

# Introduction to Computational Learning in the Limit

Timo Kötzing  
MPI for Computer Science

September 22, 2009

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

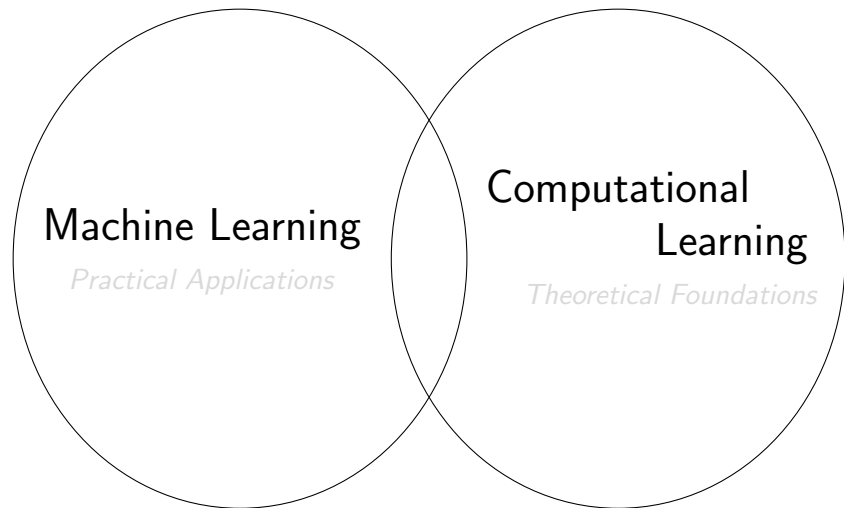
- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

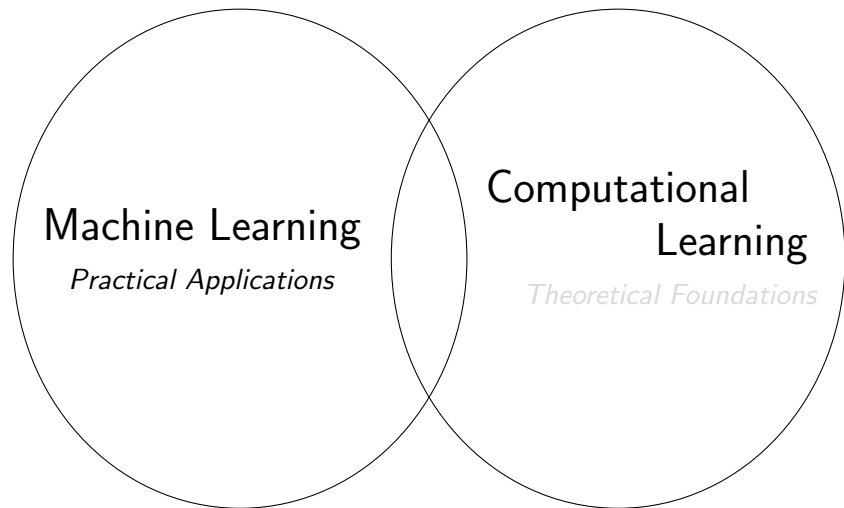
# What is Learning?

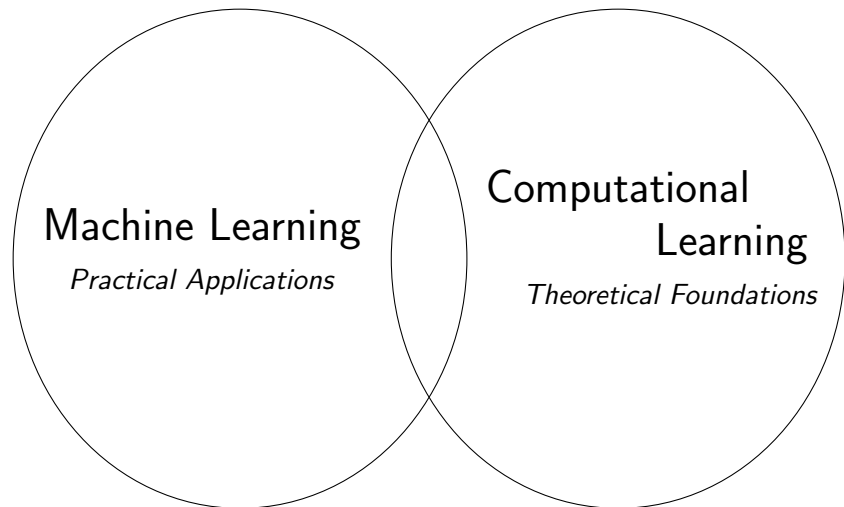
- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.

