

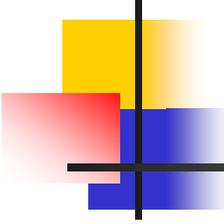
# Introduction to Computational Learning

---

Akihiro Yamamoto 山本 章博

[http://www.iip.ist.i.kyoto-u.ac.jp/member/akihiro/  
akihiro@i.kyoto-u.ac.jp](http://www.iip.ist.i.kyoto-u.ac.jp/member/akihiro/akihiro@i.kyoto-u.ac.jp)

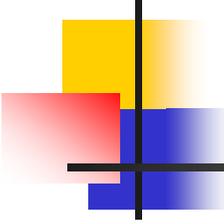
2017



# What is Computational Learning

---

- Computational learning is modeling “learning” in the same way as modeling computation.
- Practical applications of computational learning include learning or knowledge discovery from discrete data:
  - strings: texts, DNA sequences,...
  - trees: parsing trees, XML documents
  - tables: relational data,
  - graphs: ...



# Machine Learning

---

- Machine Learning originally means mechanisms which make machines wiser and wiser by training them more and more.
- Recently Machine Learning also (and mainly) means mechanisms with which we can discover **rules or structures hidden behind data.**
  - “**Make invisible structure be visible**”
  - Sometimes we fail in applying machine learning to a specific purpose because of what type of rules would be discovered.

# Why discrete data in ML(1)?

- We are surrounded by full of strings, sentences, tables,...

The image shows three overlapping web browser windows from the early 2000s. The top-left window is the Japanese Wikipedia page for '余部橋梁' (Yubu Bridge), which is a steel truss bridge in Japan. The top-right window is the Yahoo! Japan homepage. The bottom window is a Google search results page for 'machine learning', showing approximately 72,900,000 results. The search results include links to Wikipedia, academic papers, and other resources related to machine learning.

# Why discrete data in ML?(2)

- Computers work with strings(sequences) consisting of 0 and 1.

Binary	Oct	Dec	Hex	Glyph	Binary	Oct	Dec	Hex	Glyph
100 0000	100	64	40	@	110 0000	140	96	60	`
100 0001	101	65	41	A	110 0001	141	97	61	a
100 0010	102	66	42	B	110 0010	142	98	62	b
100 0011	103	67	43	C	110 0011	143	99	63	c
100 0100	104	68	44	D	110 0100	144	100	64	d
100 0101	105	69	45	E	110 0101	145	101	65	e
100 0110	106	70	46	F	110 0110	146	102	66	f
100 0111	107	71	47	G	110 0111	147	103	67	g
100 1000	110	72	48	H	110 1000	150	104	68	h
100 1001	111	73	49	I	110 1001	151	105	69	i
100 1010	112	74	4A	J	110 1010	152	106	6A	j
100 1011	113	75	4B	K	110 1011	153	107	6B	k
100 1100	114	76	4C	L	110 1100	154	108	6C	l
100 1101	115	77	4D	M	110 1101	155	109	6D	m
100 1110	116	78	4E	N	110 1110	156	110	6E	n
100 1111	117	79	4F	O	110 1111	157	111	6F	o
101 0000	120	80	50	P	111 0000	160	112	70	p
101 0001	121	81	51	Q	111 0001	161	113	71	q
101 0010	122	82	52	R	111 0010	162	114	72	r

[Wikipedia]

# Why sequences in ML?(1)

- Sentences are strings(sequences) consisting of characters in an alphabet.



[Wikipedia]

## ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

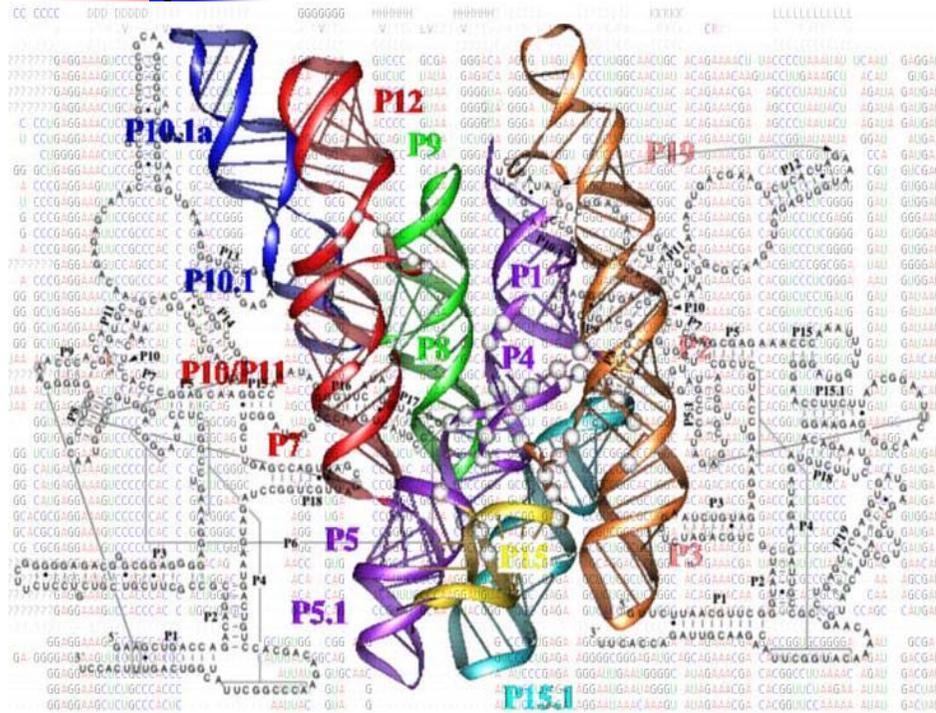
By A. M. TURING.

[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 of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbersome technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of com-

[Davis, M. : *The Undecidable*, Raven Press]

# Why sequences in ML?(2)



- Many data for academic research is now open. In particular, many string data are provided in the area of **bio-informatics**.

5' CACAUGUACAAGACUU 3'

Rfam: RNA families database of alignments and CMs

Version 8.0  
February 2007, 574 families

Enter your keyword(s) here  
  [Example](#)

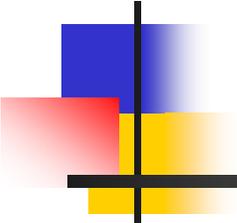
Enter an EMBL name or accession number  
  [Example](#)

Rfam Mirror Servers Worldwide

[Sanger Institute \(UK\)](#)  
[Janelia Farm \(USA\)](#)

FTP access to Rfam

You can also download the Rfam database and for instance search it locally using the [INFERNAL](#) covariance model software. [Hyperlink directly to the ftp site or View ftp site files](#)



# Learning from Numerical Data

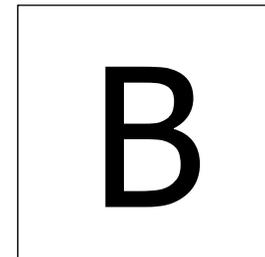
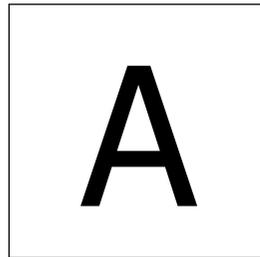
---

# Task

- Every input signal is given to the perceptron with its 'teaching' or 'target' signal which tells 'yes' or 'no'.

