Kolmogorov Complexity and How it Illuminates our Limitations to Let Machines Learn Simple Functions

Lukas Rüttgers

IIIS, Tsinghua University

July 3, 2024

Motivation

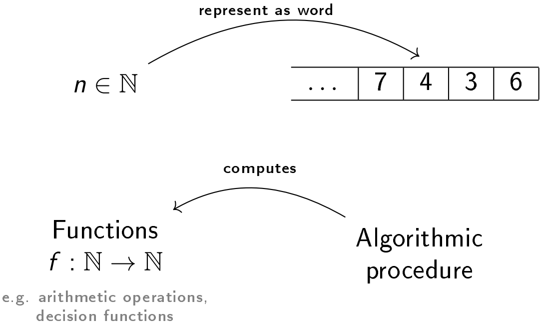
How humans learn arithmetic:

$n \in \mathbb{N} \xrightarrow{\text{represent as word}}$

...	7	4	3	6

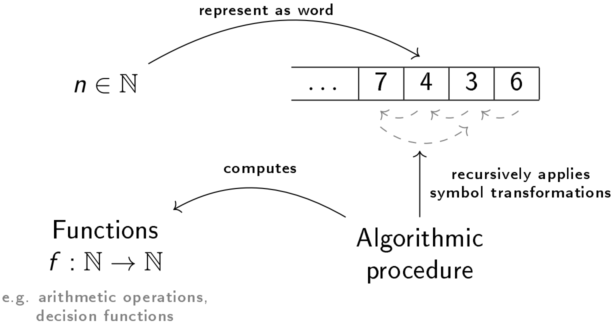
Motivation

How humans learn arithmetic:



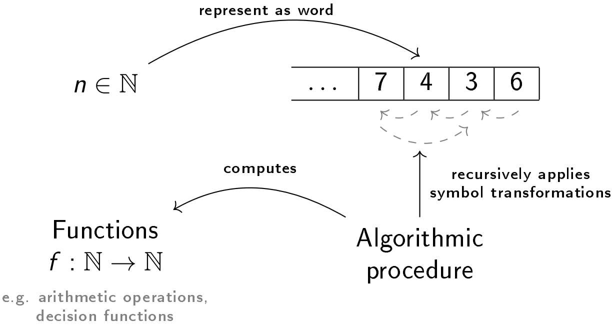
Motivation

How humans learn arithmetic:



Motivation

How humans learn arithmetic:

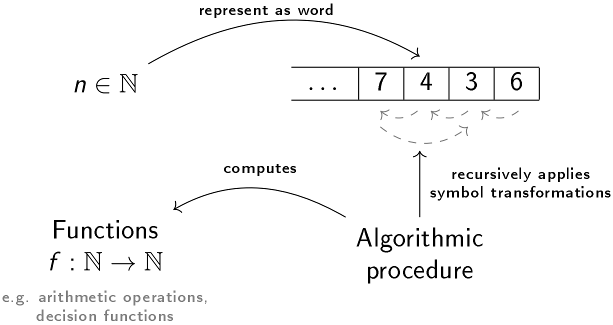


Expedient insights:

1. Recursive algorithmic descriptions generalize to unseen instances

Motivation

How humans learn arithmetic:



Expedient insights:

1. Recursive algorithmic descriptions generalize to unseen instances
2. Generalization ability relies on some inductive simplicity bias

Highlights

Kolmogorov Complexity naturally quantifies the *simplicity* of a function.

In light of this complexity measure we will see that

Highlights

Kolmogorov Complexity naturally quantifies the *simplicity* of a function.

In light of this complexity measure we will see that

- feed-forward neural networks (and any another non-recursive models) cannot express some simple functions

Highlights

Kolmogorov Complexity naturally quantifies the *simplicity* of a function.

In light of this complexity measure we will see that

- feed-forward neural networks (and any another non-recursive models) cannot express some simple functions
- incorporating Kolmogorov complexity as a simplicity bias into learning algorithms allows to
  - learn *any* computable function (e.g. prime numbers) with finite resources
  - learn some functions with *less* samples than usual (e.g. parity functions).

Outline

Supervised Learning

Objective: Learn $f : \mathcal{X} \to \mathcal{Y}$.

Given information: samples $(\underbrace{x_1}_{\text{instances}}, \underbrace{y_1}_{\text{labels}}), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}, y_i = f(x_i)$

Supervised Learning

Objective: Learn $f : \mathcal{X} \to \mathcal{Y}$.

Given information: samples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}, y_i = f(x_i)$

merely minimize?

instances labels

$$\hat{R}(f') := \frac{1}{n} \sum_{i=1}^{n} \ell(f'(x_i), f(x_i)).$$

Supervised Learning

Objective: Learn $f : \mathcal{X} \to \mathcal{Y}$.

Given information: samples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}, y_i = f(x_i)$, $x_i \overset{i.i.d.}{\sim} P_{tr}$.

instances labels

merely minimize?

$$\hat{R}(f') := \frac{1}{n} \sum_{i=1}^{n} \ell(f'(x_i), f(x_i)).$$

fit f inside instance distribution

$$R(f') := \mathbb{E}_{x \sim P_{tr}} \left[\ell(f'(x), f(x))\right].$$

Supervised Learning

Objective: Learn $f : \mathcal{X} \to \mathcal{Y}$.

Given information: samples $(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}, y_i = f(x_i)$, $x_i \overset{i.i.d.}{\sim} P_{tr}$.

instances labels

merely minimize?

$\hat{R}(f') := \frac{1}{n} \sum_{i=1}^{n} \ell(f'(x_i), f(x_i))$.

fit f inside instance distribution

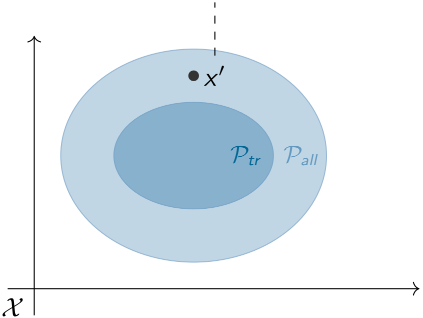
$R(f') := \mathbb{E}_{x \sim P_{tr}} \left[\ell(f'(x), f(x))\right]$.

fit f entirely

How does f' generalize out-of-distribution?

Domain Generalization

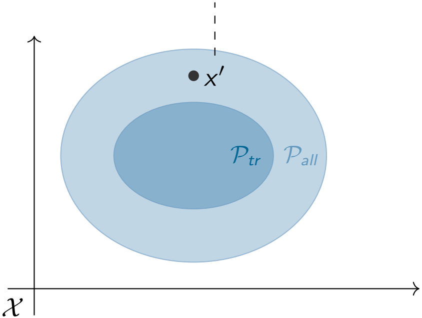
How shall we determine $f'(x')$?



Domain Generalization

How shall we determine $f'(x')$?