# What is Learning?

- ▶ What do we want to learn?
  - ▶ Finite objects (classification from finitely many attributes/values, ...);
  - ▶ Possibly infinite objects (regular languages, ...);
  - ▶ Stochastic objects (distributions over strings, ...).
- ▶ How do we get information about the target?
  - ▶ Labeled data;
  - ▶ Only positive data;
  - ▶ For stochastic objects: samples.
- ▶ Data deficiencies:
  - ▶ Incomplete data;
  - ▶ Small amount data;
  - ▶ Noisy data.
- ▶ Possible Desires:
  - ▶ Giving a model for the data;
  - ▶ Making predictions about future data.







# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# Function Learning

For this talk:

- ▶ We want to learn **infinite sequences** (total functions).
- ▶ We get **more and more** (complete and correct) **argument/value pairs**.
- ▶ We try to give models in the form of **programs**.

$$0, 1, 2, 3, 4, 5, \dots \quad x \mapsto x$$

$$0, 1, 4, 9, 16, 25, \dots \quad x \mapsto x^2$$

$$0, 1, 0, 0, 0, 0, \dots \quad x \mapsto \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise.} \end{cases}$$

$$0, 1, 1, 2, 3, 5, 8, 13, \dots \quad x \mapsto \text{fib}(x)$$

# The Problem with Extrapolation

- ▶ Consider the sequence

3, 5, 7.

- ▶ What would be the next elements after this sequence?
- ▶ Maybe it's

9, 11, 13, ...

(we get **odd numbers in order**).

- ▶ Or it could be

11, 13, 17, ...

(we get **odd primes in order**).<sup>1</sup>

- ▶ How can we decide between the two scenarios?
- ▶ Impossible with the information given!
- ▶ The particular choice of any learning algorithm is the **bias** of the algorithm.

---

<sup>1</sup>Thanks to [Chinmoy Dutta](#) for inspiration to this example. 

# The Problem with Extrapolation

- ▶ Consider the sequence

3, 5, 7.

- ▶ What would be the next elements after this sequence?
- ▶ Maybe it's

9, 11, 13, ...

(we get **odd numbers in order**).

- ▶ Or it could be

11, 13, 17, ...

(we get **odd primes in order**).<sup>1</sup>

- ▶ How can we decide between the two scenarios?
- ▶ Impossible with the information given!
- ▶ The particular choice of any learning algorithm is the **bias** of the algorithm.

---

<sup>1</sup>Thanks to [Chinmoy Dutta](#) for inspiration to this example. 

# The Problem with Extrapolation

- ▶ Consider the sequence

3, 5, 7.

- ▶ What would be the next elements after this sequence?

- ▶ Maybe it's

9, 11, 13, ...

(we get **odd numbers in order**).

- ▶ Or it could be

11, 13, 17, ...

(we get **odd primes in order**).<sup>1</sup>

- ▶ How can we decide between the two scenarios?
- ▶ Impossible with the information given!
- ▶ The particular choice of any learning algorithm is the **bias** of the algorithm.

---

<sup>1</sup>Thanks to [Chinmoy Dutta](#) for inspiration to this example. 

# The Problem with Extrapolation

- ▶ Consider the sequence

3, 5, 7.

- ▶ What would be the next elements after this sequence?
- ▶ Maybe it's

9, 11, 13, ...

(we get **odd numbers in order**).

- ▶ Or it could be

11, 13, 17, ...

(we get **odd primes in order**).<sup>1</sup>

- ▶ How can we decide between the two scenarios?
- ▶ Impossible with the information given!
- ▶ The particular choice of any learning algorithm is the **bias** of the algorithm.

---

<sup>1</sup>Thanks to [Chinmoy Dutta](#) for inspiration to this example. 

# The Problem with Extrapolation

- ▶ Consider the sequence

3, 5, 7.

