October 20, 2017

Inter-University Research Institute Corporation / Research Organization of Information and Systems
National Institute of Informatics

Learning and Computing

Data Mining (データマイニング)

Mahito Sugiyama (杉山麿人)

Objective of Today's Lecture

- ***Learn a fundamental mechanism of machine learning***
  - Machine learning is a core process in many applications in data mining
- Computational aspects of machine learning are mainly discussed
- Key issues:
  - Computing (single) vs Learning (double)
    - Finite/infinite
  - Learning targets (mathematical objects) vs Representations (programs)

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7, . . .
  - What are succeeding numbers?

Example of Learning from Data
(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- 1, 2, 4, 7, . . .
  - What are succeeding numbers?

$$1, 2, 4, 7, 11, 16, \ldots \qquad (a_n = a_{n-1} + n - 1)$$

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- $1, 2, 4, 7, \ldots$
  - What are succeeding numbers?

$$1, 2, 4, 7, 11, 16, \ldots \quad (a_n = a_{n-1} + n - 1)$$

$$1, 2, 4, 7, 12, 20, \ldots \quad (a_n = a_{n-1} + a_{n-2} + 1)$$

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- $1, 2, 4, 7, \ldots$
  - What are succeeding numbers?

$1, 2, 4, 7, 11, 16, \ldots \quad (a_n = a_{n-1} + n - 1)$

$1, 2, 4, 7, 12, 20, \ldots \quad (a_n = a_{n-1} + a_{n-2} + 1)$

$1, 2, 4, 7, 13, 24, \ldots \quad (a_n = a_{n-1} + a_{n-2} + a_{n-3})$

Example of Learning from Data
(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- $1, 2, 4, 7, \ldots$
  - What are succeeding numbers?

$1, 2, 4, 7, 11, 16, \ldots$	$(a_n = a_{n-1} + n - 1)$
$1, 2, 4, 7, 12, 20, \ldots$	$(a_n = a_{n-1} + a_{n-2} + 1)$
$1, 2, 4, 7, 13, 24, \ldots$	$(a_n = a_{n-1} + a_{n-2} + a_{n-3})$
$1, 2, 4, 7, 14, 28$	(divisors of 28)

Example of Learning from Data
(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- $1, 2, 4, 7, \ldots$
  - What are succeeding numbers?

$1, 2, 4, 7, 11, 16, \ldots$	$(a_n = a_{n-1} + n - 1)$
$1, 2, 4, 7, 12, 20, \ldots$	$(a_n = a_{n-1} + a_{n-2} + 1)$
$1, 2, 4, 7, 13, 24, \ldots$	$(a_n = a_{n-1} + a_{n-2} + a_{n-3})$
$1, 2, 4, 7, 14, 28$	(divisors of 28)
$1, 2, 4, 7, 1, 1, 5, \ldots$	(decimals of $\pi = 3.1415\ldots$, $e = 2.718\ldots$)

Example of Learning from Data

(from mlss.tuebingen.mpg.de/2013/schoelkopf_whatisML_slides.pdf)

- $1, 2, 4, 7, \ldots$
  - What are succeeding numbers?

$1, 2, 4, 7, 11, 16, \ldots$ $\quad (a_n = a_{n-1} + n - 1)$

$1, 2, 4, 7, 12, 20, \ldots$ $\quad (a_n = a_{n-1} + a_{n-2} + 1)$

$1, 2, 4, 7, 13, 24, \ldots$ $\quad (a_n = a_{n-1} + a_{n-2} + a_{n-3})$

$1, 2, 4, 7, 14, 28$ $\quad$ (divisors of 28)

$1, 2, 4, 7, 1, 1, 5, \ldots$ $\quad$ (decimals of $\pi = 3.1415\ldots$, $e = 2.718\ldots$)

- 1107 results (!) in the online encyclopedia (`https://oeis.org/`)

Analyze Learning as Scientific Problem

- Which is the correct answer (or generalization) for succeeding numbers of $1, 2, 4, 7, \ldots$?
  - Any answer is possible!

Analyze Learning as Scientific Problem

- Which is the correct answer (or generalization) for succeeding numbers of $1, 2, 4, 7, \dots$?
  - Any answer is possible!