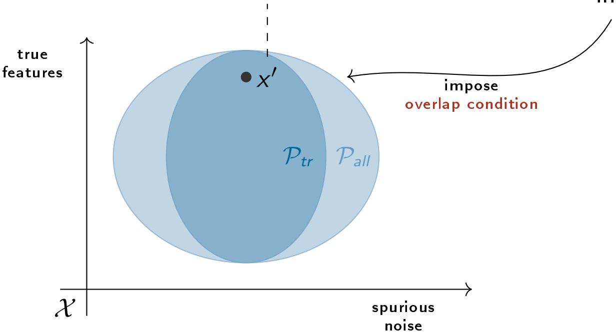
Ahuja et al.: OOD generalization is impossible in such a case [ACZ+21, Theorem 2]



Domain Generalization

How shall we determine $f'(x')$?

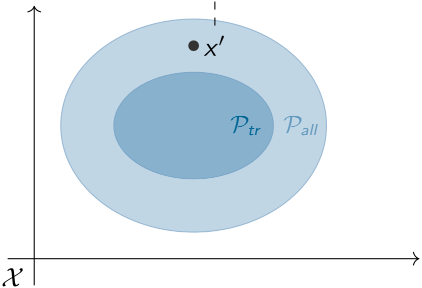
Ahuja et al.: OOD generalization is impossible in such a case [ACZ+21, Theorem 2]



Domain Generalization

How shall we determine $f'(x')$?

Ahuja et al.: OOD generalization is impossible in such a case [ACZ+21, Theorem 2]



But aren't we posing too high demands on "OOD generalization"?

Inferring the simplest consistent function

Our maximum demand on OOD generalization should follow Ockham's razor:

Infer the simplest function that remains consistent with the data.

Inferring the simplest consistent function

Our maximum demand on OOD generalization should follow Ockham's razor:

Infer the simplest function that remains consistent with the data.

How to define the simplicity of a function?

Inferring the simplest consistent function

Our maximum demand on OOD generalization should follow Ockham's razor:

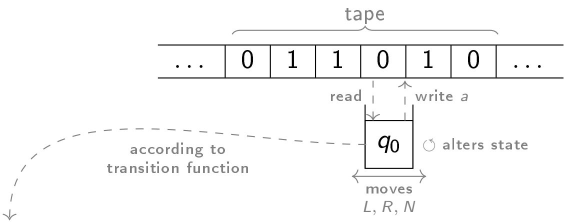
Infer the simplest function that remains consistent with the data.

How to define the simplicity of a function?

Kolmogorov: the shortest description length of a $\underbrace{\text{program}}_{\text{Turing Machine}}$ that produces this function.

Turing Machines

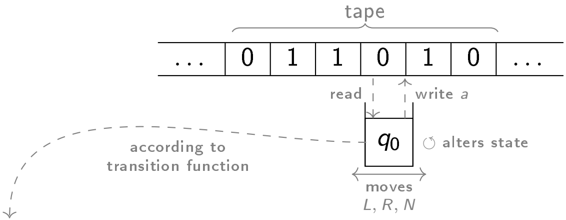
A Turing Machine $\mathcal{T}$:



δ	0	1	B
q_0			
q_1		$(q_{next}, a, \text{dir})$	
$\vdots$			
q_n			

Turing Machines

A Turing Machine $\mathcal{T}$:

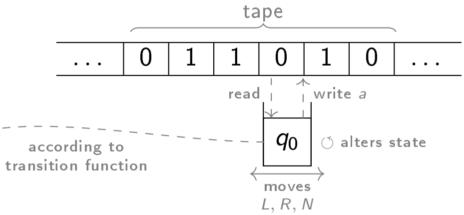


δ	0	1	B
q_0	(q_0, B, R)	(q_1, B, R)	$(q_{halt}, 0, N)$
q_1	(q_0, B, R)	(q_1, B, R)	$(q_{halt}, 1, N)$

Example: Modulo function mod_2

Turing Machines

A Turing Machine $\mathcal{T}$ computes a *partial* computable function



δ	0	1	B
q_0			
q_1		$(q_{next}, a, \text{dir})$	
$\vdots$			
q_n			

$f : D_f \to \{0,1\}^*$, $D_f \subseteq \{0,1\}^*$,

$$f_{\mathcal{T}}(x) := \begin{cases} y, & \mathcal{T} \text{ halts and outputs } y, \\ \perp, & \mathcal{T} \text{ does not halt.} \end{cases}$$

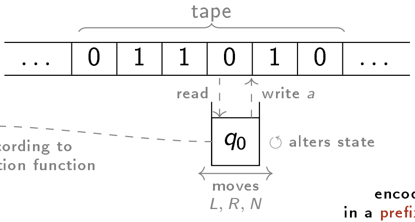
f is *total* computable if $D_f = \{0,1\}^*$.

Turing Machines

A Turing Machine $\mathcal{T}$ computes a *partial* computable function $f : D_f \to \{0,1\}^*, D_f \subseteq \{0,1\}^*$,

$$f_{\mathcal{T}}(x) := \begin{cases} y, & \mathcal{T} \text{ halts and outputs } y, \\ \perp, & \mathcal{T} \text{ does not halt.} \end{cases}$$

f is *total* computable if $D_f = \{0,1\}^*$.



δ	0	1	B
q_0			
q_1		$(q_{next}, a, \text{dir})$	
$\vdots$			
q_n			

encode $\mathcal{T}$ by δ in a prefix-free manner $\longrightarrow$ $\text{enc}(\mathcal{T})$ input z

provide to

simulates

Universal Turing Machine U

$$f_U(x) = \begin{cases} f_{\mathcal{T}}(z), & x = \text{enc}(\mathcal{T})z \\ \perp, & \text{otherwise.} \end{cases}$$

Kolmogorov Complexity

How many bits are needed to describe the encoding of a Turing Machine that computes f?

Equivalence: We write $U(p) \equiv f$ if

- $U(px) = f(x)$ for all $x \in D_f$, and
- U does not halt on px for all $x \in \{0,1\}^* \setminus D_f$.

Kolmogorov Complexity

How many bits are needed to describe the encoding of a Turing Machine that computes f?

Equivalence: We write $U(p) \equiv f$ if

- $U(px) = f(x)$ for all $x \in D_f$, and
- U does not halt on px for all $x \in \{0,1\}^* \setminus D_f$.

We formalise this as the *Kolmogorov complexity* of a computable function f (cf. [LV+08]):

$$K_U(f) := \min_{p \in \{0,1\}^*} \{l(p) \mid U(p) \equiv f\}$$

Kolmogorov Complexity

How many bits are needed to describe the encoding of a Turing Machine that computes f?

Equivalence: We write $U(p) \equiv f$ if

- $U(px) = f(x)$ for all $x \in D_f$, and
- U does not halt on px for all $x \in \{0,1\}^* \setminus D_f$.