- ▶ What would be the next elements after this sequence?
- ▶ Maybe it's

9, 11, 13, ...

(we get **odd numbers in order**).

- ▶ Or it could be

11, 13, 17, ...

(we get **odd primes in order**).<sup>1</sup>

- ▶ How can we decide between the two scenarios?
- ▶ Impossible with the information given!
- ▶ The particular choice of any learning algorithm is the **bias** of the algorithm.

---

<sup>1</sup>Thanks to [Chinmoy Dutta](#) for inspiration to this example. 

# The Problem with Extrapolation

- ▶ Consider the sequence

3, 5, 7.

- ▶ What would be the next elements after this sequence?
- ▶ Maybe it's

9, 11, 13, ...

(we get **odd numbers in order**).

- ▶ Or it could be

11, 13, 17, ...

(we get **odd primes in order**).<sup>1</sup>

- ▶ How can we decide between the two scenarios?
  - ▶ Impossible with the information given!
  - ▶ The particular choice of any learning algorithm is the **bias** of the algorithm.

---

<sup>1</sup>Thanks to [Chinmoy Dutta](#) for inspiration to this example. 

# The Problem with Extrapolation

- ▶ Consider the sequence

3, 5, 7.

- ▶ What would be the next elements after this sequence?
- ▶ Maybe it's

9, 11, 13, ...

(we get **odd numbers in order**).

- ▶ Or it could be

11, 13, 17, ...

(we get **odd primes in order**).<sup>1</sup>

- ▶ How can we decide between the two scenarios?
- ▶ Impossible with the information given!
- ▶ The particular choice of any learning algorithm is the **bias** of the algorithm.

---

<sup>1</sup>Thanks to [Chinmoy Dutta](#) for inspiration to this example. 

# The Problem with Extrapolation

- ▶ Consider the sequence

3, 5, 7.

- ▶ What would be the next elements after this sequence?
- ▶ Maybe it's

9, 11, 13, ...

(we get **odd numbers in order**).

- ▶ Or it could be

11, 13, 17, ...

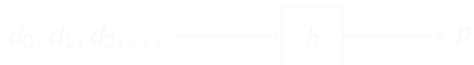
(we get **odd primes in order**).<sup>1</sup>

- ▶ How can we decide between the two scenarios?
- ▶ Impossible with the information given!
- ▶ The particular choice of any learning algorithm is the **bias** of the algorithm.

---

<sup>1</sup>Thanks to [Chinmoy Dutta](#) for inspiration to this example. 

► Finite Learning:



► Learning in the Limit:



► Finite Learning:



► Learning in the Limit:



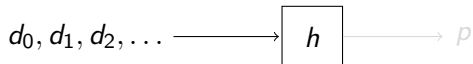
► Finite Learning:



► Learning in the Limit:



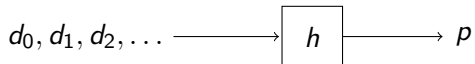
► Finite Learning:



► Learning in the Limit:



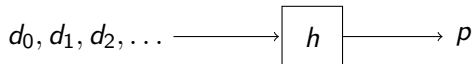
► Finite Learning:



► Learning in the Limit:



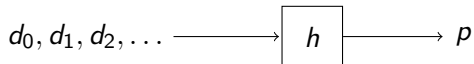
► Finite Learning:



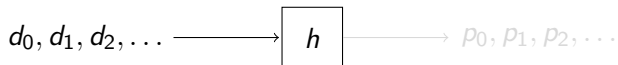
► Learning in the Limit:



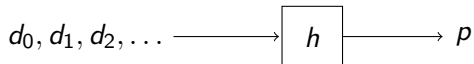
► Finite Learning:



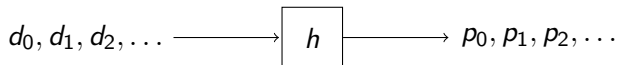
► Learning in the Limit:



► Finite Learning:



► Learning in the Limit:



# Formal Definitions

- ▶ Let  $\mathbb{N} = \{0, 1, 2, \dots\}$ .
- ▶ For  $g : \mathbb{N} \rightarrow \mathbb{N}$  and  $k \in \mathbb{N}$ :  $g[k]$  denotes the sequence  $g(0), \dots, g(k-1)$ .
- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **G-learning sequence**<sup>2</sup>  $p$  of  $h$  on  $g$  by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GEx-learns**<sup>3</sup>  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GEx-learnable** iff there is a learner  $h$  **GEx-learning** every  $g \in \mathcal{S}$ .
- ▶ For example: **PF** is **GEx-learnable**.