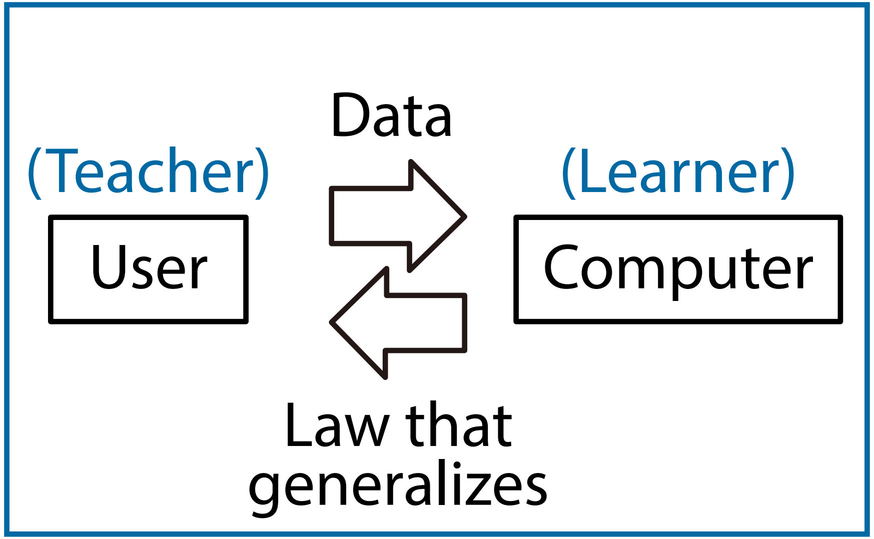
- We should take two points into consideration:

  (i) We need to formalize the problem of "learning"
    - There are two agents (teacher and learner) in learning, which are different from "computation"

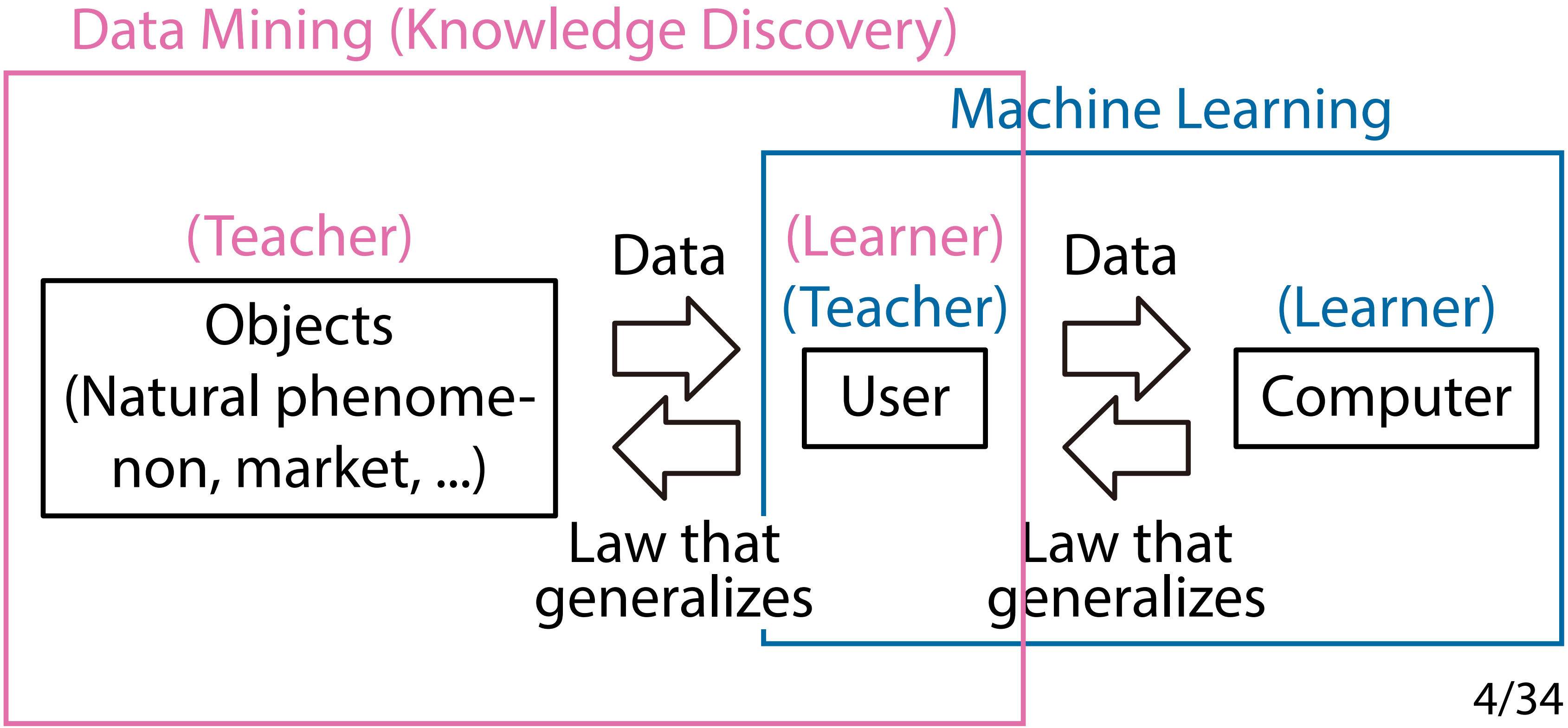
  (ii) Learning is an infinite process
    - A learner usually never knows that the current hypothesis is perfectly correct