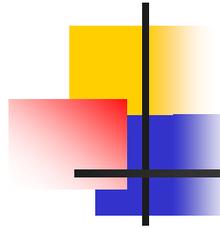
**Example** The target is A

Input

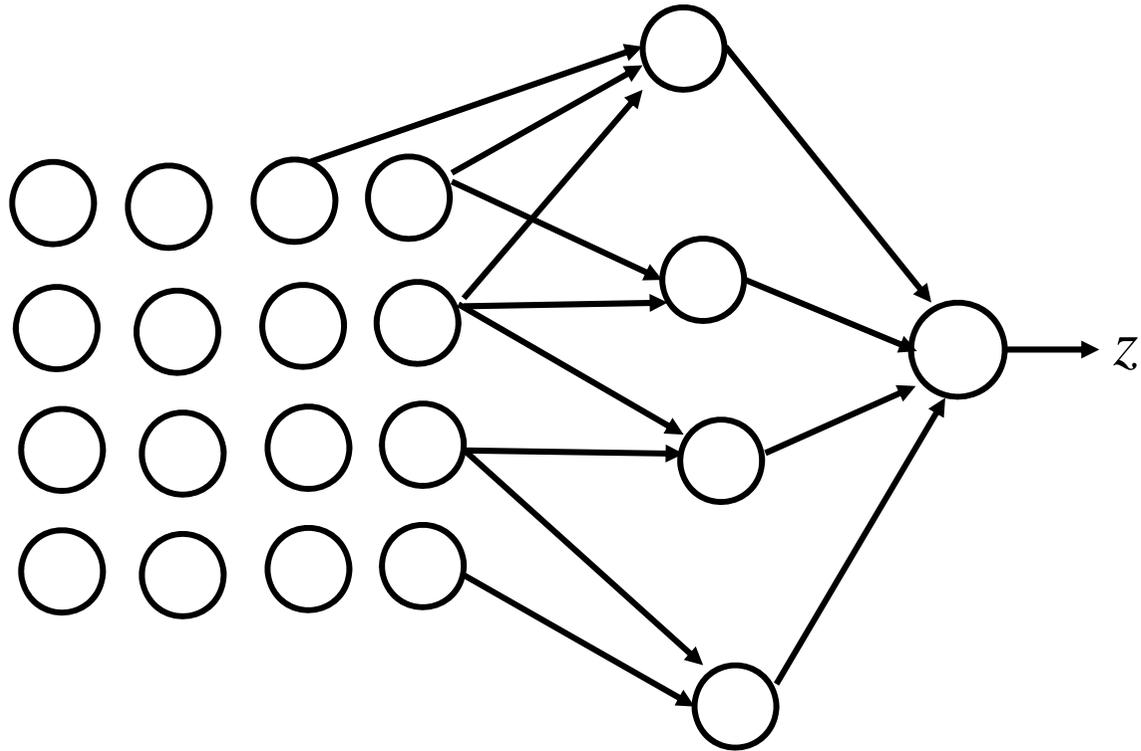
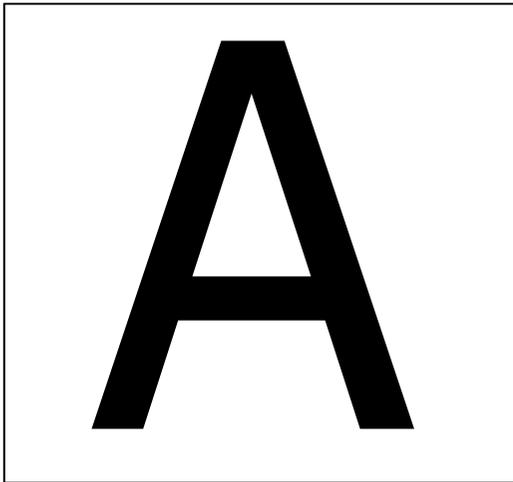


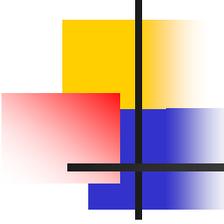
Teaching signal    yes

no



# Learning Machine





# A Simple Learning Method

---

- Revise the weights  $w_i$  according to the combination of the output of the perceptron network and the 'teaching' or 'target' signal.
  - Learning depends on the ways of the revision.
  - The so called “Perceptron Learning” adopt the revision method as follows:

If the output coincides with the teaching signal, do nothing,

otherwise add  $\rho$  to the weights  $w_i$  in the direction to the teaching signal.

# Mathematical Formalization

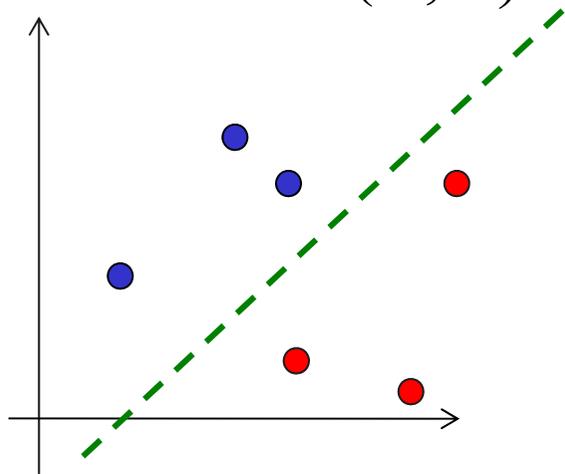
- In order to our discussion simple , we consider classification into two classes.

## Formalization of the Learning Problem

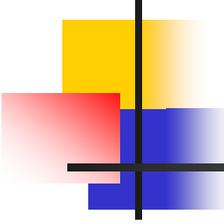
For given two finite subsets  $C$  (yes),  $D$  (no) ( $C \cap D = \emptyset$ ) in  $\mathbf{R}^n$ , find a line  $\mathbf{w}x + c = 0$  which satisfies

$$\mathbf{x} \in C \Rightarrow (\mathbf{w}, \mathbf{x}) + c > 0$$

$$\mathbf{x} \in D \Rightarrow (\mathbf{w}, \mathbf{x}) + c < 0$$



- In order to find  $c$  as well as  $\mathbf{w}$ , we regard every data  $\mathbf{x}$  as  $(\mathbf{x}, 1)$  and the target line as  $(\mathbf{w}', \mathbf{x}) = 0$ .

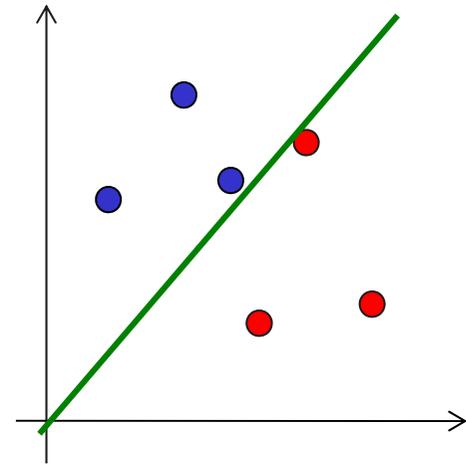
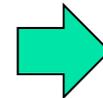
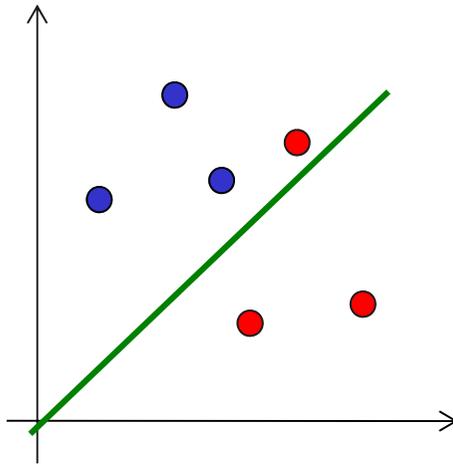
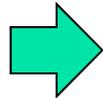
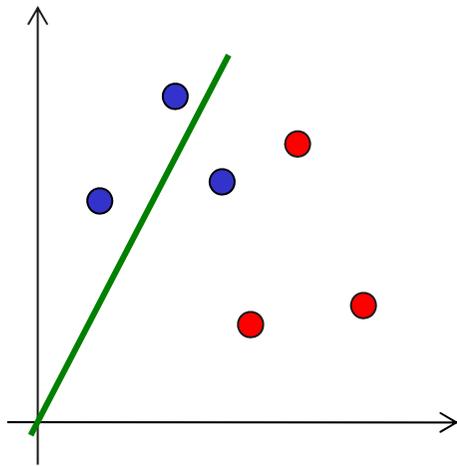


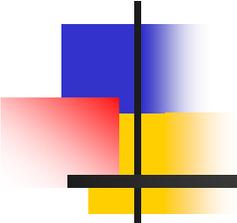
# A Simple Learning Algorithm

---

1. Let the input data  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
2. Initialize  $\mathbf{w}$  as some value.
3. For  $n = 1, 2, \dots, N$ ,
  - if  $\mathbf{x}_n \in C$  and  $(\mathbf{w}, \mathbf{x}_n) < 0$ 
    - replace  $\mathbf{w}$  with  $\mathbf{w} + \rho \mathbf{x}_n$
  - else if  $\mathbf{x}_n \in D$  and  $(\mathbf{w}, \mathbf{x}_n) > 0$ 
    - replace  $\mathbf{w}$  with  $\mathbf{w} - \rho \mathbf{x}_n$
  - otherwise
    - do nothing
4. For  $n = 1, 2, \dots, N$ 
  - if no  $\mathbf{x}_n$  satisfies
    - $(\mathbf{x}_n \in C \text{ and } (\mathbf{w}, \mathbf{x}_n) < 0) \text{ or } (\mathbf{x}_n \in D \text{ and } (\mathbf{w}, \mathbf{x}_n) > 0)$
    - terminates and return  $\mathbf{w}$
  - else
    - go to 3.