---

<sup>2</sup>G for “Gold”, see Gold '67.

<sup>3</sup>Ex for “explanatory”.

# Formal Definitions

- ▶ Let  $\mathbb{N} = \{0, 1, 2, \dots\}$ .
- ▶ For  $g : \mathbb{N} \rightarrow \mathbb{N}$  and  $k \in \mathbb{N}$ :  $g[k]$  denotes the sequence  $g(0), \dots, g(k-1)$ .
- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **G-learning sequence**<sup>2</sup>  $p$  of  $h$  on  $g$  by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GEx-learns**<sup>3</sup>  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GEx-learnable** iff there is a learner  $h$  **GEx-learning** every  $g \in \mathcal{S}$ .
- ▶ For example: **PF** is **GEx-learnable**.

---

<sup>2</sup>G for “Gold”, see Gold '67.

<sup>3</sup>Ex for “explanatory”.

# Formal Definitions

- ▶ Let  $\mathbb{N} = \{0, 1, 2, \dots\}$ .
- ▶ For  $g : \mathbb{N} \rightarrow \mathbb{N}$  and  $k \in \mathbb{N}$ :  $g[k]$  denotes the sequence  $g(0), \dots, g(k-1)$ .
- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **G-learning sequence**<sup>2</sup>  $p$  of  $h$  on  $g$  by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GEx-learns**<sup>3</sup>  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GEx-learnable** iff there is a learner  $h$  **GEx-learning** every  $g \in \mathcal{S}$ .
- ▶ For example: **PF** is **GEx-learnable**.

---

<sup>2</sup>**G** for “Gold”, see Gold '67.

<sup>3</sup>**Ex** for “explanatory”.

# Formal Definitions

- ▶ Let  $\mathbb{N} = \{0, 1, 2, \dots\}$ .
- ▶ For  $g : \mathbb{N} \rightarrow \mathbb{N}$  and  $k \in \mathbb{N}$ :  $g[k]$  denotes the sequence  $g(0), \dots, g(k-1)$ .
- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **G-learning sequence**<sup>2</sup>  $p$  of  $h$  on  $g$  by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GEx-learns**<sup>3</sup>  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GEx-learnable** iff there is a learner  $h$  **GEx-learning** every  $g \in \mathcal{S}$ .
- ▶ For example: **PF** is **GEx-learnable**.

---

<sup>2</sup>**G** for “Gold”, see Gold '67.

<sup>3</sup>**Ex** for “explanatory”.

# Formal Definitions

- ▶ Let  $\mathbb{N} = \{0, 1, 2, \dots\}$ .
- ▶ For  $g : \mathbb{N} \rightarrow \mathbb{N}$  and  $k \in \mathbb{N}$ :  $g[k]$  denotes the sequence  $g(0), \dots, g(k-1)$ .
- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **G-learning sequence**<sup>2</sup>  $p$  of  $h$  on  $g$  by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GEx-learns**<sup>3</sup>  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $S \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GEx-learnable** iff there is a learner  $h$  **GEx-learning** every  $g \in S$ .
- ▶ For example: **PF** is **GEx-learnable**.

---

<sup>2</sup>**G** for “Gold”, see Gold '67.

<sup>3</sup>**Ex** for “explanatory”.

- ▶ Let  $\mathbb{N} = \{0, 1, 2, \dots\}$ .
- ▶ For  $g : \mathbb{N} \rightarrow \mathbb{N}$  and  $k \in \mathbb{N}$ :  $g[k]$  denotes the sequence  $g(0), \dots, g(k-1)$ .
- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **G-learning sequence**<sup>2</sup>  $p$  of  $h$  on  $g$  by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GEx-learns**<sup>3</sup>  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $S \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GEx-learnable** iff there is a learner  $h$  **GEx-learning** every  $g \in S$ .
- ▶ For example: **PF** is **GEx-learnable**.

