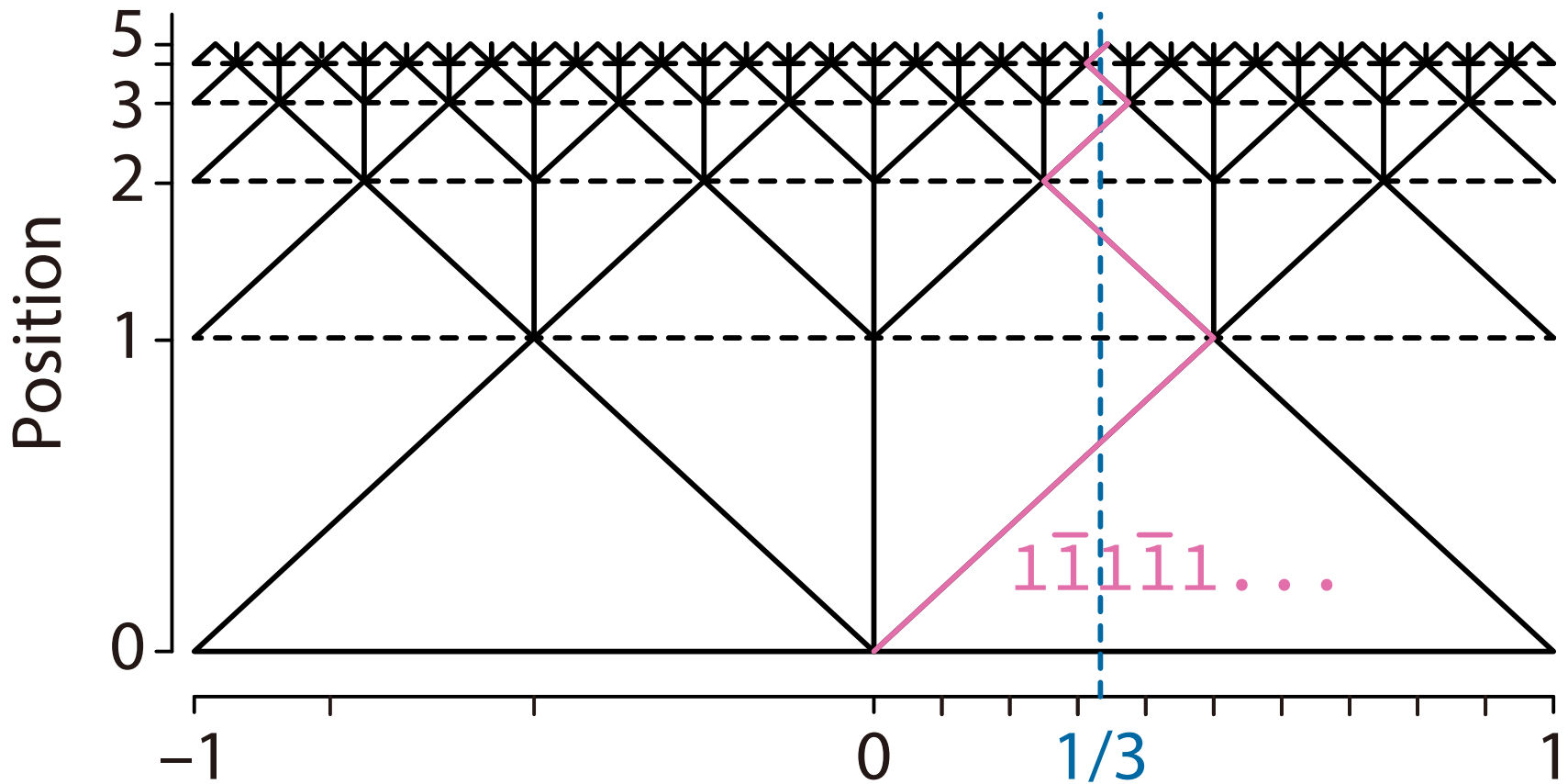
Signed Digit Representation



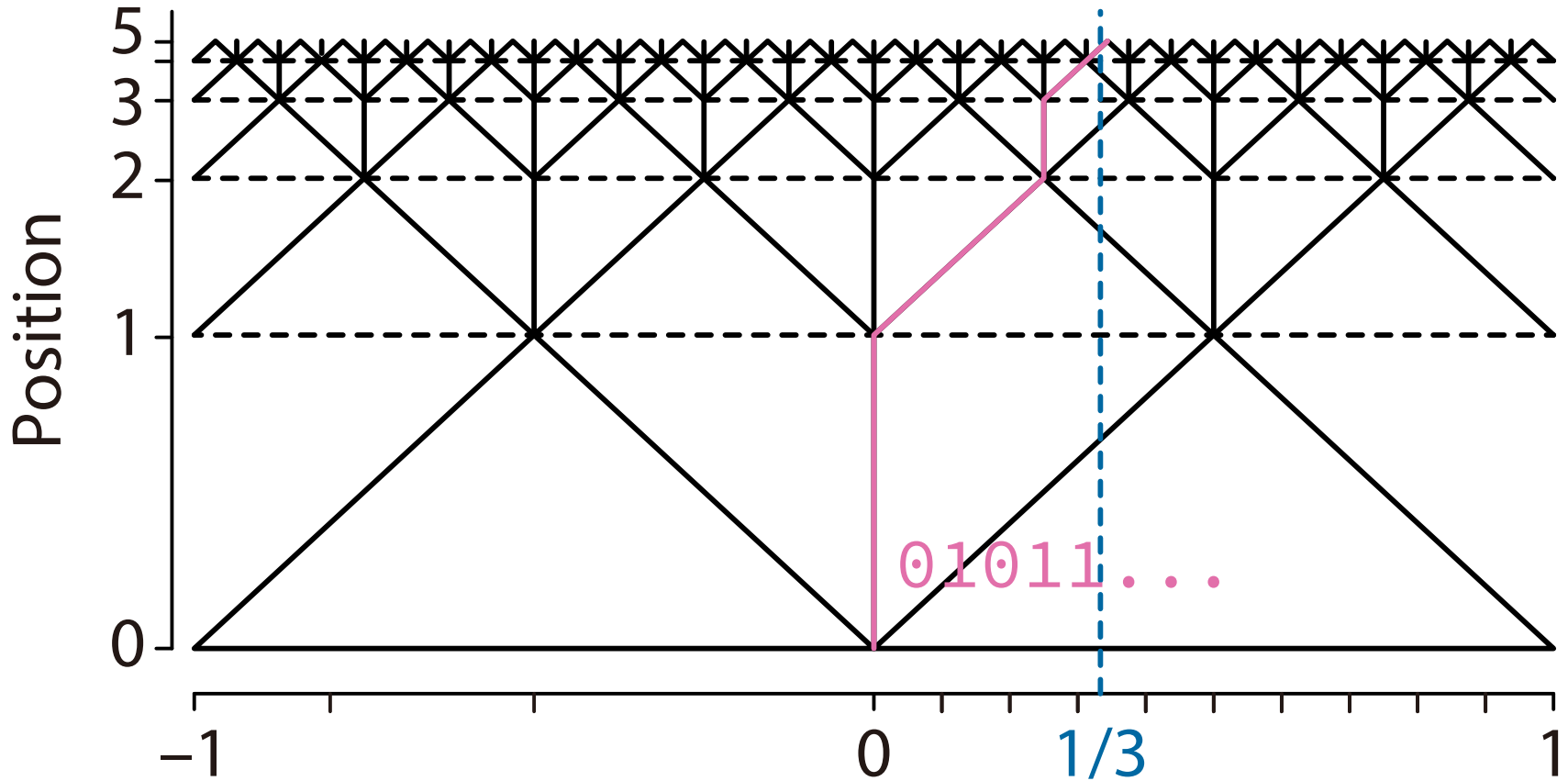
Signed Digit Representation



Signed Digit Representation



Signed Digit Representation



Gray Code

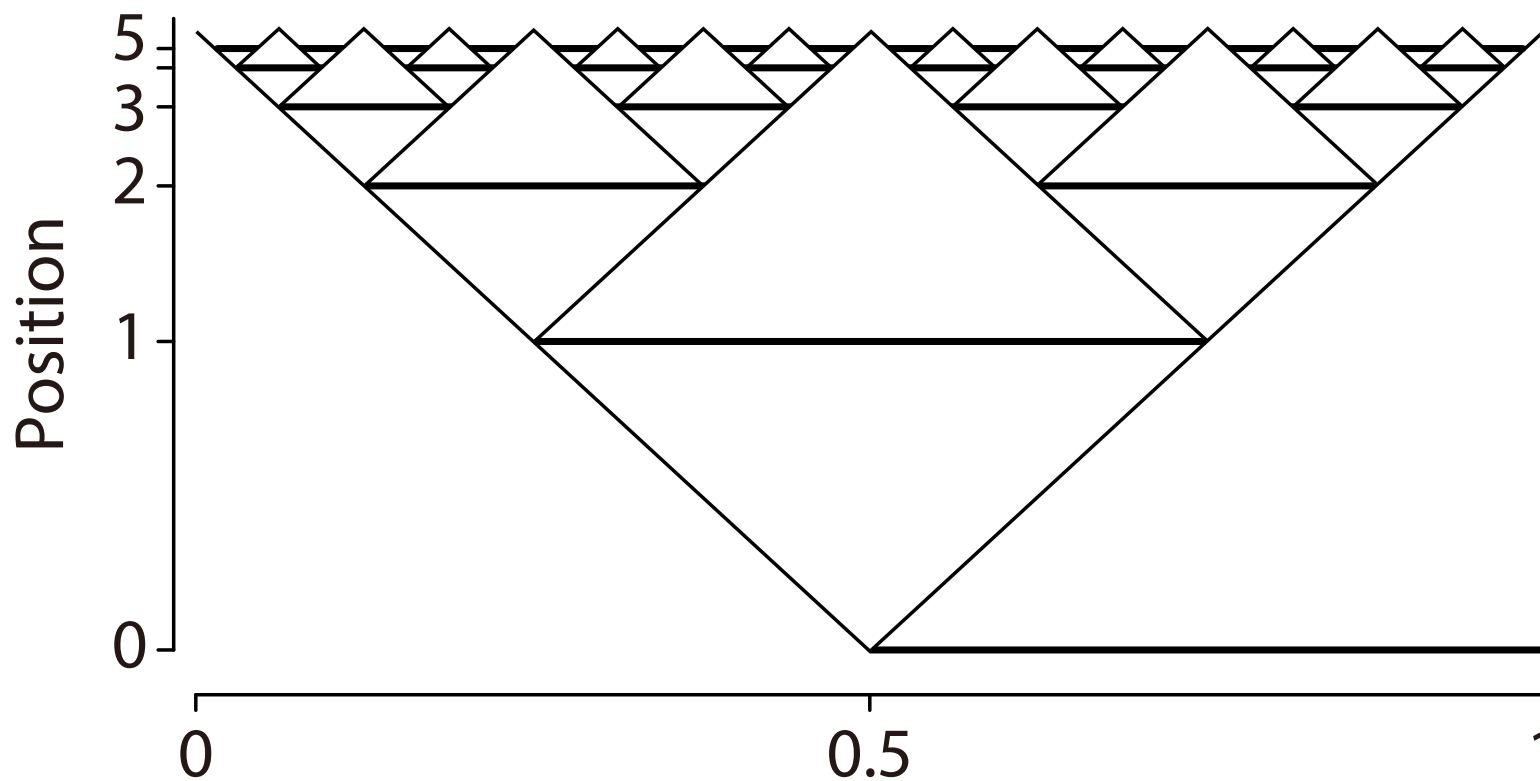
- Using signed digit representation, we can achieve computation over reals **in a natural sense**
- Another interesting representation is **Gray code** (グレイコード) by Frank Gray (1947) and Émile Baudot (1878)
 - Originally, another binary encoding of natural numbers
 - Important in applications of conversion between analog and digital information [Knuth, 2005]
- Gray codes for natural numbers:

	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111
Gray	000	001	011	010	110	111	101	100

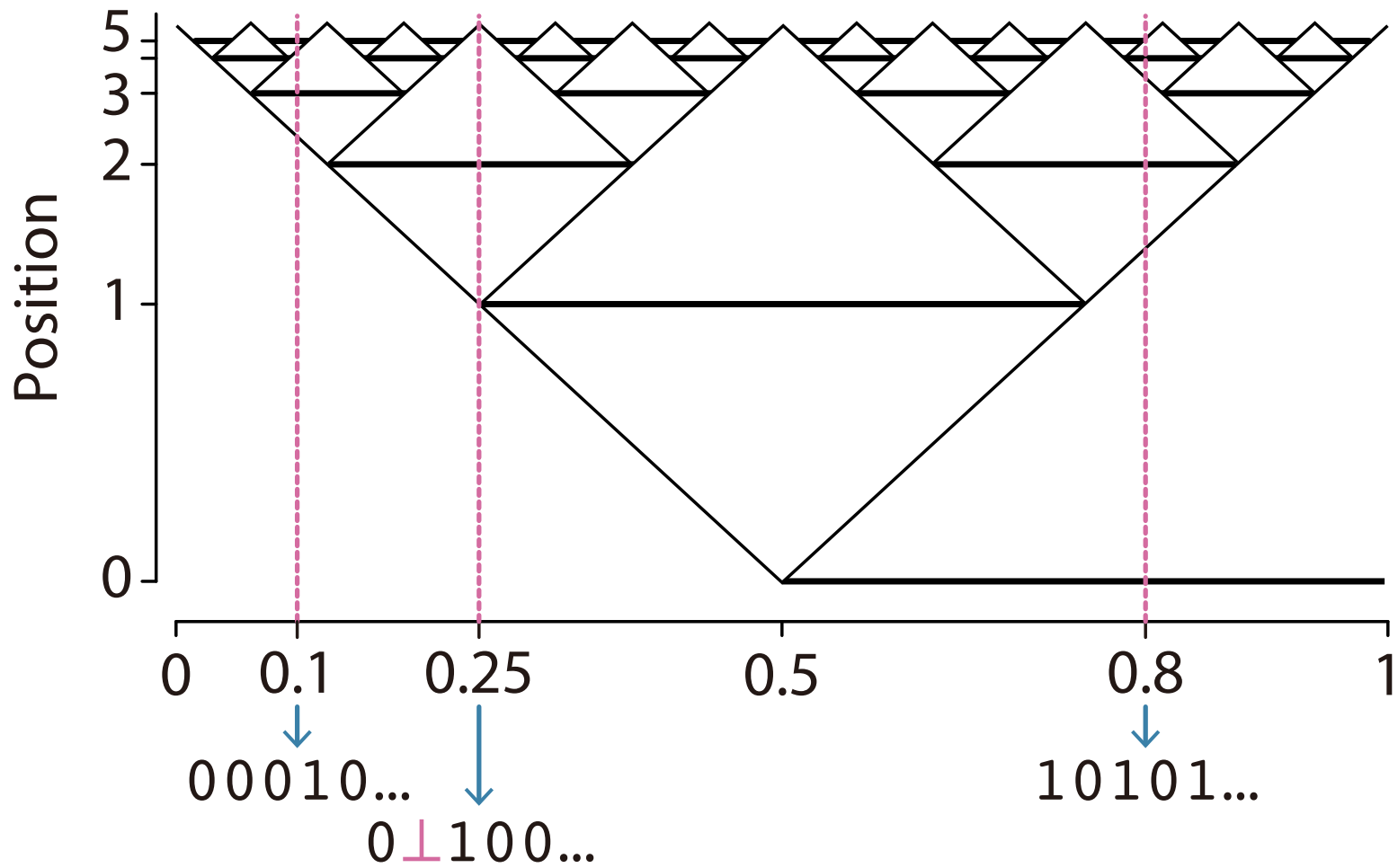
Gray Code Embedding

- Gray code can be used for real number representation
 - We use three symbols 0, 1, and \perp
- The **Gray code embedding** (グレイコード埋め込み) is an injection γ_G that maps $x \in [0, 1]$ to an infinite sequence $p_0 p_1 p_2 \dots$, where
 - $p_i := 1$ if $2^{-i} m - 2^{-(i+1)} < x < 2^{-i} m + 2^{-(i+1)}$ for an odd m ,
 - $p_i := 0$ if the same holds for an even m ,
 - $p_i := \perp$ if $x = 2^{-i} m - 2^{-(i+1)}$ for some integer m
- Power of representations for real number computation:
Gray code = signed digit representation [Dusky, 2002]
> binary representation