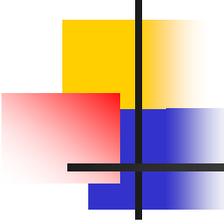
# Example





# Learning from Discrete Data by Embedding

---



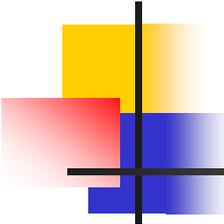
# Two approaches

---

Two approaches can be considered towards learning from discrete data:

- By transforming discrete data into data in  $\mathbf{R}^n$ , in other words, embedding discrete data into  $\mathbf{R}^n$ , and apply learning methods for data in  $\mathbf{R}^n$ .
- By analyzing properties of discrete data, in other words, and using mathematics on discrete data and develop new learning theories and methods for discrete data.

**This course is along the second approach.**



# Example of Transformation (1)

- Let  $D$  be the domain of sequences in English.
- We fix a dictionary (bag of keywords)  $W = (w_1, w_2, \dots, w_k)$ , and define a transformation  $\Phi$  as:

$$\Phi(s) = (x_1, x_2, \dots, x_k) \text{ where}$$

$x_i =$  how many times the keyword  $w_i$  appears in  $s$

for  $i = 1, 2, \dots, n$

## Example

$W = (\text{book, compute, is, paper, suppose, square, symbol, write})$

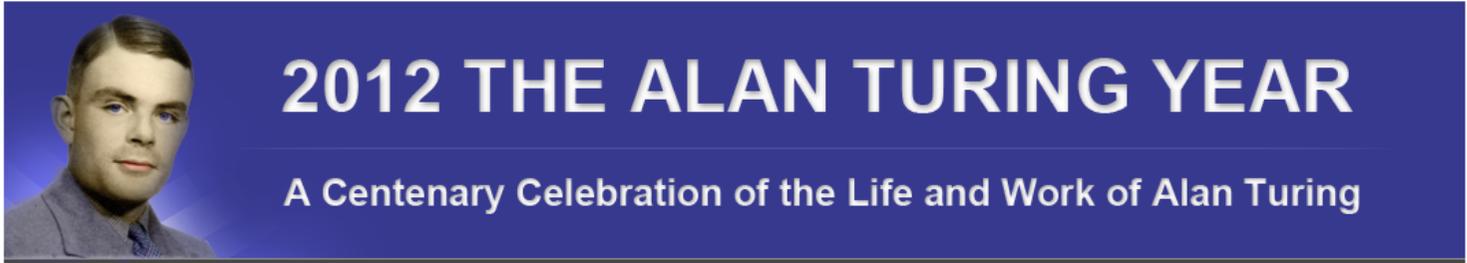
$s_1$ : **Computing** is normally done by **writing** certain **symbols** on **paper**.

$s_2$ : We may **suppose** this **paper** is divided into **squares** like a child's **arithmetic book**.

$$\Phi(s_1) = (0, 0, 1, 1, 1, 0, 0, 1, 1)$$

$$\Phi(s_2) = (1, 1, 1, 0, 1, 1, 1, 0, 0)$$

- Alan Turing: On Computable Numbers, with an Application to the Entscheidungsproblem: A correction”. Proceedings of the London Mathematical Society 43: pp. 544–6. 1937. doi:10.1112/plms/s2-43.6.544



**2012 THE ALAN TURING YEAR**  
A Centenary Celebration of the Life and Work of Alan Turing

**Centenary Events**

- ATY EVENTS OVERVIEW
- ATY EVENTS CALENDAR
- ATY EVENTS A4 HANDOUT
- ATY RESOURCES
- TCAC Arts & Culture Subcttee
- TCAC Media Group
- Turing Manchester 2012
- TCAC Manchester
- Alan Turing Jahr 2012
- TCAC Germany
- Alan Turing Jaar 2012
- AAAI Turing Lecture **New!**
- ACE 2012, Cambridge
- ACM Centenary Celebration
- AI at Donetsk, Ukraine
- AI\*IA Symp. Artificial Intelligence
- Alan Mathison Turing, Roma
- Alan Turing Centenary in Calgary
- ALAN TURING CONF, Manchester
- Alan Turing Days in Lausanne
- AMS Special Session, USA
- AMS-ASL Joint Math Meeting
- Animation12, Manchester
- ASL Turing Conference **New!**

- **To link to this webpage** please use the url: <http://www.turingcentenary.eu/> - and add the **ATY logo** (suitably resized) to your webpage. See also **pdf version** or **monochrome version**
- **If you wish to be included in the Turing Centenary email list**, please enter your email address here and press **Submit**:

The Alan Turing Year on Facebook - and on Twitter

ATY Press and Media Contact - Daniela Derbyshire - email: [turing@live.co.uk](mailto:turing@live.co.uk)

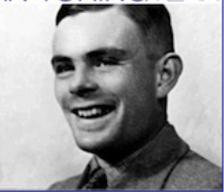
**THE TURING TEST**  
An opera by Julian Wagstaff

**The Turing Test**  
a one-hour opera, sung in English  
UK tour 2012 - help us make it happen!



ALAN TURING YEAR

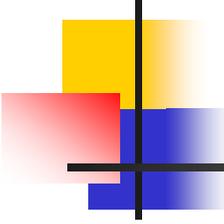
2012



**News**

- 19.04.12**  
GCHQ releases two codebreaking papers by Alan Turing
- 09.04.12**  
The biography of Alan M Turing by his mother Sara appears
- 06.04.12**  
Manchester Pride Festival to honour Alan Turing

**June 23, 2012, is the Centenary of Alan Turing's birth in London.** During his relatively brief life, Turing made a unique impact on the history of computing, computer science, artificial intelligence, developmental biology, and the mathematical theory of computability.



## Example of Transformation (2)

A simple method for embedding sequences into  $\mathbf{R}^n$  is using  $N$ -grams.

- Let  $D$  be the domain of sequences consisting of characters **a** and **b**.
- An  $N$ -gram is a sequence consisting of  $N$  characters.

For example 3-grams are **aaa**, **aab**, **aba**, ..., **bbb**.

- We have  $2^N$  different  $N$ -grams for the domain  $D$ , and enumerate them as  $w_1, w_2, \dots, w_n$ , where  $n = 2^N$ .
- We define a transformation  $\Phi : D \rightarrow \mathbf{R}^n$  as:

$$\Phi(s) = (x_1, x_2, \dots, x_k) \text{ where}$$

$$x_i = \text{how often } w_i \text{ appears in } s \text{ for } i = 1, 2, \dots, n$$