---

<sup>2</sup>**G** for “Gold”, see Gold '67.

<sup>3</sup>**Ex** for “explanatory”.

# Postdictive Completeness

- ▶ A learner  $h$  is called **postdictively complete** iff  $\forall g, k, h(g[k])$  correctly postdicts  $g[k]$  ( $= g(0), \dots, g(k-1)$ ), i.e., the programs  $h(g[k])$  correctly compute  $g$  on all inputs  $< k$ .
- ▶ The class of all sets of functions **GEx**-learnable postdictively completely is denoted by **GPcpEx**.
- ▶ **Fact (Barzdin '74, Blum and Blum '75)**: There are Ex-learnable sets of functions, which are not learnable by a postdictively complete learner (a.k.a. the **Inconsistency Phenomenon**).

$$\text{GPcpEx} \subsetneq \text{GEx}.$$

- ▶ **Why?** Because of the undecidability of the halting problem, so that the learner cannot check its desired output for postdictive completeness.

# Postdictive Completeness

- ▶ A learner  $h$  is called **postdictively complete** iff  $\forall g, k, h(g[k])$  correctly postdicts  $g[k]$  ( $= g(0), \dots, g(k-1)$ ), i.e., the programs  $h(g[k])$  correctly compute  $g$  on all inputs  $< k$ .
- ▶ The class of all sets of functions **GEx**-learnable postdictively completely is denoted by **GPcpEx**.
- ▶ **Fact (Barzdin '74, Blum and Blum '75)**: There are Ex-learnable sets of functions, which are not learnable by a postdictively complete learner (a.k.a. the **Inconsistency Phenomenon**).

$$\text{GPcpEx} \subsetneq \text{GEx}.$$

- ▶ **Why?** Because of the undecidability of the halting problem, so that the learner cannot check its desired output for postdictive completeness.

# Postdictive Completeness

- ▶ A learner  $h$  is called **postdictively complete** iff  $\forall g, k, h(g[k])$  correctly postdicts  $g[k]$  ( $= g(0), \dots, g(k-1)$ ), i.e., the programs  $h(g[k])$  correctly compute  $g$  on all inputs  $< k$ .
- ▶ The class of all sets of functions **GEx**-learnable postdictively completely is denoted by **GPcpEx**.
- ▶ **Fact (Barzdin '74, Blum and Blum '75):** There are Ex-learnable sets of functions, which are not learnable by a postdictively complete learner (a.k.a. the **Inconsistency Phenomenon**).

$$\text{GPcpEx} \subsetneq \text{GEx}.$$

- ▶ **Why?** Because of the undecidability of the halting problem, so that the learner cannot check its desired output for postdictive completeness.

# Postdictive Completeness

- ▶ A learner  $h$  is called **postdictively complete** iff  $\forall g, k, h(g[k])$  correctly postdicts  $g[k]$  ( $= g(0), \dots, g(k-1)$ ), i.e., the programs  $h(g[k])$  correctly compute  $g$  on all inputs  $< k$ .
- ▶ The class of all sets of functions **GEx**-learnable postdictively completely is denoted by **GPcpEx**.
- ▶ **Fact (Barzdin '74, Blum and Blum '75):** There are Ex-learnable sets of functions, which are not learnable by a postdictively complete learner (a.k.a. the **Inconsistency Phenomenon**).

$$\text{GPcpEx} \subsetneq \text{GEx}.$$

- ▶ **Why?** Because of the undecidability of the halting problem, so that the learner cannot check its desired output for postdictive completeness.

# Postdictive Completeness

- ▶ A learner  $h$  is called **postdictively complete** iff  $\forall g, k, h(g[k])$  correctly postdicts  $g[k]$  ( $= g(0), \dots, g(k-1)$ ), i.e., the programs  $h(g[k])$  correctly compute  $g$  on all inputs  $< k$ .
- ▶ The class of all sets of functions **GEx**-learnable postdictively completely is denoted by **GPcpEx**.
- ▶ **Fact (Barzdin '74, Blum and Blum '75):** There are Ex-learnable sets of functions, which are not learnable by a postdictively complete learner (a.k.a. the **Inconsistency Phenomenon**).

$$\mathbf{GPcpEx} \subsetneq \mathbf{GEx}.$$

- ▶ **Why?** Because of the undecidability of the halting problem, so that the learner cannot check its desired output for postdictive completeness.