Framework of Learning (ML vs DM)

Machine Learning



Framework of Learning (ML vs DM)



Computation — Core Engine of Learning/Mining

- Machine learning/data mining is usually achieved using a computer
- Computing behavior is mathematically formulated by Alan Turing in 1936
  - A. M. Turing, **On Computable Numbers, with the Application to the Entscheidungsproblem**, Proceedings of the London Mathematical Society, 42(1), 230–265, 1937
- The model of computation, known as a Turing machine, is developed for simulating computation by human beings

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions

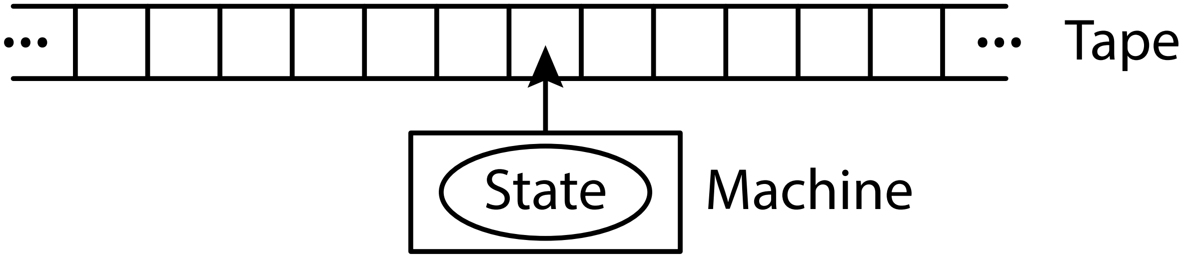
Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent†. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols. Thus an Arabic numeral such as

† If we regard a symbol as literally printed on a square we may suppose that the square is $0 \leqslant x \leqslant 1$, $0 \leqslant y \leqslant 1$. The symbol is defined as a set of points in this square, viz. the set occupied by printer's ink. If these sets are restricted to be measurable, we can define the "distance" between two symbols as the cost of transforming one symbol into the other if the cost of moving unit area of printer's ink unit distance is unity, and there is an infinite supply of ink at $x = 2$, $y = 0$. With this topology the symbols form a conditionally compact space.

17 or 999999999999999 is normally treated as a single symbol. Similarly in any European language words are treated as single symbols (Chinese, however, attempts to have an enumerable infinity of symbols). The differences from our point of view between the single and compound symbols is that the compound symbols, if they are too lengthy, cannot be observed at one glance. This is in accordance with experience. We cannot tell at a glance whether 9999999999999999 and 999999999999999 are the same.

The behaviour of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment. We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations. We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused. Again, the restriction

Turing Machine



- The machine repeats the following:
  - Read a symbol *a* of a cell
  - Do the following from *a* and the current state *s* according to a set of rules in its memory
    - Replace the symbol *a* at the square
    - Move the head
    - Change the state *s*

Computing vs Learning

- In ***computation***, the process is completed on its own
  - No interaction
    - The Turing machine automatically works according to programmed rules
  - A finite process

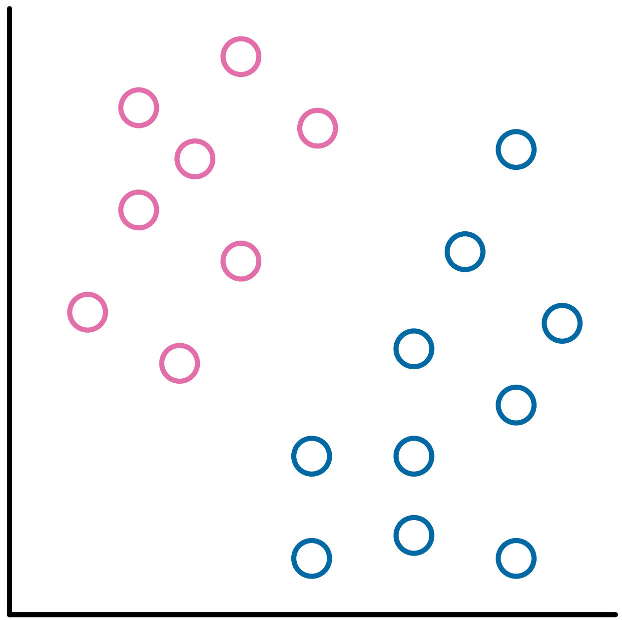
Computing vs Learning

- In ***computation***, the process is completed on its own
  - No interaction
    - The Turing machine automatically works according to programmed rules
  - A finite process
- In ***learning***, there are two agents (teacher and learner)
  - Interaction between agents should be considered
    - A learning protocol between a teacher and a learne r is essentially needed
  - An infinite process