We formalise this as the *Kolmogorov complexity* of a computable function f (cf. [LV+08]):

$$K_U(f) := \min_{p \in \{0,1\}^*} \{ l(p) \mid U(p) \equiv f \}$$

Accordingly, the *conditional Kolmogorov complexity* given z is defined as:

$$K_U(f \mid z) := \min_{p,p' \in \{0,1\}^*} \{ l(p) + l(p') \mid U(p\underbrace{[z]}_{\text{self-delimiting encoding}}p') \equiv f \}$$

Our Limitations to Learn Simple Functions in Practice

Models

have access to

function set

$\tau = \{f_1, \dots, f_j\}$

e.g. constants,
activation functions,
arithmetic operations,
logical expressions

Simplicity

Algorithms

Our Limitations to Learn Simple Functions in Practice

Models **Simplicity** **Algorithms**

$$\begin{array}{c} \textbf{Models} \\ \big\downarrow \text{have access to} \\ \text{function set} \\ \tau = \{f_1, \ldots, f_j\} \\ \big\downarrow \text{all finite concatenations} \\ \text{non-recursive functions } \mathcal{F}_\tau \end{array}$$

Our Limitations to Learn Simple Functions in Practice

Models **Simplicity** **Algorithms**

have access to

function set
$\tau = \{f_1, \ldots, f_j\}$

all finite concatenations

non-recursive functions $\mathcal{F}_\tau$

representation class within

non-recursive models over τ

Our Limitations to Learn Simple Functions in Practice

Models **Simplicity** **Algorithms**

have access to

function set
$\tau = \{f_1, \ldots, f_j\}$

all finite concatenations

non-recursive functions $\mathcal{F}_\tau$

representation class within

non-recursive models over τ

cannot express

recursive completion $f_\tau^\Diamond$

Our Limitations to Learn Simple Functions in Practice

Models

have access to

function set

$\tau = \{f_1, \ldots, f_j\}$

all finite concatenations

non-recursive functions $\mathcal{F}_\tau$

representation class within

non-recursive models over τ

cannot express

Simplicity

recursive completion $f_\tau^\diamond$

constantly low given τ

Kolmogorov complexity

Algorithms

Our Limitations to Learn Simple Functions in Practice

Models | **Simplicity** | **Algorithms**

have access to

function set
$\tau = \{f_1, \ldots, f_j\}$

all finite concatenations

non-recursive functions $\mathcal{F}_\tau$

representation class within

non-recursive models over τ

cannot express

recursive completion $f_\tau^\Diamond$

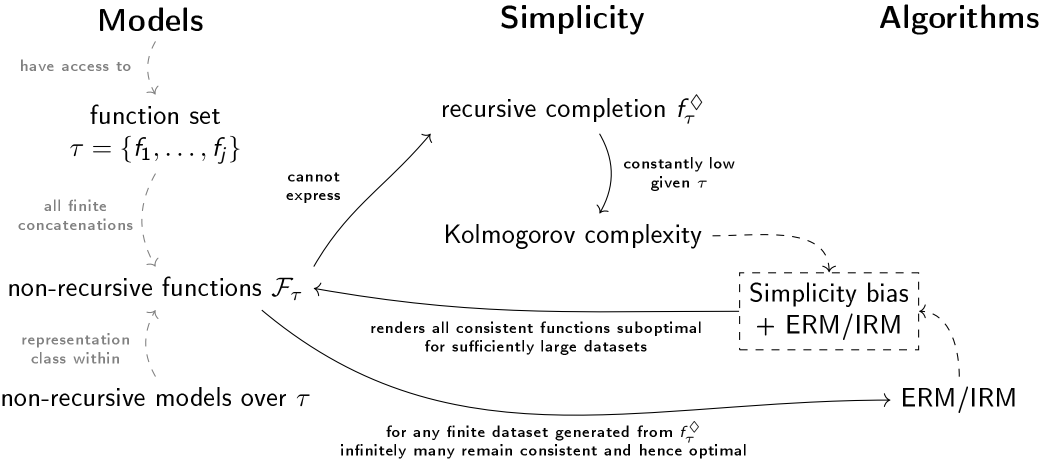
constantly low given τ

Kolmogorov complexity

ERM/IRM

for any finite dataset generated from $f_\tau^\Diamond$
infinitely many remain consistent and hence optimal

Our Limitations to Learn Simple Functions in Practice



Inductive definition of non-recursive functions $\mathcal{F}_\tau$

Given: $\tau := \{f_1, \ldots, f_j\}, f_i : \mathbb{N}^k \to \mathbb{N}$

Base case: Identity function $\mathrm{I} \in \mathcal{F}_\tau$, $\mathrm{I}(n_i) = n_i, n_i \in \mathbb{N}$

Inductive definition of non-recursive functions $\mathcal{F}_\tau$

Given: $\tau := \{f_1, \ldots, f_j\}, f_i : \mathbb{N}^k \to \mathbb{N}$

Base case: Identity function $\mathrm{I} \in \mathcal{F}_\tau$, $\mathrm{I}(n_i) = n_i, n_i \in \mathbb{N}$

$\boldsymbol{n}_m = \left(n_{i_1}, \ldots, n_{i_{\mathrm{ar}(g_m)}}\right)$

Induction step: If $g_1, \ldots, g_k \in \mathcal{F}_\tau$, and $f_i \in \tau$ is k-ary,

$g_1(\boldsymbol{n}_1), \quad \ldots \quad, g_k(\boldsymbol{n}_k)$

then $h := f_i \circ (g_1, \ldots, g_k) \in \mathcal{F}_\tau$.

$h(\boldsymbol{n}) = f_i\big(\, g_1(\boldsymbol{n}_1), \ldots, g_k(\boldsymbol{n}_k) \,\big)$

Inductive definition of non-recursive functions $\mathcal{F}_\tau$

Given: $\tau := \{f_1, \dots, f_j\}, f_i : \mathbb{N}^k \to \mathbb{N}$

Base case: Identity function $\mathrm{I} \in \mathcal{F}_\tau$, $\mathrm{I}(n_i) = n_i, n_i \in \mathbb{N}$

Induction step: If $g_1, \dots, g_k \in \mathcal{F}_\tau$, and $f_i \in \tau$ is k-ary,

then $h := f_i \circ (g_1, \dots, g_k) \in \mathcal{F}_\tau$.

$$\boldsymbol{n}_m = (n_{i_1}, \dots, n_{i_{\mathrm{ar}(g_m)}})$$

$$g_1(\boldsymbol{n}_1), \quad \dots \quad , g_k(\boldsymbol{n}_k)$$

$$h(\boldsymbol{n}) = f_i(\, g_1(\boldsymbol{n}_1), \dots, g_k(\boldsymbol{n}_k) \,)$$

Example (Linear Functions): $\tau = \{f_0, f_1, f_+\}$, $f_0(n_i) = 0$, $f_1(n_i) = 1$, $f_+(n_i, n_j) = n_i + n_j$.

Then, $\mathcal{F}_\tau = \{f : f(n) = an + b \mid a, b \in \mathbb{N}\}$.