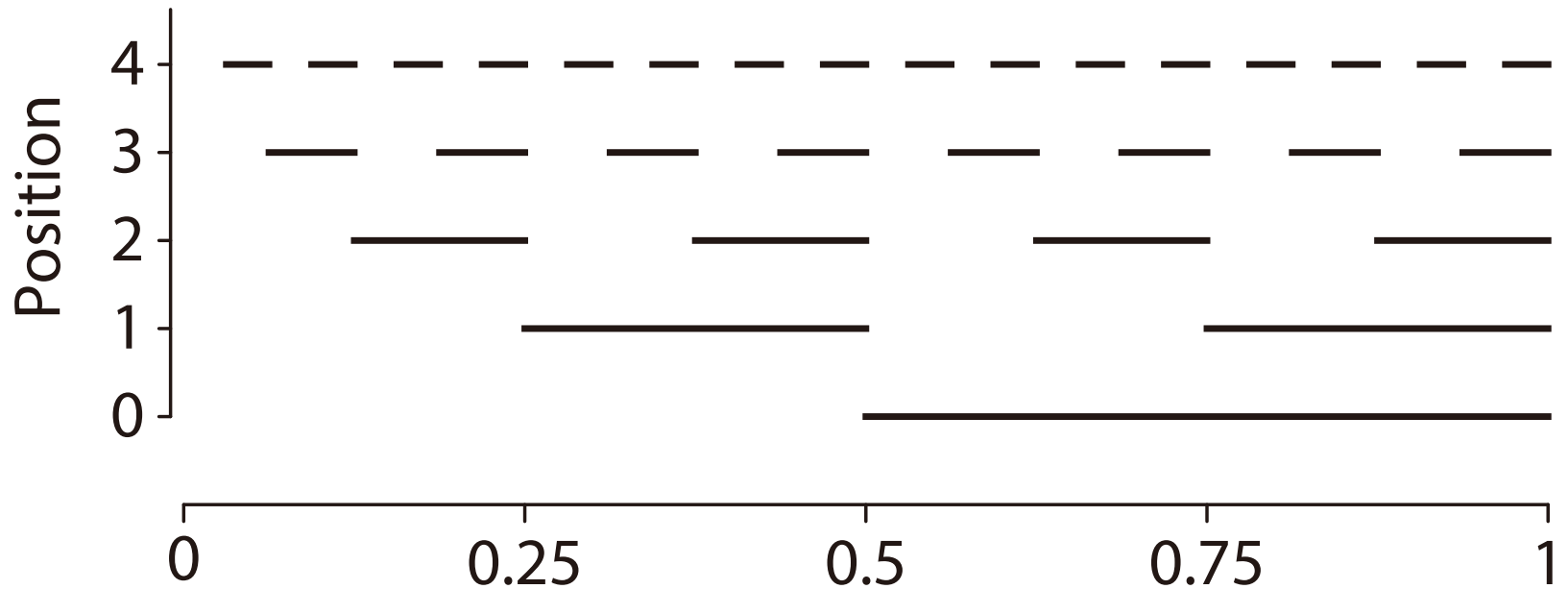
Gray Code on Reals



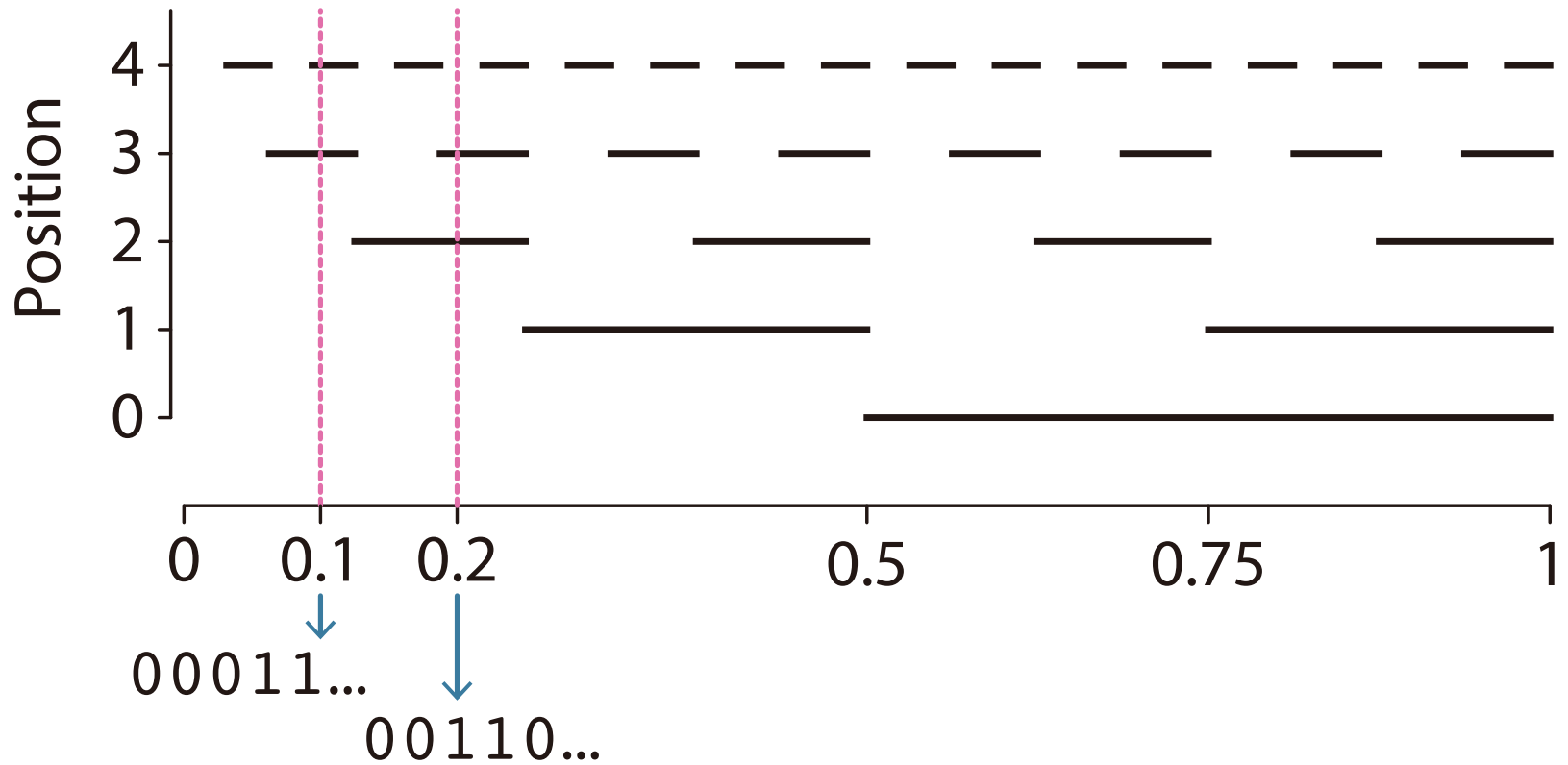
Gray Code on Reals



Binary Representation



Binary Representation

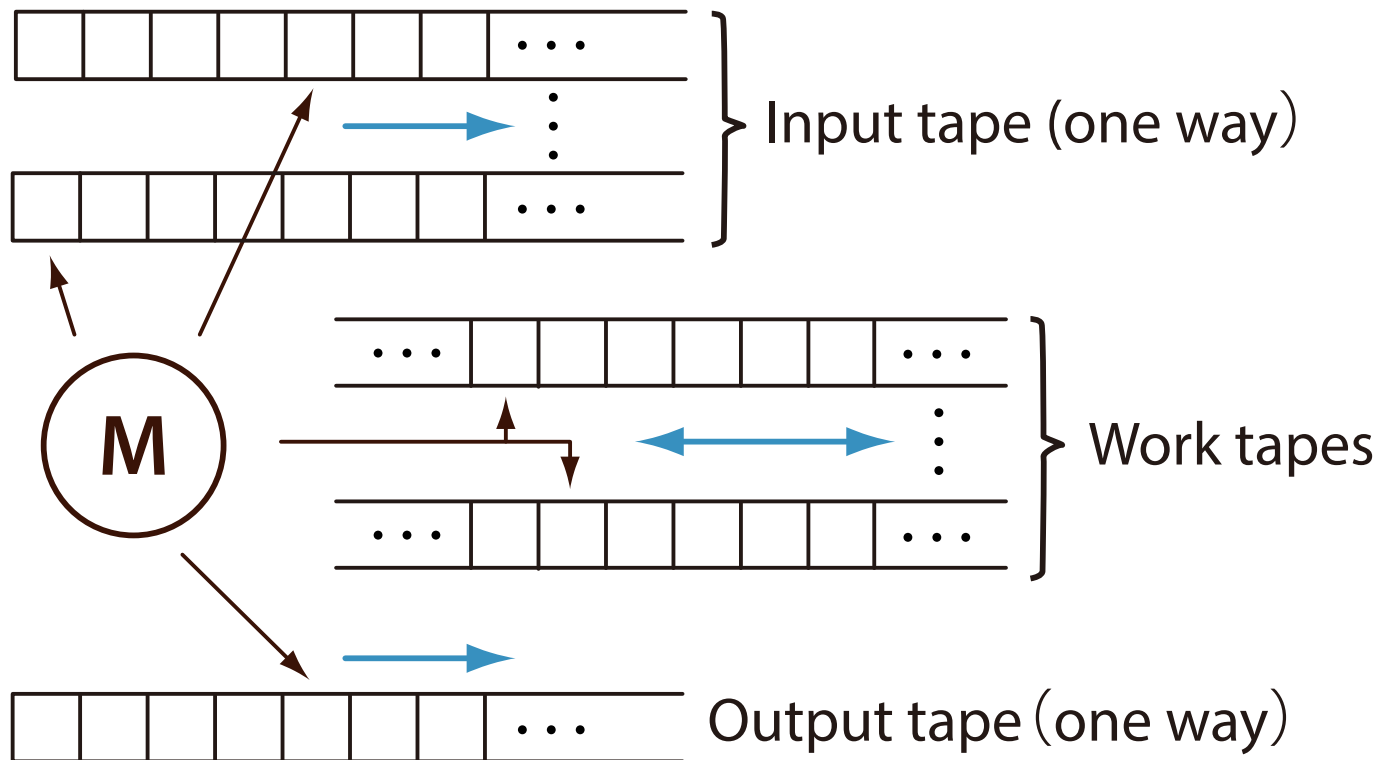


Computation via Type-2 Machine

- Computation of real numbers is realized as conversion between their representations (infinite sequences)
- Computation on infinite sequences in Σ^ω is formulated using **Type-2 machine**

$$\begin{array}{ccc} \Sigma^\omega & \xrightarrow{g} & \Sigma^\omega \\ \xi \downarrow & & \downarrow \zeta \\ X & \xrightarrow{f} & Y \end{array}$$

Type-2 Machine



Discretization and Learning

- In finite time, a computer (Type-2 machine) receives a finite **prefix** (接頭辞) of an infinite sequence that represents a real number
 - The input is thus **discretized** (離散化)