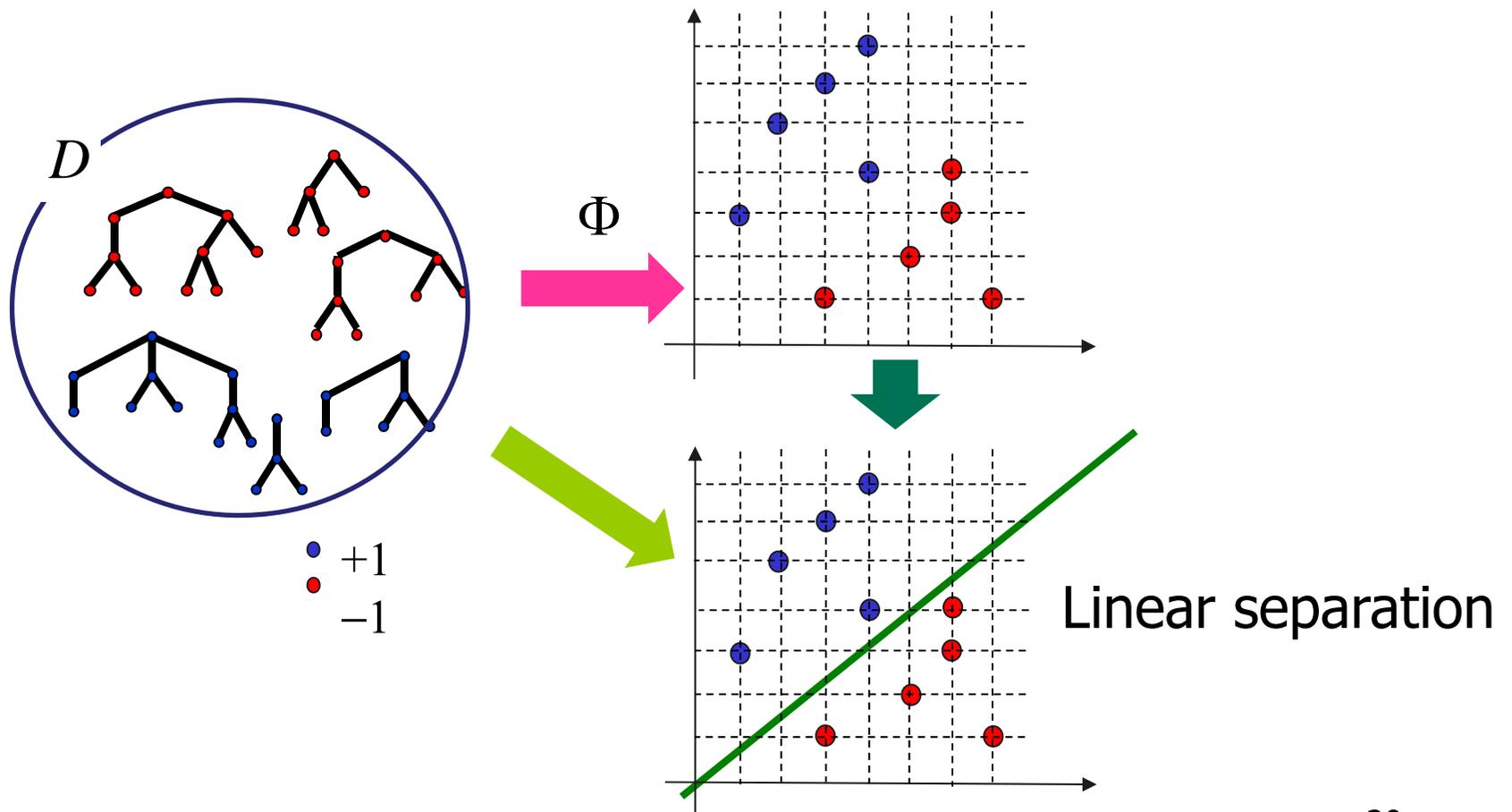
### Example

Let  $w_1 = \mathbf{aaa}$ ,  $w_2 = \mathbf{aab}$ ,  $w_3 = \mathbf{aba}$ ,  $w_4 = \mathbf{abb}$ , ...,  $w_8 = \mathbf{bbb}$   
and  $s = \mathbf{aabbaaabb}$ .

$$\Phi(s_1) = (1, 2, 0, 1, \dots, 1)$$

# Example of Transformation (3)

- We do not need the expression of  $\Phi$ , but need the value  $K(x, y) = \Phi(x) \cdot \Phi(y)$ , called the **kernel function**.



# Support Vector Machine(1)

- **Input:** a set of numerical data

$$\{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_m, c_m)\} \quad \mathbf{x}_i \in \mathbf{R}^n$$

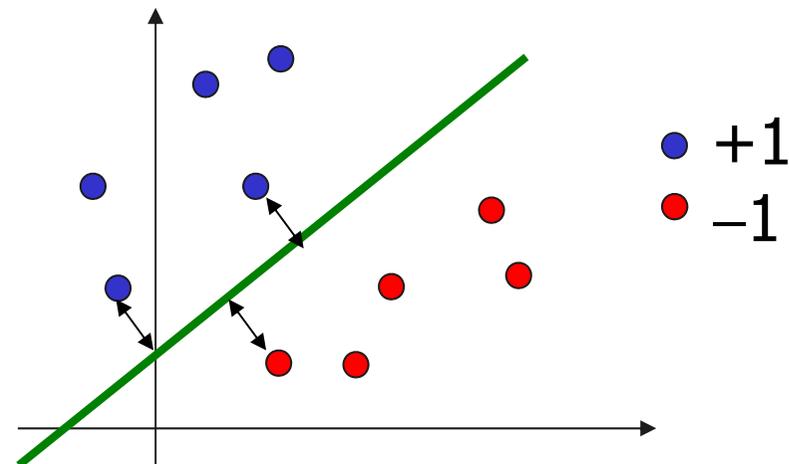
where each  $c_i \in \{+1, -1\}$  is a class signal for  $\mathbf{x}_i$

**Output:** find a linear function (hyper-plane)

$$f(\mathbf{x}) = \sum w_i \mathbf{x}_i \cdot \mathbf{x} + b$$

which  $\text{sign}(f(\mathbf{x}_i)) = y_i$  for all  $i$  and

maximize the margin  $\min_{1 \leq i \leq m} d(f, \mathbf{x}_i)$



# Support Vector Machine(2)

- In order to find  $c$  as well as  $\mathbf{w}$ , we regard every data

$$\mathbf{x}_i \text{ as } \mathbf{x}'_i = (\mathbf{x}_i, 1) \text{ and } f(\mathbf{x}'_i) = \mathbf{w}' \cdot \mathbf{x}'_i = 0.$$

Moreover, we can represent two conditions

$$c_i = +1 \Rightarrow \mathbf{w}' \cdot \mathbf{x}'_i \geq 0 \text{ and } c_i = -1 \Rightarrow \mathbf{w}' \cdot \mathbf{x}'_i \leq 0$$

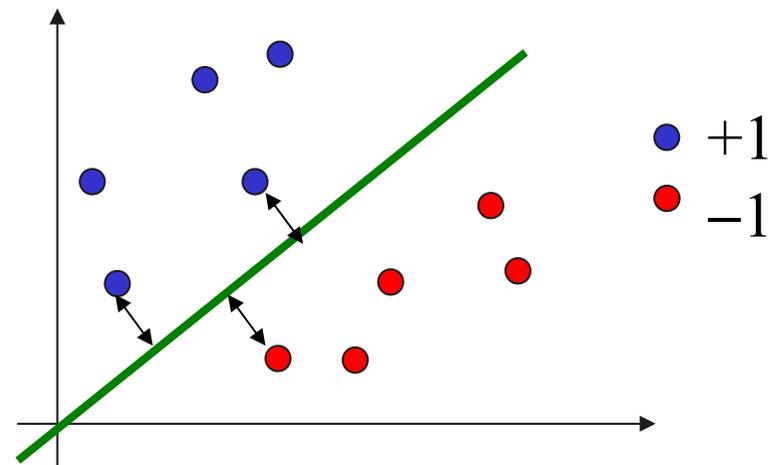
into one

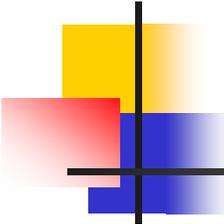
$$c_i (\mathbf{w}' \cdot \mathbf{x}'_i) \geq 0$$

- In this setting

$$d(f, \mathbf{x}'_i) = \|\mathbf{x}'_i\| \cos \theta_i = \frac{1}{\|\mathbf{w}'\|}$$

- In the followings, we write  $\mathbf{x}_i$  for  $\mathbf{x}'_i$  and  $\mathbf{w}_i$  for  $\mathbf{w}'_i$ .