Roadmap

Models | **Simplicity** | **Algorithms**

have access to

function set
$\tau = \{f_1, \dots, f_j\}$

all finite concatenations

non-recursive functions $\mathcal{F}_\tau$

representation class within

non-recursive models over τ

cannot express

recursive completion $f_\tau^\diamond$

constantly low given τ

Kolmogorov complexity

Simplicity bias + ERM/IRM

renders all consistent functions suboptimal for sufficiently large datasets

ERM/IRM

for any finite dataset generated from $f_\tau^\diamond$ infinitely many remain consistent and hence optimal

Recursive concatenation

Recursive concatenation of k-ary f:

$$f^{(0)}(n) := n, \qquad n \in \mathbb{N} \tag{1}$$

$$f^{(m+1)}(n) := f\big(\underbrace{f^{(m)}(n), \dots, f^{(m)}(n)}_{k \text{ times}}\big), \qquad n, m \in \mathbb{N}. \tag{2}$$

Recursive concatenation

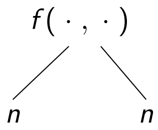
Recursive concatenation of k-ary f:

$$f^{(0)}(n) := n, \qquad n \in \mathbb{N} \tag{1}$$

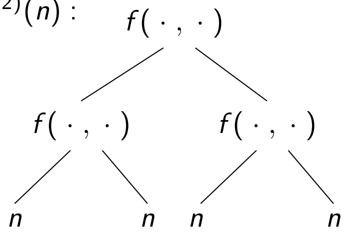
$$f^{(m+1)}(n) := f\big(\underbrace{f^{(m)}(n), \dots, f^{(m)}(n)}_{k \text{ times}}\big), \qquad n, m \in \mathbb{N}. \tag{2}$$

Example for 2-ary f:

$f^{(1)}(n)$:



$f^{(2)}(n)$:



Recursive Completion

Function set $\tau = \{f_1, \ldots, f_j\}$

Recursive concatenation $f^{(m)}(n)$

Recursive Completion

Function set $\tau = \{f_1, \ldots, f_j\}$

sum up

$f_\tau := \sum_{i=1}^{j} f_i$

Recursive concatenation $f^{(m)}(n)$

Recursive Completion

Function set $\tau = \{f_1, \ldots, f_j\}$

sum up

$f_\tau := \sum_{i=1}^{j} f_i$

Recursive concatenation $f^{(m)}(n)$

special case

Recursive completion $f^{\Diamond}(n) := f^{(n)}(n)$

Recursive Completion

Function set $\tau = \{f_1, \ldots, f_j\}$

sum up

$f_\tau := \sum_{i=1}^{j} f_i$

Recursive concatenation $f^{(m)}(n)$

special case

Recursive completion $f^{\lozenge}(n) := f^{(n)}(n)$

combine

Recursive completion over τ

$f_\tau^{\lozenge}(n) := \sum_{i=1}^{j} f_i\left(f_\tau^{(n-1)}(n)\right)$

Why Recursive Completion Excels Every Non-Recursive Function

$f_T^{\Diamond}(n) = \Sigma_{i=1}^{2} f_i^{\Diamond}(n)$ ------ f_1 const., f_2 str. mon. incr. 3-ary

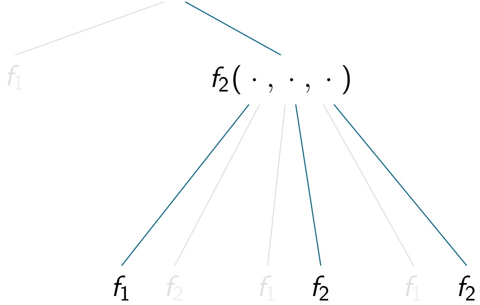
$f = f_2(f_1, f_2(\dots), f_2(\dots))$

Why Recursive Completion Excels Every Non-Recursive Function

$$f_T^{\lozenge}(n) = \Sigma_{i=1}^{2} f_i^{\lozenge}(n)$$

f_1 const.,
f_2 str. mon. incr. 3-ary

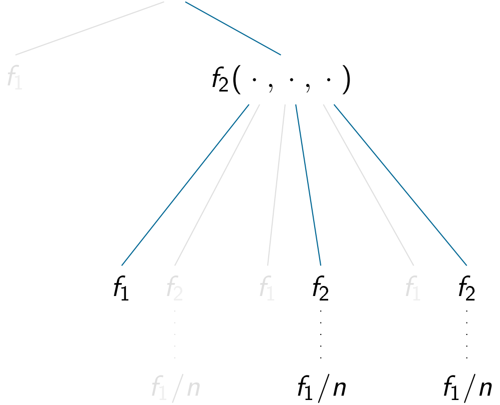
$$f = f_2(f_1, f_2(\dots), f_2(\dots))$$



Why Recursive Completion Excels Every Non-Recursive Function

$f_T^{\Diamond}(n) = \Sigma_{i=1}^{2} f_i^{\Diamond}(n)$ - - - - - - f_1 const., f_2 str. mon. incr. 3-ary

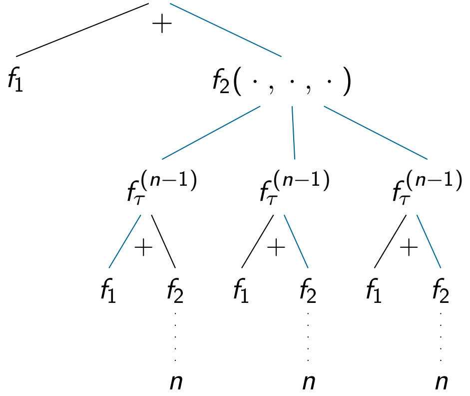
$$f = f_2(f_1, f_2(\dots), f_2(\dots))$$



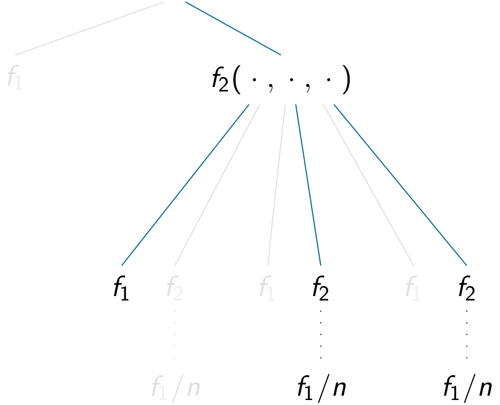
Why Recursive Completion Excels Every Non-Recursive Function