- A computer continues to output succeeding digits of output which is getting closer to the true value

Discretization and Learning

- In finite time, a computer (Type-2 machine) receives a finite **prefix** (接頭辞) of an infinite sequence that represents a real number
 - The input is thus **discretized** (離散化)
- A computer continues to output succeeding digits of output which is getting closer to the true value
- *This is similar to the mechanism of **learning***
 - Discretized approximation (in **computing**)
 - Partial information of concepts (in **learning**)

Discretization and Learning

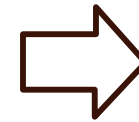
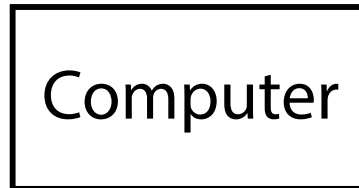
Partial information
of the target

Approximation
of the result

Discretized
real number



Computing

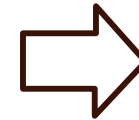


Discretized
real number

Data



Learning



Hypotheses

Real Number Computation as Learning

- **Concept** (learning target): a real number $x \in \mathbb{R}$
- **Hypothesis**: a finite sequence $H = a_1 a_2 \dots a_k$
 - A hypothesis H represents an interval $u(H)$
- **Data**: prefixes of $x = \rho(a_1 a_2 \dots)$
- **Correctness**:
 - **Consistency**: H is always consistent with x , i.e., $x \in u(H)$
 - Instead of convergence in identification in the limit, we have **effectivity**:
For a sequence of hypotheses w_1, w_2, w_3, \dots ,
 $u(w_i) \supseteq u(w_{i+1})$ always holds

Summary of Real Number Computation in Machine Learning Framework

Target	Real number
Representation	Gray code/signed digit representation
Data	Prefix (Discretized value, interval)
Algorithm	Depends on functions
Correctness	Consistency & Effectivity