# Kernel function for Boolean Data

---

- If vectors  $\mathbf{x}$  and  $\mathbf{y}$  are boolean, the dot product  $\mathbf{x} \cdot \mathbf{y}$  represents: **how many coordinates of  $\mathbf{x}$  and  $\mathbf{y}$  coincide.**

## Example

$$\mathbf{x} = (0, 0, 1, 1, 1, 0, 0, 1, 1)$$

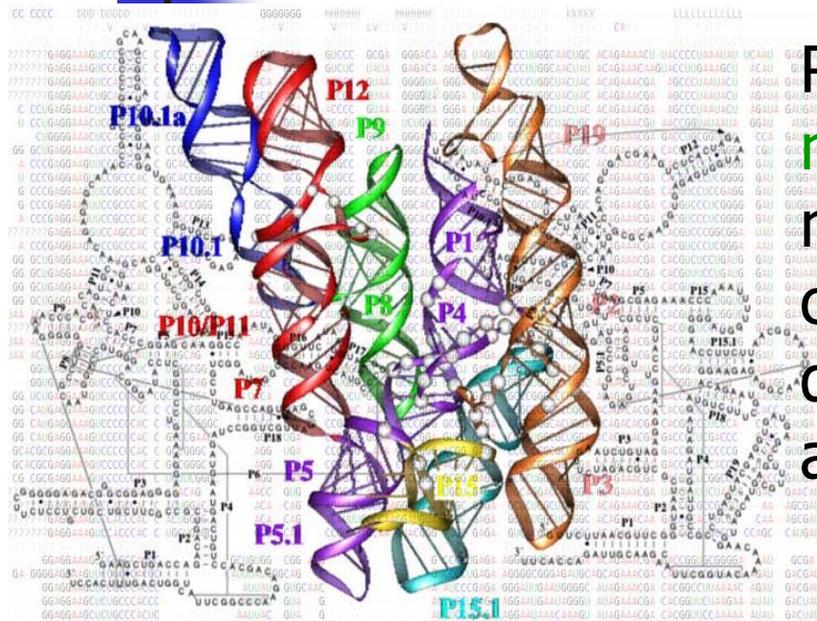
$$\mathbf{y} = (1, 1, 1, 0, 1, 1, 1, 0, 0)$$

$$\mathbf{x} \cdot \mathbf{y} = 2$$

- For two boolean values  $x$  and  $y$ , the logical conjunction  $x \cdot y$  coincides with the product  $x y$  as real numbers.
- This dot product is too simple and the DNF kernel is developed [Sadohara01, Kahdon05]

$$K(\mathbf{x}, \mathbf{y}) = 2^{(\mathbf{x} \cdot \mathbf{y})} - 1$$

# RNA Classification 1



Recently, in **bioinformatics**, classifying **non-coding RNAs** (ncRNAs) is paid to much attention, because they are considered to be a factor of the difference between **higher organism** and **others**.

**RNA sequences** are accumulated in **RNA families**, and the members of each family have similar structures and functions.

We can get the RNA sequences in **Rfam database**.

Rfam: Rfam Home Page - Microsoft Internet Explorer

RNA families database of alignments and CMs

Home Keyword Search Sequence Search Browse Rfam Genomes ftp Help miRNA

Rfam Home Page

Rfam is a joint project involving researchers based at the [Wellcome Trust Sanger Institute](#), Cambridge, UK and [Janella Farm](#), Ashburn, VA, USA. Rfam is a large collection of multiple sequence alignments and covariance models covering many common non-coding RNA families. For each family in Rfam you can:

- View and download multiple sequence alignments
- Read family annotation
- Examine species distribution of family members
- Follow links to otherdatabases

In conjunction with the [INFERNAL](#) software suite, Rfam can be used to annotate sequences (including complete genomes) for homologues to known non-coding RNAs. Please read important information about [using Rfam for genome annotation](#). We provide pre-calculated lists of putative RNAs in over [200 complete genomes](#), and a [web search facility](#) for short sequences.

Rfam makes use of a large amount of available data, especially published multiple sequence alignments, and repackages these data in a single searchable and sustainable resource. We have made every effort to credit individual sources on family pages, and have compiled a list of links to these sources [here](#). If you find any of the data presented here useful, please also be sure to credit the primary source.

For more information on Rfam, and using this site, click [here](#).

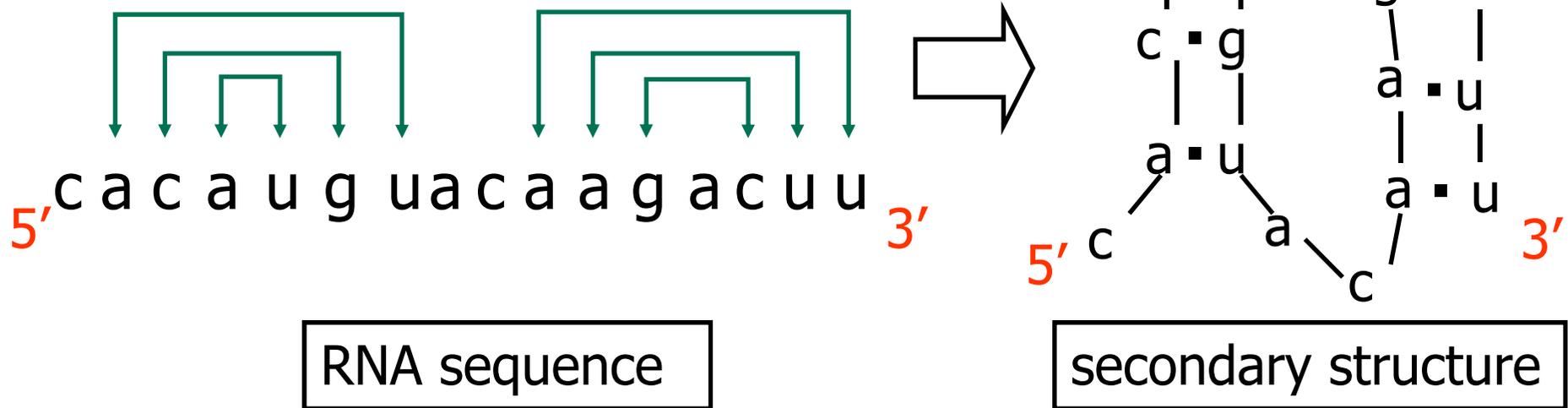
Rfam Mirror Servers Worldwide FTP access to Rfam

[Sanger Institute \(UK\)](#)  
[Janella Farm \(USA\)](#)

You can also download the Rfam database and for instance search it locally using the [INFERNAL](#) covariance model software. [Hypertlink directly to the ftp site](#) or [View ftp site files](#)

# RNA Classification 2

In RNA classification, the **secondary structures** detected by base pairs (**a-u, c-g**) are important.

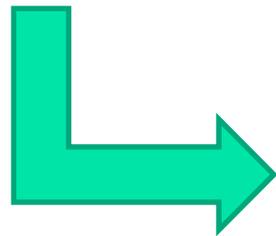


The purpose

To distinguish between the **member sequences** in a given **RNA family** and **non-member** sequences by taking secondary structures into account.

# Structure as coordinate

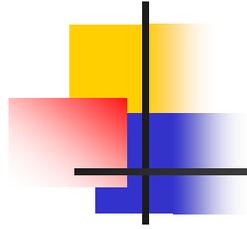
- We use **structures** as coordinates (attributes) for the transformation of RNA sequences.



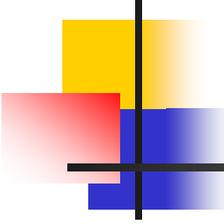
The Naïve Algorithm takes  $O(L^4)$  time.

An improved algorithm runs in  $O(L^3)$ .

RNA			
CAGCAUCGAUGA	18	43	36
CAGCAUGGAGAC	17	33	34
AUUAGUUGUCGA	16	34	39



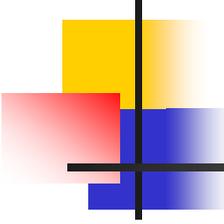
# Learning from Discrete Data by Discrete Mathematics



# Problems on the first approach

---

- The obtained data in  $\mathbf{R}^n$  might not locate densely.
  - They sometimes in  $N^n$  .
- Even if a rule is obtained by some learning machine, it might be difficult to interpret the rule, or what the rule mean.

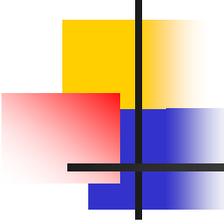


# On the second approach

---

We have to know mathematics on discrete data.