$$f_T^{\Diamond}(n) = \Sigma_{i=1}^{2} f_i^{\Diamond}(n)$$

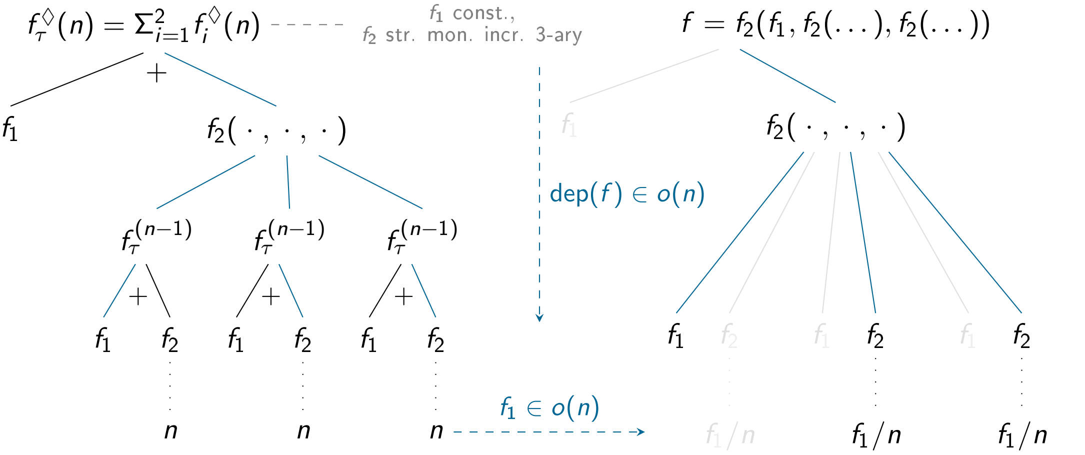
f_1 const.,
f_2 str. mon. incr. 3-ary



$$f = f_2(f_1, f_2(\dots), f_2(\dots))$$



Why Recursive Completion Excels Every Non-Recursive Function

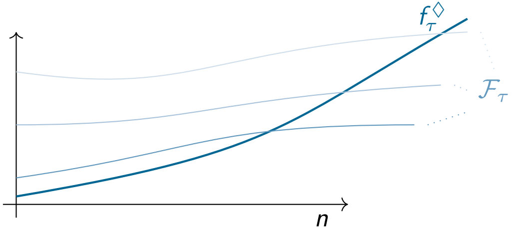


Non-recursion functions do not capture recursive completion

Fix an arbitrary function set $\tau = \{f_1, \ldots, f_j\}, f_i : \mathbb{N}^k \to \mathbb{N}$, where

- each f_i is strictly monotonously increasing or bounded,
- some strictly monotonously increasing f_i has arity $\mathrm{ar}(f_i) > 1$ (e.g. f_+).

For any non-recursive function $f \in \mathcal{F}_\tau$, there is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $f_\tau^\lozenge(n) > f(n)$.

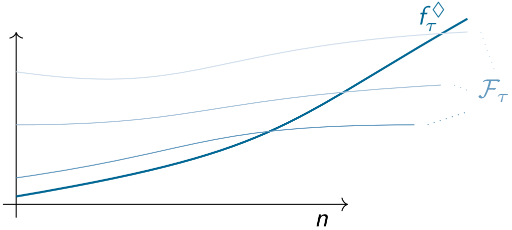


Non-recursion functions do not capture recursive completion

Fix an arbitrary function set $\tau = \{f_1, \ldots, f_j\}, f_i : \mathbb{N}^k \to \mathbb{N}$, where

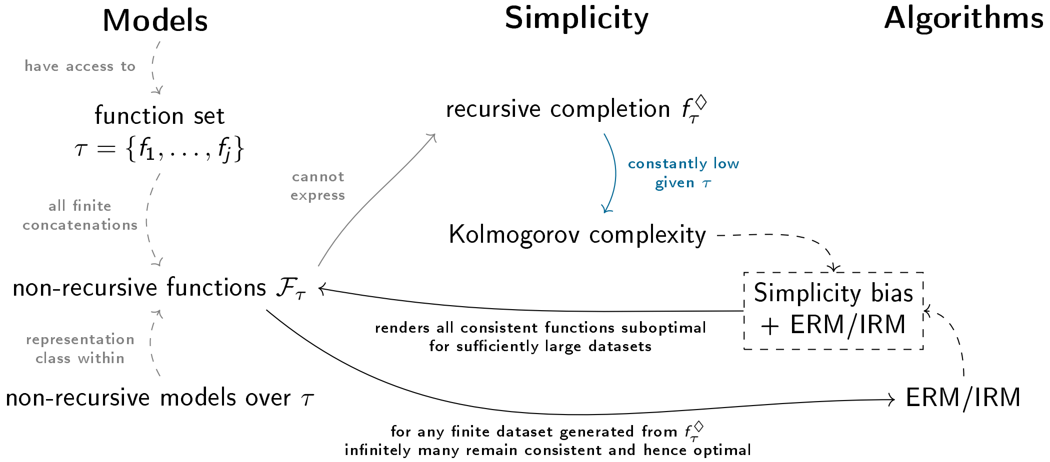
- each f_i is strictly monotonously increasing or bounded,
- some strictly monotonously increasing f_i has arity $\mathrm{ar}(f_i) > 1$ (e.g. f_+).

For any non-recursive function $f \in \mathcal{F}_\tau$, there is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $f_\tau^\Diamond(n) > f(n)$.



This result can be extended to $f_i : \mathbb{Z}^k \to \mathbb{Z}$ with realistic assumptions.

Roadmap


Models
Simplicity
Algorithms
have access to
function set
$\tau = \{f_1, \ldots, f_j\}$
all finite
concatenations
non-recursive functions $\mathcal{F}_\tau$
representation
class within
non-recursive models over τ
cannot
express
recursive completion $f_\tau^\Diamond$
constantly low
given τ
Kolmogorov complexity
Simplicity bias
+ ERM/IRM
renders all consistent functions suboptimal
for sufficiently large datasets
ERM/IRM
for any finite dataset generated from $f_\tau^\Diamond$
infinitely many remain consistent and hence optimal

Uniform Simplicity of Recursive Completion

Uniform TM $\mathcal{T}_\Diamond$:

input tape

...	$\mathrm{enc}(\mathcal{T}_1)$	$\mathrm{ar}(f_1)$	...	$\mathrm{enc}(\mathcal{T}_j)$	$\mathrm{ar}(f_j)$	n	...

loop tape

...		...

computation tape

...		...

output tape

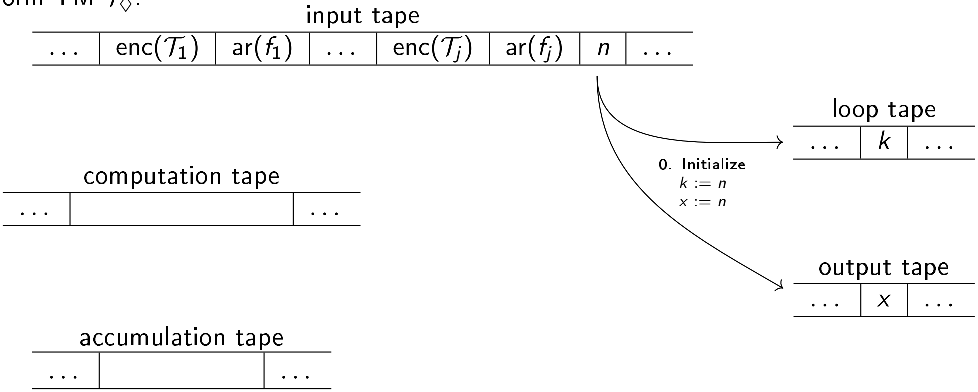
...		...

accumulation tape

...		...