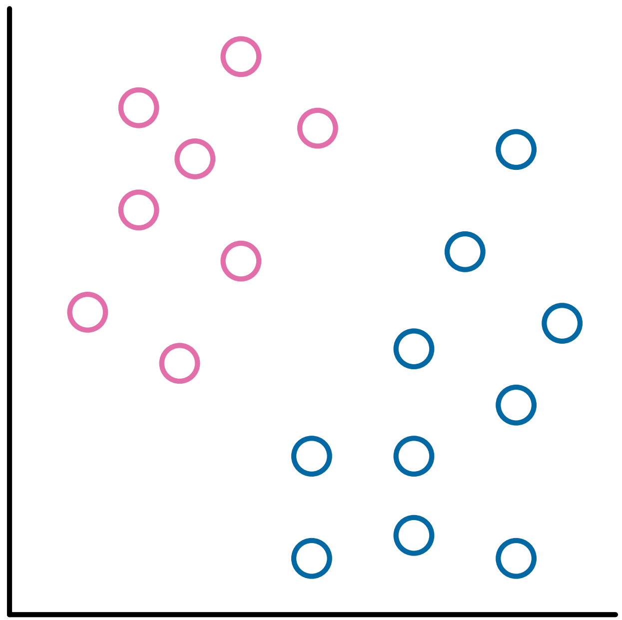
Formalize Learning in Computational Manner

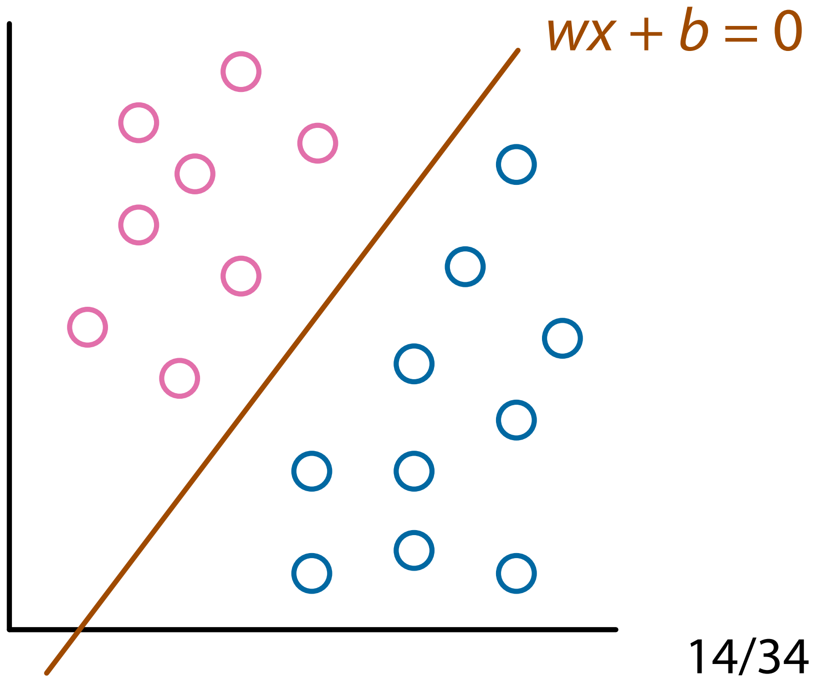
1. What are targets of learning?
2. How to represent targets and hypotheses?
3. How are data provided to a learner?
4. How does the learner work?
5. When can we say that the learner correctly learns the target?

Learning of Binary Classifier



Learning of Binary Classifier





Example: Perceptron (by F. Rosenblatt, 1958)

- **Learning target**: two subsets $F, G \subseteq \mathbb{R}^d$ s.t. $F \cap G = \varnothing$
  - Assumption: F and G are linearly separable
    - There exists a function (classifier) $f_*(x) = w_* x + b$ s.t.
      $f_*(x) > 0 \quad \forall x \in F, \qquad f_*(x) < 0 \quad \forall x \in G$