# Postdictive Completeness

- ▶ A learner  $h$  is called **postdictively complete** iff  $\forall g, k, h(g[k])$  correctly postdicts  $g[k]$  ( $= g(0), \dots, g(k-1)$ ), i.e., the programs  $h(g[k])$  correctly compute  $g$  on all inputs  $< k$ .
- ▶ The class of all sets of functions **GEx**-learnable postdictively completely is denoted by **GPcpEx**.
- ▶ **Fact (Barzdin '74, Blum and Blum '75):** There are Ex-learnable sets of functions, which are not learnable by a postdictively complete learner (a.k.a. the **Inconsistency Phenomenon**).

$$\mathbf{GPcpEx} \subsetneq \mathbf{GEx}.$$

- ▶ **Why?** Because of the undecidability of the halting problem, so that the learner cannot check its desired output for postdictive completeness.

Other **learner restrictions** given in the literature:

- ▶ **Conservativeness**: the learner may only change its hypothesis if it contradicts known data; and
- ▶ **Prudence**: the learner may only output hypotheses for functions it successfully learns.
- ▶ **Non-U-Shapedness**: the learner may never abandon a correct conjecture.

Other **learner restrictions** given in the literature:

- ▶ **Conservativeness**: the learner may only change its hypothesis if it contradicts known data; and
- ▶ **Prudence**: the learner may only output hypotheses for functions it successfully learns.
- ▶ **Non-U-Shapedness**: the learner may never abandon a correct conjecture.

Other **learner restrictions** given in the literature:

- ▶ **Conservativeness**: the learner may only change its hypothesis if it contradicts known data; and
- ▶ **Prudence**: the learner may only output hypotheses for functions it successfully learns.
- ▶ **Non-U-Shapedness**: the learner may never abandon a correct conjecture.

Other **learner restrictions** given in the literature:

- ▶ **Conservativeness**: the learner may only change its hypothesis if it contradicts known data; and
- ▶ **Prudence**: the learner may only output hypotheses for functions it successfully learns.
- ▶ **Non-U-Shapedness**: the learner may never abandon a correct conjecture.

Other **learner restrictions** given in the literature:

- ▶ **Conservativeness**: the learner may only change its hypothesis if it contradicts known data; and
- ▶ **Prudence**: the learner may only output hypotheses for functions it successfully learns.
- ▶ **Non-U-Shapedness**: the learner may never abandon a correct conjecture.

# Memory Limited Learners

- ▶ Given learner  $h$  and learnee  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **It-learning sequence**  $p$  of  $h$  on  $g$  by

$$\begin{aligned} p(0) &= 0; \\ \forall k : p(k+1) &= h(p(k), k, g(k)). \end{aligned}$$

- ▶  $h$  **ItEx-learns**  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **ItEx-learnable** iff there is a learner  $h$  **ItEx-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Results:

$$\text{ItEx} \subsetneq \text{GEx} \text{ and } \text{ItPcpEx} = \text{GPcpEx}.$$

# Memory Limited Learners

- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **It-learning sequence**  $p$  of  $h$  on  $g$  by

$$\begin{aligned} p(0) &= 0; \\ \forall k : p(k+1) &= h(p(k), k, g(k)). \end{aligned}$$

- ▶  $h$  **ItEx-learns**  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **ItEx-learnable** iff there is a learner  $h$  **ItEx-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Results:

$$\text{ItEx} \subsetneq \text{GEx} \text{ and } \text{ItPcpEx} = \text{GPcpEx}.$$

# Memory Limited Learners

- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **It-learning sequence**  $p$  of  $h$  on  $g$  by

$$\begin{aligned} p(0) &= 0; \\ \forall k : p(k+1) &= h(p(k), k, g(k)). \end{aligned}$$

- ▶  $h$  **ItEx-learns**  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **ItEx-learnable** iff there is a learner  $h$  **ItEx-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Results:

$$\text{ItEx} \subsetneq \text{GEx} \text{ and } \text{ItPcpEx} = \text{GPcpEx}.$$