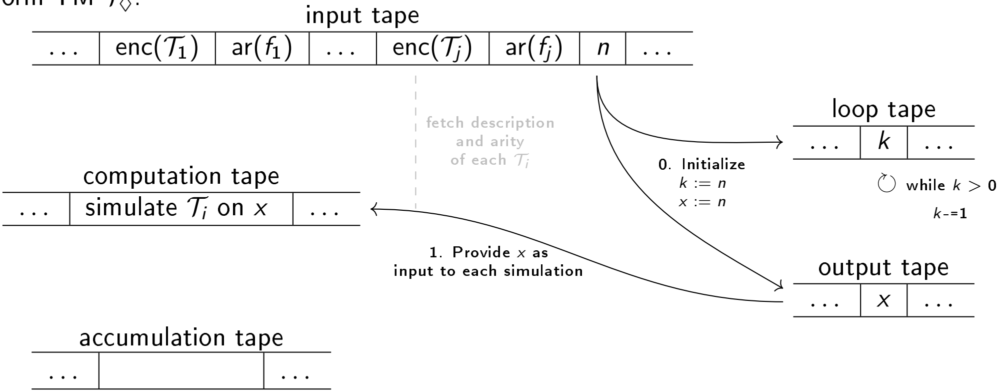
Uniform Simplicity of Recursive Completion

Uniform TM $\mathcal{T}_\Diamond$:



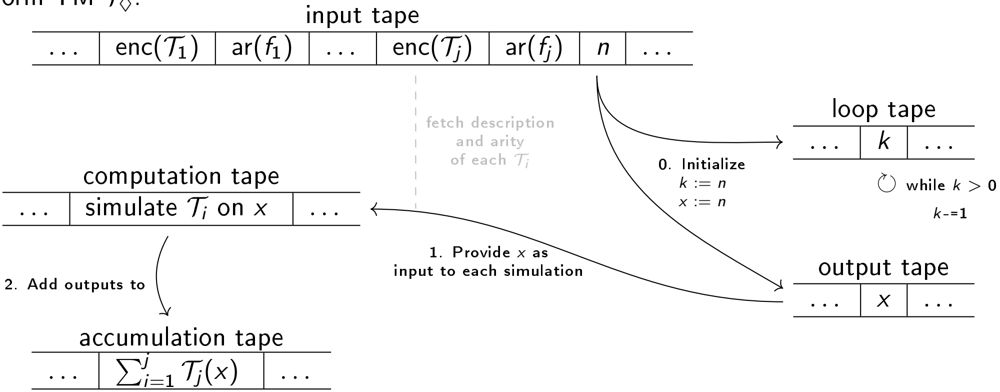
Uniform Simplicity of Recursive Completion

Uniform TM $\mathcal{T}_\Diamond$:



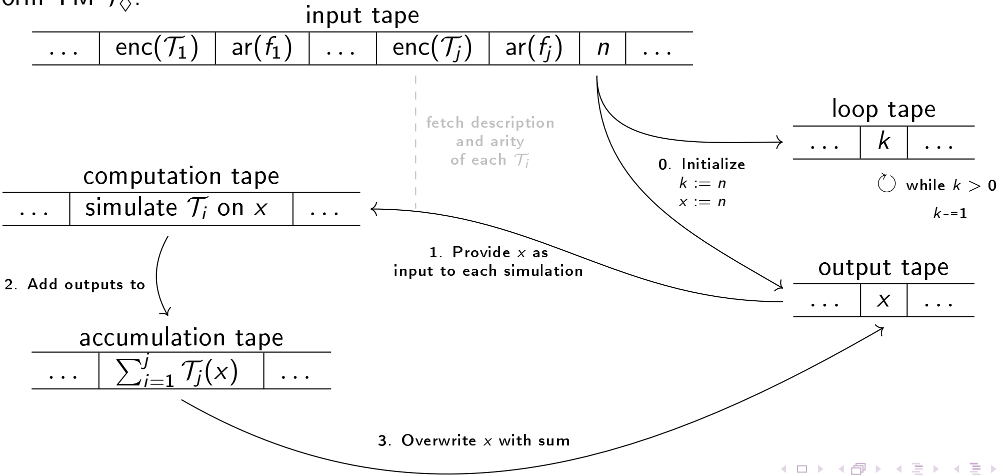
Uniform Simplicity of Recursive Completion

Uniform TM $\mathcal{T}_\Diamond$:



Uniform Simplicity of Recursive Completion

Uniform TM $\mathcal{T}_\Diamond$:



Roadmap

Models | **Simplicity** | **Algorithms**

have access to

function set
$\tau = \{f_1, \ldots, f_j\}$

all finite concatenations

non-recursive functions $\mathcal{F}_\tau$

representation class within

non-recursive models over τ

cannot express

recursive completion $f_\tau^\Diamond$

constantly low given τ

Kolmogorov complexity

Simplicity bias
+ ERM/IRM

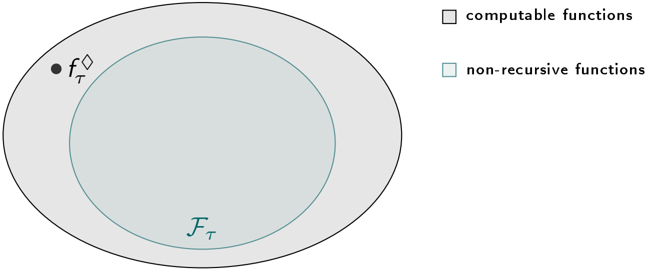
renders all consistent functions suboptimal
for sufficiently large datasets

ERM/IRM

for any finite dataset generated from $f_\tau^\Diamond$
infinitely many remain consistent and hence optimal

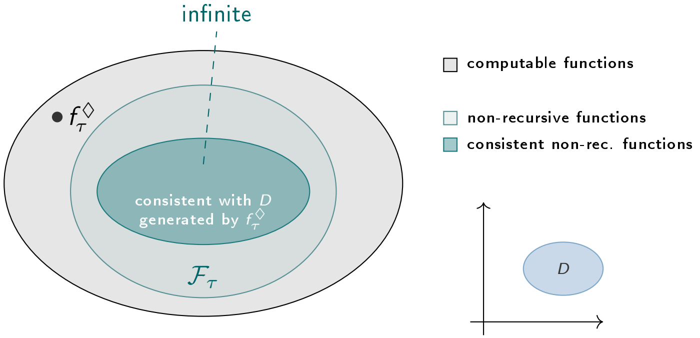
Eliminating non-recursive functions with a simplicity bias

For any $f' \in \mathcal{F}_\tau$, there exists an n_0 such that $f_\tau^\lozenge(n) > f'(n)$ for all $n \geq n_0$.