Example: Binary Representation

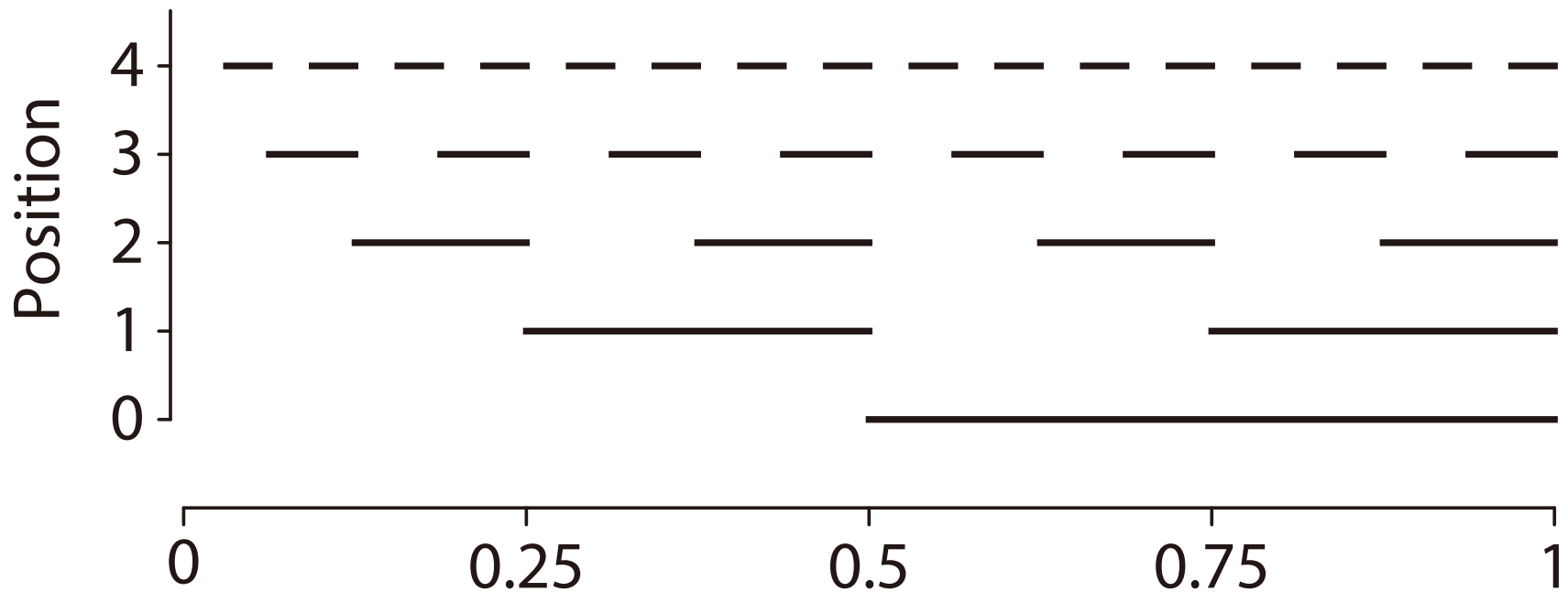
- $\Sigma = \{0, 1\}$
- Binary representation $\rho : \Sigma^\omega \rightarrow [0, 1]$:

$$\rho(a_1 a_2 \dots) = \sum_{i=1}^{\infty} a_i \cdot 2^{-i}$$

- Binary representation for finite sequence
 $v : \Sigma^* \rightarrow \mathcal{P}([0, 1])$:

$$\begin{aligned} v(a_1 a_2 \dots a_k) &= [\rho(a_1 a_2 \dots a_k 000 \dots), \rho(a_1 a_2 \dots a_k 111 \dots)] \\ &= \left[\sum_{i=1}^k a_i \cdot 2^{-i}, \sum_{i=1}^k a_i \cdot 2^{-i} + 2^{-k} \right] \end{aligned}$$