# Memory Limited Learners

- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **It-learning sequence**  $p$  of  $h$  on  $g$  by

$$\begin{aligned} p(0) &= 0; \\ \forall k : p(k+1) &= h(p(k), k, g(k)). \end{aligned}$$

- ▶  $h$  **ItEx-learns**  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **ItEx-learnable** iff there is a learner  $h$  **ItEx-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Results:

$$\text{ItEx} \subsetneq \text{GEx} \text{ and } \text{ItPcpEx} = \text{GPcpEx}.$$

# Memory Limited Learners

- ▶ Given learner  $h$  and learner  $g : \mathbb{N} \rightarrow \mathbb{N}$ , define the **It-learning sequence**  $p$  of  $h$  on  $g$  by

$$\begin{aligned} p(0) &= 0; \\ \forall k : p(k+1) &= h(p(k), k, g(k)). \end{aligned}$$

- ▶  $h$  **ItEx-learns**  $g$  iff, for some  $i$ ,  $p(i)$  computes  $g$  and  $p(i) = p(i+1) = p(i+2) = \dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **ItEx-learnable** iff there is a learner  $h$  **ItEx-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Results:

$$\mathbf{ItEx} \subsetneq \mathbf{GEx} \text{ and } \mathbf{ItPcpEx} = \mathbf{GPcpEx}.$$

# Extrapolation

- ▶ Reminder:
- ▶ For learner  $h$  and learner  $g$ , the **G-learning** sequence  $p$  of  $h$  on  $g$  is defined by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GM-learns**<sup>4</sup>  $g$  iff, for some  $i$ ,  $p(i) = g(i)$ ,  
 $p(i+1) = g(i+1)$ ,  $p(i+2) = g(i+2)$ ,  $\dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GM-learnable** iff there is a learner  $h$   
**GM-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Result:

$$\mathbf{GEx} \subsetneq \mathbf{GM}.$$

---

<sup>4</sup>M for “matches”.

► **Reminder:**

- For learner  $h$  and learner  $g$ , the **G-learning** sequence  $p$  of  $h$  on  $g$  is defined by

$$\forall k : p(k) = h(g[k]).$$

- $h$  **GM-learns**<sup>4</sup>  $g$  iff, for some  $i$ ,  $p(i) = g(i)$ ,  
 $p(i+1) = g(i+1)$ ,  $p(i+2) = g(i+2)$ ,  $\dots$
- $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GM-learnable** iff there is a learner  $h$   
**GM-learning** every  $g \in \mathcal{S}$ .
- **Example Result:**

$$\mathbf{GEx} \subsetneq \mathbf{GM}.$$

---

<sup>4</sup>M for “matches”.

- ▶ Reminder:
- ▶ For learner  $h$  and learner  $g$ , the **G-learning** sequence  $p$  of  $h$  on  $g$  is defined by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GM-learns**<sup>4</sup>  $g$  iff, for some  $i$ ,  $p(i) = g(i)$ ,  
 $p(i+1) = g(i+1)$ ,  $p(i+2) = g(i+2)$ ,  $\dots$
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GM-learnable** iff there is a learner  $h$   
**GM-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Result:

$$\text{GEx} \subsetneq \text{GM}.$$

---

<sup>4</sup>M for “matches”.

- ▶ Reminder:
- ▶ For learner  $h$  and learner  $g$ , the **G-learning** sequence  $p$  of  $h$  on  $g$  is defined by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GM-learns**<sup>4</sup>  $g$  iff, for some  $i$ ,  $p(i) = g(i)$ ,  
 $p(i+1) = g(i+1)$ ,  $p(i+2) = g(i+2)$ ,  $\dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GM-learnable** iff there is a learner  $h$   
**GM-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Result:

$$\text{GEx} \subsetneq \text{GM}.$$

---

<sup>4</sup>M for “matches”.

- ▶ Reminder:
- ▶ For learner  $h$  and learner  $g$ , the **G-learning** sequence  $p$  of  $h$  on  $g$  is defined by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GM-learns**<sup>4</sup>  $g$  iff, for some  $i$ ,  $p(i) = g(i)$ ,  
 $p(i+1) = g(i+1)$ ,  $p(i+2) = g(i+2)$ ,  $\dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GM-learnable** iff there is a learner  $h$  **GM-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Result:

$$\text{GEx} \subsetneq \text{GM}.$$

---

<sup>4</sup>M for “matches”.