- The mathematics may vary from type to type of data.
  - mathematics on sequences, mathematics on trees, mathematics on graphs,...
- We must notice that we need mathematics for machine learning.
  - We make machine learning more abstract, and then observe the correspondence between numerical data and discrete data.



# More General Learning

---

- Recently a machine learning method is recognized as one to find

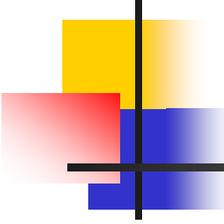
$$\operatorname{argmin}_{f \in H} (\sum_{\mathbf{x} \in D} \operatorname{Loss}(f, \mathbf{x}) + \lambda P(f))$$

where

$\operatorname{Loss}(f, \mathbf{x})$  is a loss function and

$P(f)$  is a penalty function.

- This definition is declarative.
- This course we introduce some of the instances of  $\operatorname{Loss}(f, \mathbf{x})$  and  $P(f)$ .



# Abstract Classification

---

- A half-plane  $P$  which contains  $C$  (yes) and excludes  $D$  (no) is to be learned
- The half-plane  $P$  is represented as a pair  $(\mathbf{w}, c)$  which means the linear inequation  $(\mathbf{w}, \mathbf{x}) + c > 0$ .

- Let  $C(p) = \{\mathbf{x} \in \mathbf{R}^n \mid p(\mathbf{x})\}$  for a predicate  $p$ .

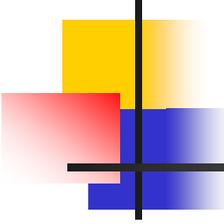
Then the search space (version space) is

$$\mathbf{C} = \{C(\lambda \mathbf{x} \cdot ((\mathbf{w}, \mathbf{x}) + c > 0)) \mid \mathbf{w} \in \mathbf{R}^n, c \in \mathbf{R}^n\}.$$

The set of parameter  $s$  are from

$$\mathbf{H} = \{(\mathbf{w}, c) \mid \mathbf{w} \in \mathbf{R}^n, c \in \mathbf{R}^n\}.$$

- The training examples are provided as the sets  $C$  and  $D$ .
- A learning algorithm is provided.



# Learning from string data

---

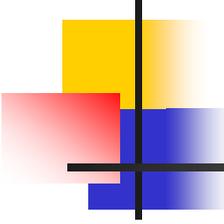
- Assume that we are treating data on the domain of sequences of characters.
- Then we treat the problem of classifying two finite sets of sequences  $C$  (yes),  $D$  (no) ( $C \cap D = \emptyset$ ).

## Example

Let  $D$  be the domain of sequences consisting of characters  $a$  and  $b$ .

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

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



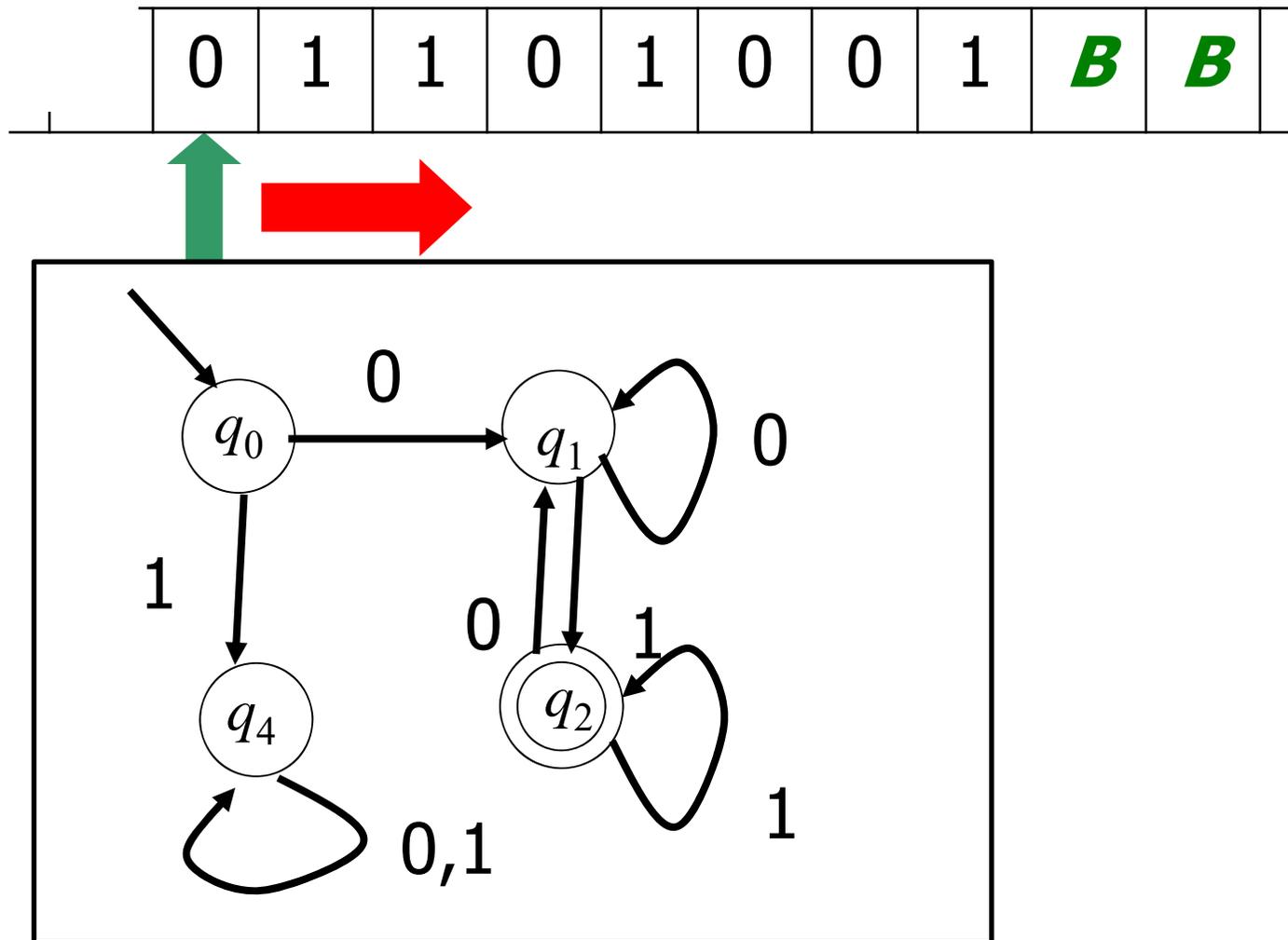
# How to distinguish data

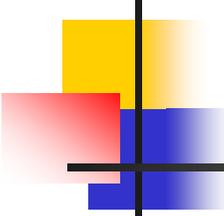
---

In this course we treat the following methods:

- Abstract machines to distinguish data
  - Finite state automata, Turing Machine, ...
- Formal grammar with production rules
  - Linear grammar, Context free grammar, ...
- Regarding string data as mathematical objects
  - Based on the operation :  $aaba$  means  $a \cdot (a \cdot (b \cdot a))$ 
    - monomials (patterns), instead of linear combinations

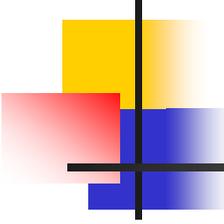
# Finite state automaton





# Grammar with productions

- $S = \{ a^n b^n \mid n \geq 1 \} = \{ \underbrace{a \dots a}_{n \text{ times}} \underbrace{b \dots b}_{n \text{ times}} \mid n \geq 1 \}$   
 $= \{ ab, aabb, aaabbb, aaaabbbb, \dots \}$
- The language is defined with a set of productions:  
 $S \rightarrow ab, S \rightarrow aSb$
- Some examples of derivations:  
 $S \Rightarrow ab$   
 $S \Rightarrow aSb \Rightarrow aabb$   
 $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaabbb$   
 $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaabbbb$
- It is easy to show that there is no FA which accepts  $L$ .



# Patterns (Monomials)

---

- A **pattern**  $\pi$  is a sequence consisting symbols and variables
  - Assuming that we can distinguish characters and variables.