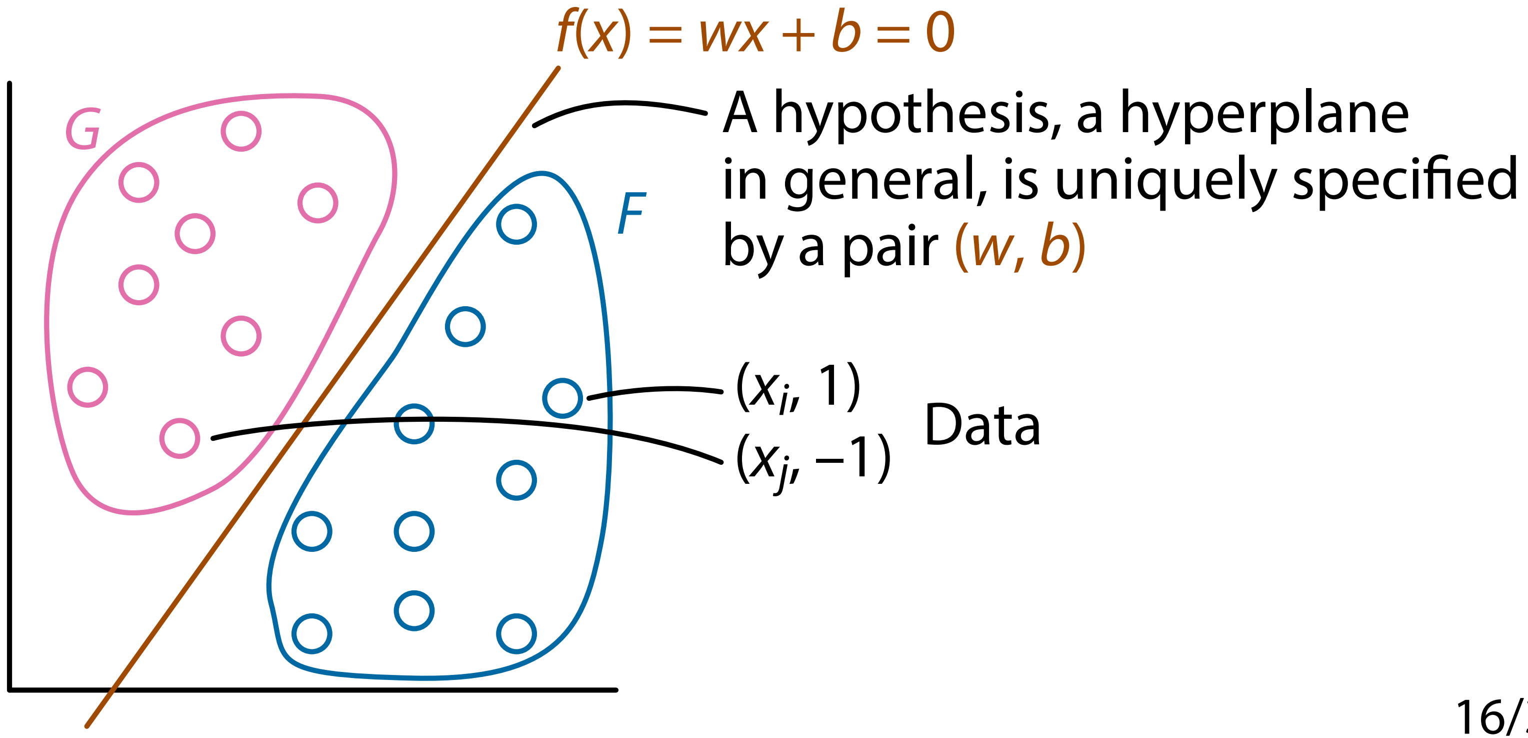
Example: Perceptron (by F. Rosenblatt, 1958)

- **Learning target**: two subsets $F, G \subseteq \mathbb{R}^d$ s.t. $F \cap G = \varnothing$
  - Assumption: F and G are linearly separable
    - There exists a function (classifier) $f_*(x) = w_* x + b$ s.t.
      $f_*(x) > 0 \quad \forall x \in F, \qquad f_*(x) < 0 \quad \forall x \in G$
- **Hypotheses**: hyperplanes on $\mathbb{R}^d$
  - If we consider a linear equation $f(x) = wx + b$, each line can be uniquely specified by a pair of two parameters (w, b) (hypothesis)

Example: Perceptron (by F. Rosenblatt, 1958)

- **Learning target**: two subsets $F, G \subseteq \mathbb{R}^d$ s.t. $F \cap G = \emptyset$
  - Assumption: F and G are linearly separable
    - There exists a function (classifier) $f_*(x) = w_* x + b$ s.t. $f_*(x) > 0 \quad \forall x \in F, \qquad f_*(x) < 0 \quad \forall x \in G$
- **Hypotheses**: hyperplanes on $\mathbb{R}^d$
  - If we consider a linear equation $f(x) = wx + b$, each line can be uniquely specified by a pair of two parameters (w, b) (hypothesis)
- **Data**: a sequence of pairs $(x_1, y_1), (x_2, y_2), \dots$
  - (x_i, y_i): (a real-valued vector in $\mathbb{R}^d$, a label)
  - $x_i \in F \cup G$, $y_i \in \{1, -1\}$, and $y_i = 1$ $(y_i = -1)$ if $x_i \in F$ $(x_i \in G)$