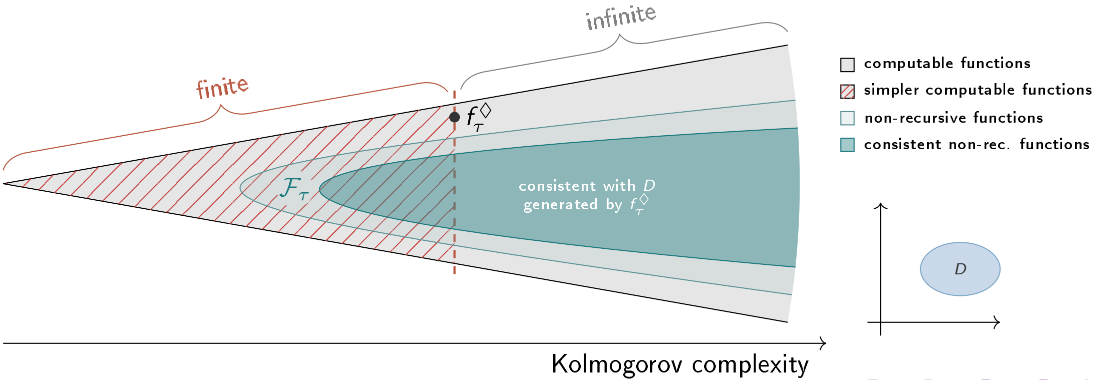
Eliminating non-recursive functions with a simplicity bias

For any $f' \in \mathcal{F}_\tau$, there exists an n_0 such that $f_\tau^\Diamond(n) > f'(n)$ for all $n \geq n_0$.
But non-recursive functions can still memorize the training data.



Eliminating non-recursive functions with a simplicity bias

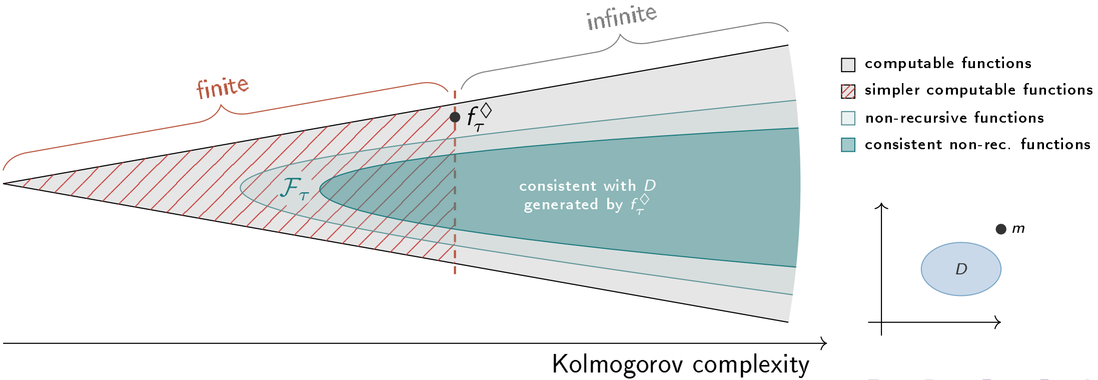
For any $f' \in \mathcal{F}_\tau$, there exists an n_0 such that $f_\tau^\Diamond(n) > f'(n)$ for all $n \geq n_0$.



Eliminating non-recursive functions with a simplicity bias

For any $f' \in \mathcal{F}_\tau$, there exists an n_0 such that $f_\tau^\diamondsuit(n) > f'(n)$ for all $n \geq n_0$.

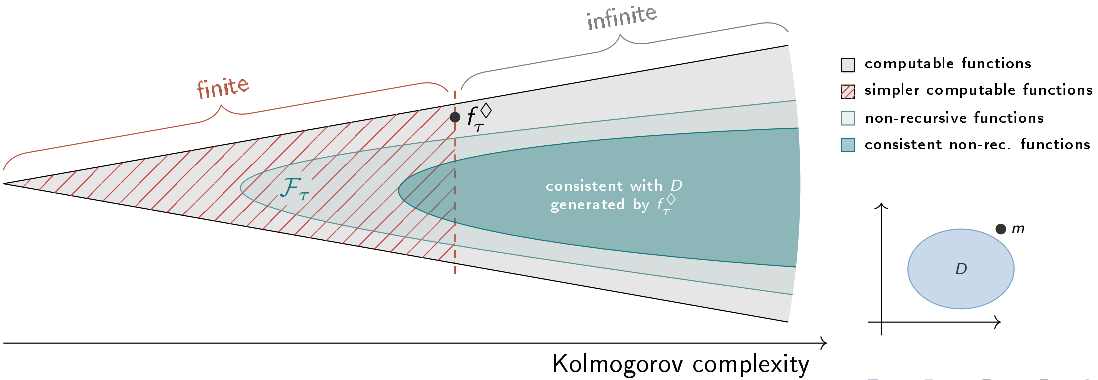
$\longrightarrow$ Include large enough sample $(m, f_\tau^\diamondsuit(m))$ into dataset D.



Eliminating non-recursive functions with a simplicity bias

For any $f' \in \mathcal{F}_\tau$, there exists an n_0 such that $f_\tau^\Diamond(n) > f'(n)$ for all $n \geq n_0$.

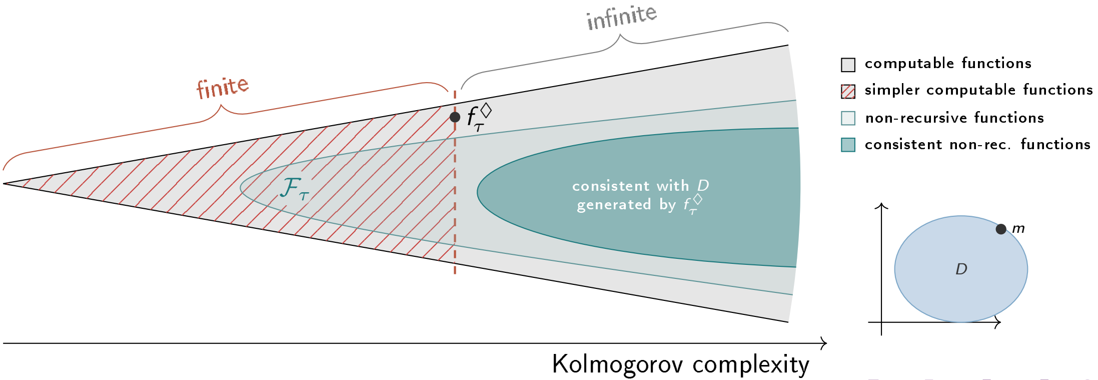
$\longrightarrow$ Include large enough sample $(m, f_\tau^\Diamond(m))$ into dataset D.