## Example

Characters: **a**, **b**, Variables :  $x$ ,  $y$ ,...

Patterns: **a** $x$ **b**, **b** $x$ **a** $y$ **b**, **a** $x$ **b** $y$ **b $x$ **a**,...**

The set defined with a pattern

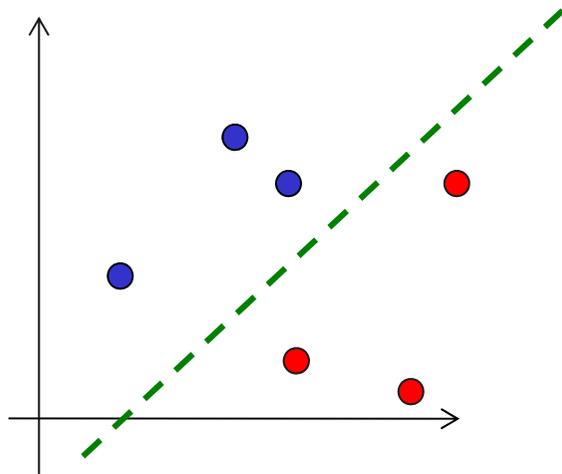
$$L(\mathbf{a}x\mathbf{b}) = \{\mathbf{a}ab, ab\mathbf{b}, \mathbf{a}ab, \mathbf{a}bb, a\mathbf{b}ab, ab\mathbf{b}b, \dots\}$$

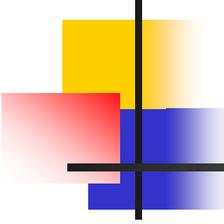
$$L(\mathbf{b}x\mathbf{a}y\mathbf{b}) = \{\mathbf{b}aa\mathbf{b}, \mathbf{b}aabb, \mathbf{b}aaa\mathbf{b}, \mathbf{b}aaabb, \\ \mathbf{b}aabab, \dots\}$$

# Making the learning be abstract

- In the case of treating sequences, **what is the correspondence to the linear inequation?**

$$\begin{aligned} \mathbf{x} \in C &\Rightarrow (\mathbf{w}, \mathbf{x}) + c \geq 0 \\ \mathbf{x} \in D &\Rightarrow (\mathbf{w}, \mathbf{x}) + c \leq 0 \end{aligned} \quad \leftarrow \text{parameter } (\mathbf{w}, c)$$





# A Learning Algorithm

---

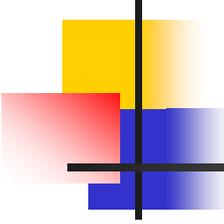
1. Let the input data  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
2. Initialize  $\mathbf{w}$  as some value.
3. For  $n = 1, 2, \dots, N$ ,
  - if  $\mathbf{x}_n \in C$  and  $(\mathbf{w}, \mathbf{x}_n) < 0$ 
    - replace  $\mathbf{w}$  with  $\mathbf{w} + \rho \mathbf{x}_n$
  - else if  $\mathbf{x}_n \in D$  and  $(\mathbf{w}, \mathbf{x}_n) > 0$ 
    - replace  $\mathbf{w}$  with  $\mathbf{w} - \rho \mathbf{x}_n$
  - otherwise
    - do nothing
4. For  $n = 1, 2, \dots, N$ 
  - if no  $\mathbf{x}_n$  satisfies
    - $(\mathbf{x}_n \in C \text{ and } (\mathbf{w}, \mathbf{x}_n) < 0) \text{ or } (\mathbf{x}_n \in D \text{ and } (\mathbf{w}, \mathbf{x}_n) > 0)$
    - terminates and return  $\mathbf{w}$
  - else
    - go to 3.

# Machines with parameters

- We regard the inequation  $(\mathbf{w}, \mathbf{x}) + c \geq 0$  as a **machine** to distinguish whether or not every datum  $\mathbf{x}$  is in  $C$ .



- For the case of treating strings, we should adopt machines which can distinguish whether or not every string  $\mathbf{x}$  is in  $C$ .



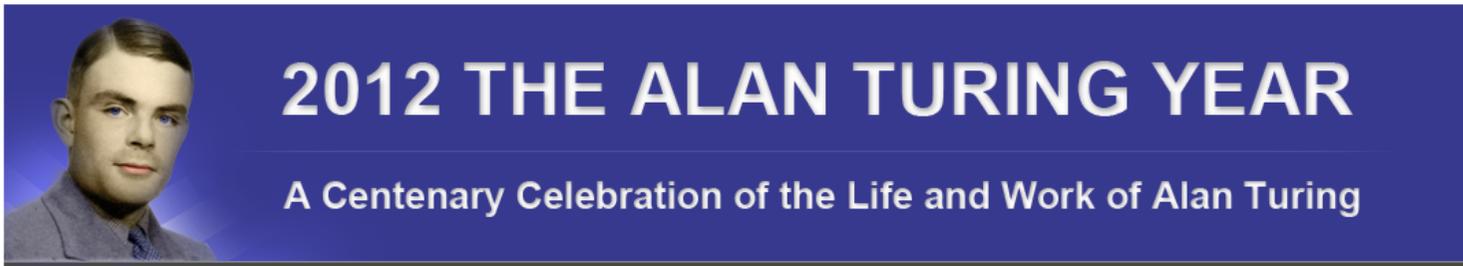
# Learning Finite State Automata

---

1. Let the input data  $x_1, x_2, \dots, x_N$
2. Initialize  $L$  as some automaton.
3. For  $n = 1, 2, \dots, N$ ,
  - if  $x_n \in C$  and  $M$  does not accept  $x_n$   
replace  $M$  with another  $M'$
  - else if  $x_n \in D$  and  $M$  accepts  $x_n$   
replace  $M$  with another  $M'$
  - otherwise  
do nothing
4. For  $n = 1, 2, \dots, N$ 
  - if no  $x_n$  satisfies  
( $x_n \in C$  and  $M$  does not accept  $x_n$ ) or ( $x_n \in D$  and  $M$  accepts  $x_n$ )  
terminates and return  $M$
  - else  
go to 3.

# Turing Machine

- Alan Turing: On Computable Numbers, with an Application to the Entscheidungsproblem: A correction”. Proceedings of the London Mathematical Society 43: pp. 544–6. 1937. doi:10.1112/plms/s2-43.6.544



**2012 THE ALAN TURING YEAR**  
A Centenary Celebration of the Life and Work of Alan Turing

**Centenary Events**

- ATY EVENTS OVERVIEW
- ATY EVENTS CALENDAR
- ATY EVENTS A4 HANDOUT
- ATY RESOURCES
- TCAC Arts & Culture Subcttee
- TCAC Media Group
- Turing Manchester 2012
- TCAC Manchester
- Alan Turing Jahr 2012
- TCAC Germany
- Alan Turing Jaar 2012
- AAAI Turing Lecture **New!**
- ACE 2012, Cambridge
- ACM Centenary Celebration
- AI at Donetsk, Ukraine
- AI\*IA Symp. Artificial Intelligence
- Alan Mathison Turing, Roma
- Alan Turing Centenary in Calgary
- ALAN TURING CONF, Manchester
- Alan Turing Days in Lausanne
- AMS Special Session, USA
- AMS-ASL Joint Math Meeting
- Animation12, Manchester
- ASL Turing Conference **New!**

- **To link to this webpage** please use the url: <http://www.turingcentenary.eu/> - and add the **ATY logo** (suitably resized) to your webpage. See also **pdf version** or **monochrome version**
- **If you wish to be included in the Turing Centenary email list**, please enter your email address here and press **Submit**:

The Alan Turing Year on Facebook - and on Twitter

ATY Press and Media Contact - Daniela Derbyshire - email: [turing@live.co.uk](mailto:turing@live.co.uk)

**THE TURING TEST**  
An opera by Julian Wagstaff  
**The Turing Test**  
a one-hour opera, sung in English  
UK tour 2012 - help us make it happen!

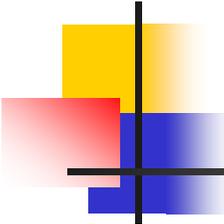
**June 23, 2012, is the Centenary of Alan Turing's birth in London.** During his relatively brief life, Turing made a unique impact on the history of computing, computer science, artificial intelligence, developmental biology, and the mathematical theory of computability.