Learning Model for Perceptron



Learning Procedure of Perceptron

1. $w \leftarrow 0, b \leftarrow 0$ (or a small random value) // initialization
2. for $i = 1, 2, 3, \ldots$ do
3. Receive i-th pair (x_i, y_i)
4. Compute $a = \sum_{j=1}^{d} w^j x_i^j + b$
5. if $y_i \cdot a < 0$ then // x_i is misclassified
6. $w \leftarrow w + y_i x_i$ // update the weight
7. $b \leftarrow b + y_i$ // update the bias
8. end if
9. end for

Correctness of Perceptron

- It is guaranteed that a perceptron always converges to a correct classifier
  - A correct classifier is a function f s.t.

    $$\begin{aligned} f(x) > 0 \quad & \forall x \in F, \\ f(x) < 0 \quad & \forall x \in G \end{aligned}$$

  - The convergence theorem

- Note: there are (infinitely) many functions that correctly classify F and G
  - A perceptron converges to one of them

Summary: Perceptron

Target	Two disjoint subsets of $\mathbb{R}^d$
Representation	Two parameters (w, b) of linear equation $f(x) = wx + b$
Data	Real vectors from target subsets
Algorithm	Perceptron
Correctness	Convergence theorem

Example 2: Maximum Likelihood Estimation

- Estimate the probability of a coin being a head in a toss

Target	Bernoulli distribution
Representation	Parameter (probability) p
Data	Sampling
Algorithm	Maximum Likelihood Estimation
	$\hat{p} = k/n$
Correctness	Consistency

Basic Definitions of Learning

- Target: a classifier $f_* : X \to \{0, 1\}$
  - A class $\mathcal{C}$ of classifiers is usually pre-determined
  - Each target can be viewed as the set $F_* = \{a \in X \mid f_*(a) = 1\}$
    - F_* is called a concept

Basic Definitions of Learning

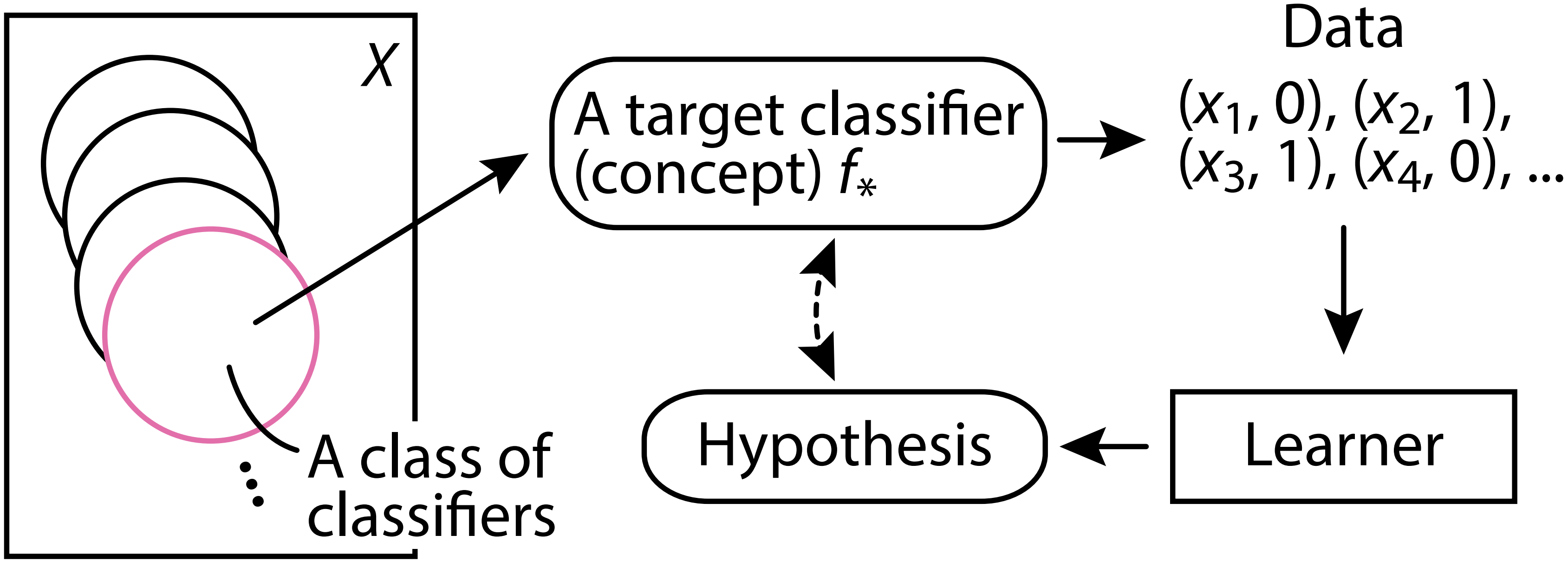
- Target: a classifier $f_* : X \to \{0, 1\}$
  - A class $\mathcal{C}$ of classifiers is usually pre-determined
  - Each target can be viewed as the set $F_* = \{a \in X \mid f_*(a) = 1\}$
    - F_* is called a concept