Binary Representation



Example: Signed Digit Representation

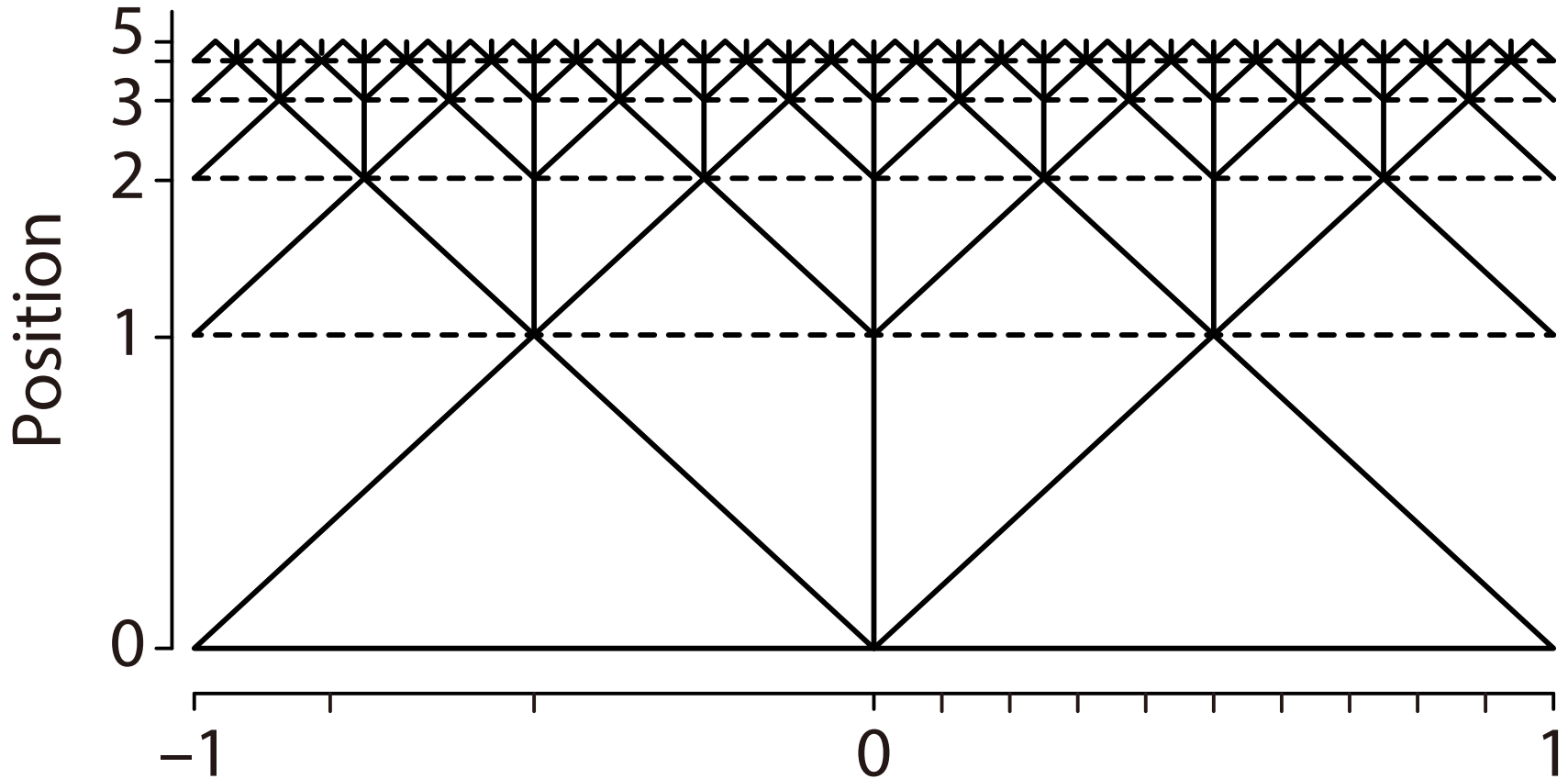
- $\Sigma = \{0, 1, \bar{1}\}$
- Signed digit representation $\rho : \Sigma^\omega \rightarrow [0, 1]$:

$$\rho(a_1 a_2 \dots) = \sum_{i=1}^{\infty} a_i \cdot 2^{-i}$$

- Signed digit representation for finite sequence
 $v : \Sigma^* \rightarrow \mathcal{P}([0, 1])$:

$$\begin{aligned} v(a_1 a_2 \dots a_k) &= [\rho(a_1 a_2 \dots a_k \bar{1} \bar{1} \bar{1} \dots), \rho(a_1 a_2 \dots a_k 1 1 1 \dots)] \\ &= \left[\sum_{i=1}^k a_i \cdot 2^{-i} - 2^{-k}, \sum_{i=1}^k a_i \cdot 2^{-i} + 2^{-k} \right] \end{aligned}$$

Signed Digit Representation



Refinement

- Refinement of signed digit representation is simple:
 1. $w \xrightarrow{\rho} w0$
 2. $w \xrightarrow{\rho} w1$
 3. $w \xrightarrow{\rho} w\bar{1}$
- “Learning with refinement” = “Real number computation”

Efficient Learning with Refinement

1. $i \leftarrow 1, S \leftarrow \emptyset, H \leftarrow T, Q \leftarrow \emptyset$ // Q is a list of candidate hypotheses
2. repeat
3. $S \leftarrow S \cup \{(x_i, y_i)\}$
4. while H is not consistent with S
5. if $x \in v(H)$ for some $(x, o) \in S$ then
6. Append all $\rho(H)$ to the tail of Q
7. end if
8. $H \leftarrow$ the first hypothesis in Q , and remove it from Q
9. end while
10. $H_i \leftarrow H$ and output H_i
11. $i \leftarrow i + 1$
12. until forever

Conclusion

- Computing and learning have been studied in different fields
- However, if we consider computation over \mathbb{R} , there is a close connection between computing and learning
- This is still a developing field
 - No textbook!
 - Some interesting papers:
 - de Brecht, M., **Topological and Algebraic Aspects of Algorithmic Learning Theory**, *PhD thesis* (2010)
 - Sugiyama, M. and Hirowatari, E. and Tsuiki, H. and Yamamoto, A., **Learning Figures with the Hausdorff Metric by Fractals—Towards Computable Binary Classification**, *Machine Learning* (2012)

Take-Home Messages

1. Learning \simeq Computing on \mathbb{R} \neq Computing on \mathbb{N}
2. Representation of objects is essential
3. Structure of hypothesis space is crucial for efficiency
4. We are learners in data mining