**ALAN TURING YEAR**  
2012

**News**

- 19.04.12**  
GCHQ releases two codebreaking papers by Alan Turing
- 09.04.12**  
The biography of Alan M Turing by his mother Sara appears
- 06.04.12**  
Manchester Pride Festival to honour Alan Turing

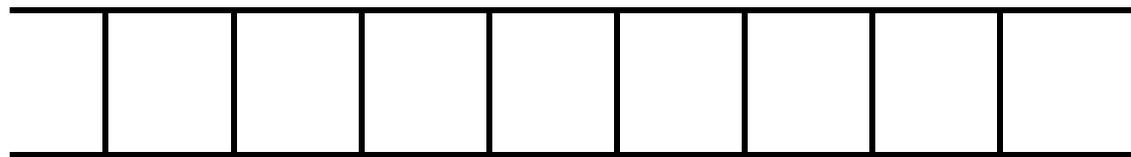




# Turing Machine(2)

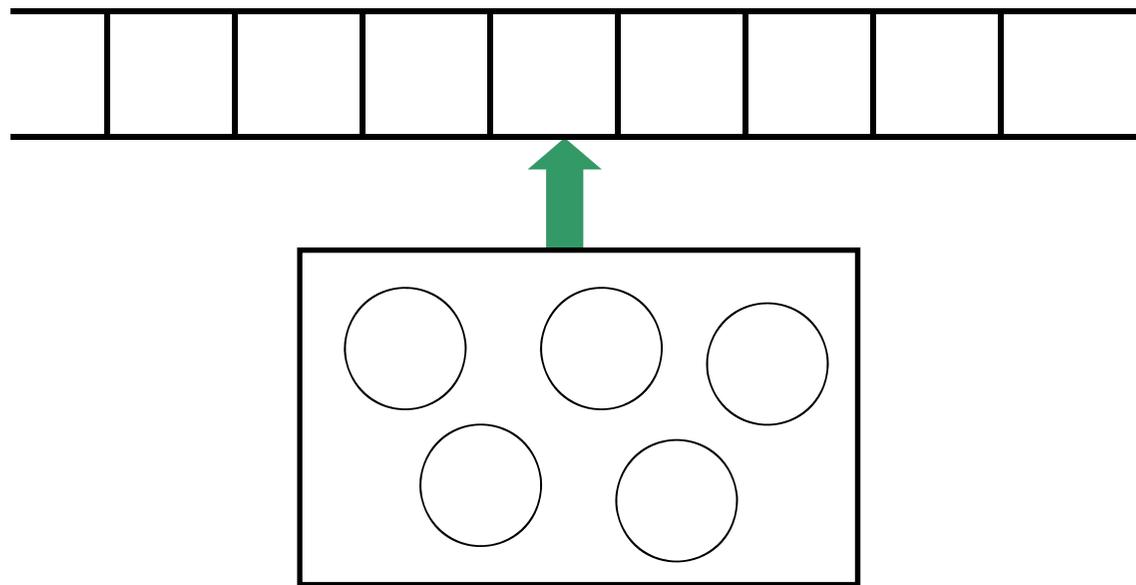
---

- 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.
- I assume then that the computation is carried out on one-dimensional paper, i.e. on a tape divided into squares.

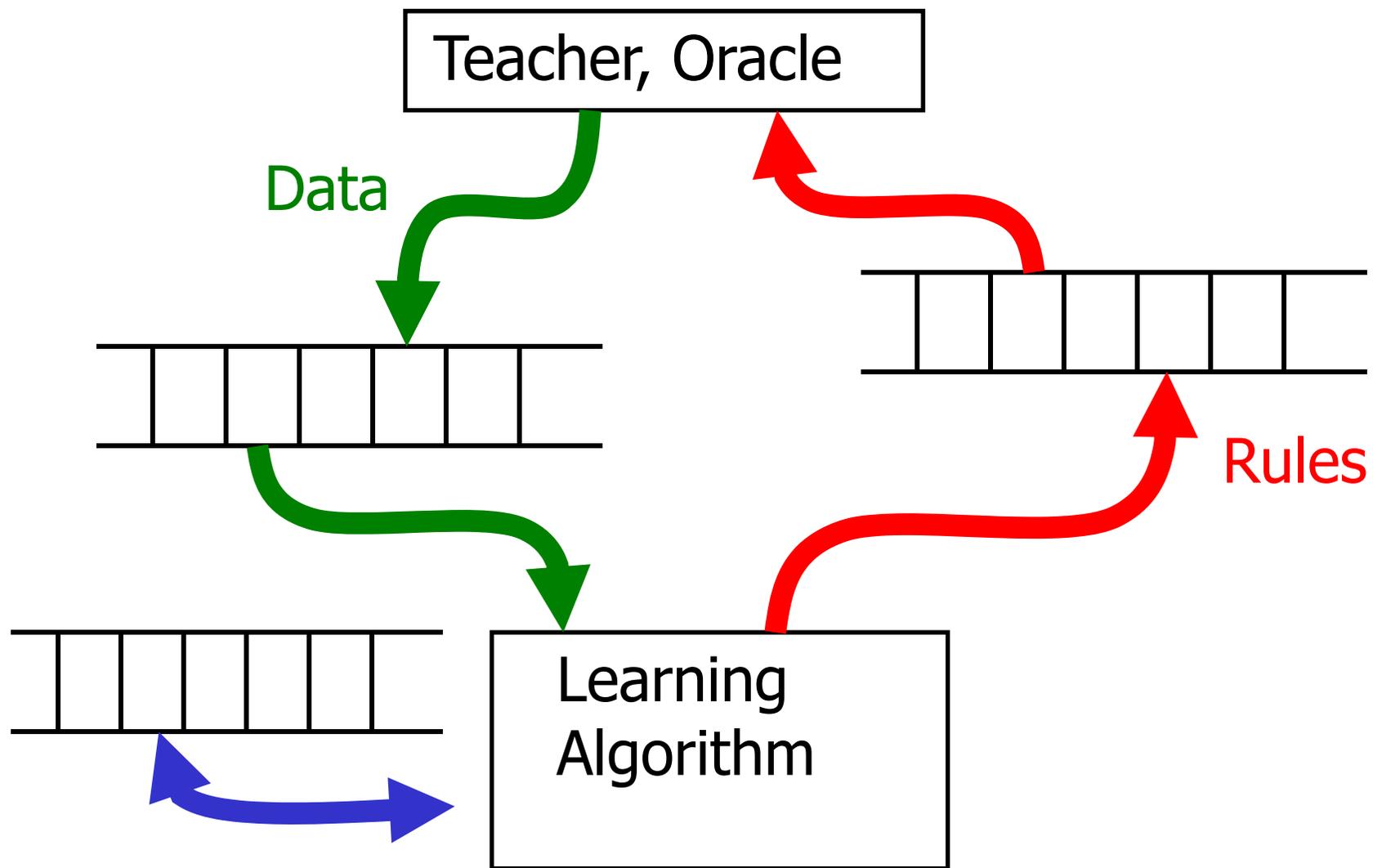


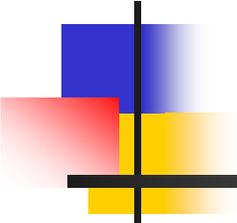
# Turing Machine(3)

- 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 will also suppose that the number of states of mind which need be taken into account is finite.



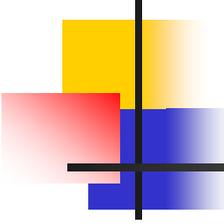
# Learning Machine





# Conclusion

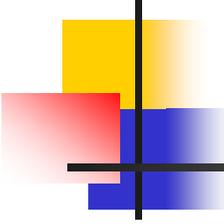
---



# Elements of Learning Theories

---

- A class of rules to be learned
- A uniform representation method of each rule
  - We assume that each rule is represented by an expression/a formula defined by a grammar.
- A representation method of training examples/observation
- A learning algorithm
- A method evaluation / some criteria of justification of the learning algorithm



# References

---

- Colin de la Higuera, *Grammatical Inference*, Cambridge University press, 2010..
- 横森・榊原・小林：“計算論的学習” 培風館