- Hypothesis space: $\mathcal{R}$
  - Each hypothesis $H \in \mathcal{R}$ represents a classifier
  - $\mathcal{R} \subseteq \Sigma^*$ usually holds (Σ^* is the set of finite strings)

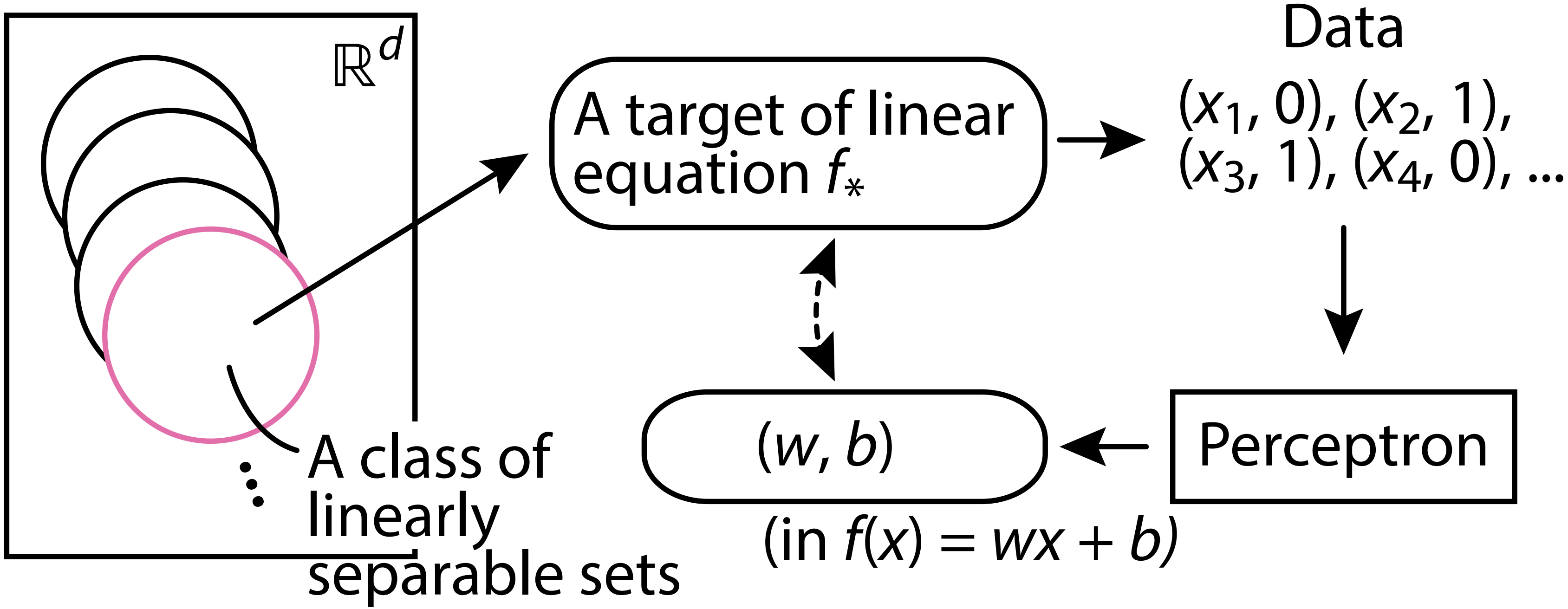
Basic Definitions of Learning

- Target: a classifier $f_* : X \to \{0, 1\}$
  - A class $\mathcal{C}$ of classifiers is usually pre-determined
  - Each target can be viewed as the set $F_* = \{a \in X \mid f_*(a) = 1\}$
    - F_* is called a concept
- Hypothesis space: $\mathcal{R}$
  - Each hypothesis $H \in \mathcal{R}$ represents a classifier
  - $\mathcal{R} \subseteq \Sigma^*$ usually holds (Σ^* is the set of finite strings)
- Data: Example $(a, f_*(a))$
  - $a \in X$
  - An example $(a, 1)$ is called positive, $(a, 0)$ is called negative

Learning Model



Learning Model (e.g. Perceptron)



Gold's Learning Model (Identification in the Limit)

- Gold gave the first basic learning model, called "Identification in the limit"
  - E. M. Gold, **Language identification in the limit**, Information and Control, 10(5), 447–474, 1967
- This model was originally introduced to analyze the learnability of formal languages
  - His motivation was to model infant's learning process of natural languages

INFORMATION AND CONTROL 10, 447–474 (1967)

Language Identification in the Limit

E MARK GOLD*

The RAND Corporation

Language learnability has been investigated. This refers to the following situation: A class of possible languages is specified, together with a method of presenting information to the learner about an unknown language, which is to be chosen from the class. The question is now asked, "Is the information sufficient to determine which of the possible languages is the unknown language?" Many definitions of learnability are possible, but only the following is considered here: Time is quantized and has a finite starting time. At each time the learner receives a unit of information and is to make a guess as to the