- ▶ Reminder:
- ▶ For learner  $h$  and learner  $g$ , the **G-learning** sequence  $p$  of  $h$  on  $g$  is defined by

$$\forall k : p(k) = h(g[k]).$$

- ▶  $h$  **GM-learns**<sup>4</sup>  $g$  iff, for some  $i$ ,  $p(i) = g(i)$ ,  
 $p(i+1) = g(i+1)$ ,  $p(i+2) = g(i+2)$ ,  $\dots$ .
- ▶  $\mathcal{S} \subseteq \mathbb{N} \rightarrow \mathbb{N}$  is called **GM-learnable** iff there is a learner  $h$   
**GM-learning** every  $g \in \mathcal{S}$ .
- ▶ Example Result:

$$\mathbf{GEx} \subsetneq \mathbf{GM}.$$

---

<sup>4</sup>M for “matches”.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting next week on Wednesday (September 30th)
- ▶ on techniques and results in computational learning in the limit
- ▶ 45 minutes each.

Topics:

Lecture 1 Basics of Computability Theory and basic Learning;

Lecture 2 The Inconsistency Phenomenon;

Lecture 3 Memory limited learners.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting **next week on Wednesday** (September 30th)
- ▶ on **techniques** and **results** in computational learning in the limit
- ▶ **45 minutes** each.

Topics:

Lecture 1 **Basics** of Computability Theory and basic Learning;

Lecture 2 The **Inconsistency Phenomenon**;

Lecture 3 **Memory limited** learners.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting **next week on Wednesday** (September 30th)
- ▶ on **techniques** and **results** in computational learning in the limit
- ▶ **45 minutes** each.

Topics:

Lecture 1 **Basics** of Computability Theory and basic Learning;

Lecture 2 The **Inconsistency Phenomenon**;

Lecture 3 **Memory limited** learners.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting **next week on Wednesday** (September 30th)
- ▶ on **techniques** and **results** in computational learning in the limit
- ▶ **45 minutes** each.

Topics:

Lecture 1 **Basics** of Computability Theory and basic Learning;

Lecture 2 The **Inconsistency Phenomenon**;

Lecture 3 **Memory limited** learners.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting **next week on Wednesday** (September 30th)
- ▶ on **techniques** and **results** in computational learning in the limit
- ▶ **45 minutes** each.

Topics:

Lecture 1 **Basics** of Computability Theory and basic Learning;

Lecture 2 The **Inconsistency Phenomenon**;

Lecture 3 **Memory limited** learners.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting **next week on Wednesday** (September 30th)
- ▶ on **techniques** and **results** in computational learning in the limit
- ▶ **45 minutes** each.

Topics:

Lecture 1 **Basics** of Computability Theory and basic Learning;

Lecture 2 The **Inconsistency Phenomenon**;

Lecture 3 **Memory limited** learners.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting **next week on Wednesday** (September 30th)
- ▶ on **techniques** and **results** in computational learning in the limit
- ▶ **45 minutes** each.

Topics:

Lecture 1 **Basics** of Computability Theory and basic Learning;

Lecture 2 The **Inconsistency Phenomenon**;

Lecture 3 **Memory limited** learners.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting **next week on Wednesday** (September 30th)
- ▶ on **techniques** and **results** in computational learning in the limit
- ▶ **45 minutes** each.

Topics:

Lecture 1 **Basics** of Computability Theory and basic Learning;

Lecture 2 The **Inconsistency Phenomenon**;

Lecture 3 **Memory limited** learners.

# Mini-Series on Computational Learning in the Limit

I plan to give a mini-series of lecture

- ▶ starting **next week on Wednesday** (September 30th)
- ▶ on **techniques** and **results** in computational learning in the limit
- ▶ **45 minutes** each.

Topics:

Lecture 1 **Basics** of Computability Theory and basic Learning;

Lecture 2 The **Inconsistency Phenomenon**;

Lecture 3 **Memory limited** learners.

Thank You.