Formal Languages

- Alphabet Σ: a nonempty finite set
  - Each element $a \in \Sigma$ is called a symbol
- Word $w = a_1 a_2 \ldots a_n$: a finite sequence of symbols
  - Null word ε, whose length is 0
- The set of words Σ^* (with ε) and Σ^+ (without ε)

  $$\Sigma^* = \{ a_1 a_2 \ldots a_n \mid a_i \in \Sigma, n \geq 0 \}$$

  $$\Sigma^+ = \{ a_1 a_2 \ldots a_n \mid a_i \in \Sigma, n \geq 1 \} = \Sigma^* \setminus \{\varepsilon\}$$
- Formal language: a subset of Σ^*

Representation of Languages

- We connect syntax and semantics using a mapping f
- For a hypothesis $H \in \mathcal{R}$, $f(H, w)$ is 0 or 1 for $w \in \Sigma^*$
  - H is a program of a classifier
  - w is a (binary) code of the input to H
- $L(H) = \{ w \in \Sigma^* \mid f(H, w) = 1 \}$
- $\mathcal{R}$ is usually a recursively enumerable set
  - There is an algorithm that enumerates all elements of $\mathcal{R}$
  - $\mathcal{R}$ is often identified with $\mathbb{N}$
    - Each natural number encodes a classifier (hypothesis)

Setting of Gold's Learning Model

- A class of languages $\mathcal{C} \subseteq \{ A \mid A \subseteq \Sigma^* \}$ is chosen
- For a language $L \in \mathcal{C}$, an infinite sequence $\sigma = (x_1, y_1), (x_2, y_2), \ldots$ is a complete presentation of L if
  (i) $\{x_1, x_2, \ldots\} = \Sigma^*$
  (ii) $y_i = 1 \iff x_i \in L$ for all i
  – $\sigma[i] = (x_1, y_1), \ldots, (x_i, y_i)$ (a prefix of σ)
- A learner is a procedure M that receives σ and generates an infinite sequence of hypotheses $\gamma = H_1, H_2, \ldots$
  – M outputs H_i if it gets $\sigma[i]$

Identification in the Limit

- If γ converges to some hypothesis H and H represents L,
  we say that M identifies L in the limit

- If M identifies any $L \in \mathcal{C}$ in the limit,
  we say that M identifies $\mathcal{C}$ in the limit

Basic Strategy: Generate and Test

- Input: a complete presentation σ of a language L
- Output: $\gamma = H_1, H_2, \ldots$

1. $i \leftarrow 1, S \leftarrow \varnothing$
2. Repeat
3. $\quad S \leftarrow S \cup \{(x_i, y_i)\}$
4. $\quad k \leftarrow \min\{\, j \in \mathbb{N} \mid L(H(j))$ consistent with $S\,\}$
5. $\quad\quad$ // $H(j)$ is a hypothesis encoded by a natural number j
6. $\quad H_i \leftarrow H(k)$ and output H_i
7. $\quad i \leftarrow i + 1$
8. until forever

Power of Generate and Test Strategy

- 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 $L \in \mathcal{C}$ in the limit
- Unfortunately, this strategy is very inefficient
  - More intelligent strategy can be designed for each learning target
  - <u>One of the most important tasks in studies of machine learning!</u>

Learning from Positive Data

- In many cases, in particular in data mining,
  we obtain only positive data
  - Imagine supervised vs unsupervised learning

- A positive presentation of a language $L \in \mathcal{C}$
  is an infinite sequence $x_1, x_2, \ldots$ s.t. $L = \{x_1, x_2, \ldots\}$

- If γ (an infinite sequence of hypotheses of a learner M) converges to a hypothesis H s.t. $L(H) = L$,
  we say that M identifies L in the limit from positive data

Limitation of Learning from Positive Data

- Consider the following class $\mathcal{C}$
  (i) All finite languages are included in $\mathcal{C}$
  (ii) At least one infinite language is included in $\mathcal{C}$
  – $\mathcal{C}$ is called superfinite

- Gold proved that a superfinite class cannot be learned from positive data
  – e.g. $\Sigma = \{a\}$, $\mathcal{C}$ contains all finite languages and $\{a^n \mid n \geq 1\}$

- Although this fact shows a limitation, there still exist rich classes of interesting languages
  – For example, pattern language

References

- If you are interested in computational learning theory, the following books might be interesting:
  - 榊原康文, 横森貴, 小林聡, 計算論的学習, 培風館, 2001
  - S. Jain, D. N. Osherson, J. S. Royer, A. Sharma, Systems That Learn, A Bradford Book, 1999

- These books are not necessarily for this lecture