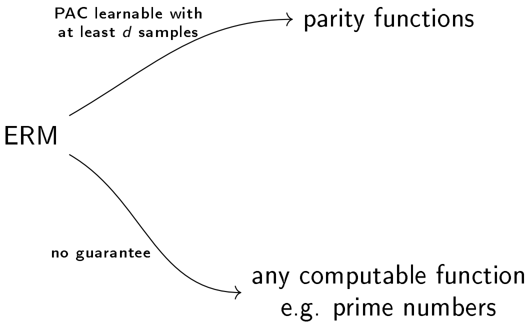
Eliminating non-recursive functions with a simplicity bias

For any $f' \in \mathcal{F}_\tau$, there exists an n_0 such that $f_\tau^\Diamond(n) > f'(n)$ for all $n \geq n_0$.

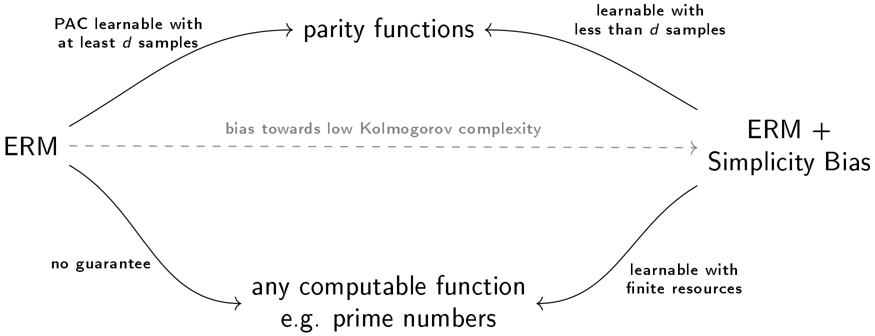
$\longrightarrow$ Include large enough sample $(m, f_\tau^\Diamond(m))$ into dataset D.



Teaser


PAC learnable with
at least d samples
parity functions
ERM
no guarantee
any computable function
e.g. prime numbers

Teaser


PAC learnable with
at least d samples
parity functions
learnable with
less than d samples
bias towards low Kolmogorov complexity
ERM
ERM +
Simplicity Bias
no guarantee
any computable function
e.g. prime numbers
learnable with
finite resources

PAC learning computable functions

PAC learnable

$\mathcal{H}$ is PAC learnable if there is

- a learning algorithm A *(ERM)*
- a sample number threshold $n_0(\varepsilon, \delta)$

such that for

- any error and failure probabilities $\varepsilon, \delta \in [0, 1)$,
- any hypothesis $h \in \mathcal{H}$, } hypothesis conditions
- any marginal distribution $P : \mathcal{X} \to [0, 1]$, and
- any dataset $D = \{(X_i, h(X_i)) \mid i = 1, \ldots, n\}, X_i \overset{i.i.d}{\sim} P$, with $n \geq n_0(\varepsilon, \delta)$,

} data conditions

$$\Pr\left[R(A(D)) \leq \varepsilon\right] \geq 1 - \delta.$$

PAC learning computable functions

PAC learnable $\xleftarrow{\text{VC dimension}}$ ~~$\mathcal{H}$ finite?~~
PAC learnable $\xLeftarrow{\text{Rademacher complexity}}$ ~~$\mathcal{H}$ bounded?~~

$\mathcal{H}$ is PAC learnable if there is

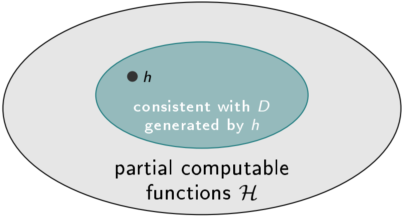
- a learning algorithm A *(ERM)*
- a sample number threshold $n_0(\varepsilon, \delta)$

such that for

- any error and failure probabilities $\varepsilon, \delta \in [0, 1)$,
- any hypothesis $h \in \mathcal{H}$, } hypothesis conditions
- any marginal distribution $P : \mathcal{X} \to [0, 1]$, and
- any dataset $D = \{(X_i, h(X_i)) \mid i = 1, \ldots, n\}, X_i \overset{i.i.d}{\sim} P$, with $n \geq n_0(\varepsilon, \delta)$,

} data conditions

$$\Pr\left[R(A(D)) \leq \varepsilon\right] \geq 1 - \delta.$$



PAC learning computable functions

PAC learnable

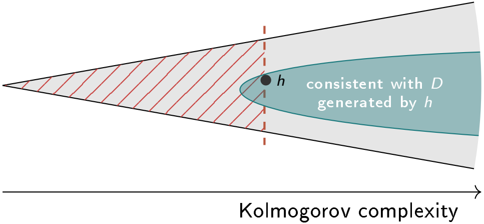
$\mathcal{H}$ is PAC learnable if there is

- a learning algorithm A ***(ERM + Simplicity Bias)***
- a sample number threshold $\boldsymbol{n_0(\delta, k)}$

such that for

- any failure probability $\delta \in (0, 1)$,
- any hypothesis $h \in \mathcal{H}$ **with Kolmogorov complexity** $\boldsymbol{k = K(h)}$, } hypothesis conditions
- any marginal distribution $P : \mathcal{X} \to [0, 1]$, and
- any dataset $D = \{(X_i, h(X_i)) \mid i = 1, \ldots, n\}, X_i \overset{i.i.d}{\sim} P$, with $n \geq \boldsymbol{n_0(\delta, k)}$,

} data conditions



$$\Pr\underbrace{\left[\ \boldsymbol{A(D) = h}\ \right]}_{\text{perfect learnability}} \geq 1 - \delta.$$

Why conditioning learnability on the sample size is impossible in general

- **Example 1:** Unbounded Kolmogorov Complexity, but one sample suffices

- **Example 2:** Low Kolmogorov Complexity, but infinite dataset insufficient

Why conditioning learnability on the sample size is impossible in general

▽ **Example 1:** Unbounded Kolmogorov Complexity, but one sample suffices

$D_y := \{(0, y)\}, y \in \{0, 1\}^*$.
For each y, there is a different simplest consistent function.
But any infinite function class is unbounded in terms of Kolmogorov complexity.

▷ **Example 2:** Low Kolmogorov Complexity, but infinite dataset insufficient

Why conditioning learnability on the sample size is impossible in general

▷ **Example 1:** Unbounded Kolmogorov Complexity, but one sample suffices

▽ **Example 2:** Low Kolmogorov Complexity, but infinite dataset insufficient

We want to learn the modulo function $\text{mod}_2(x) = x \bmod 2$.
But the infinite dataset $D = \{(2n, 0) \mid n \in \mathbb{N}\}$ leaves the (simpler) constant function $f_0(x) = 0$ consistent.

Alternative conditions on the data

Learnability

$\mathcal{H}$ is learnable if there is

- a learning algorithm A ***(ERM + Simplicity Bias)***
- ~~a sample number threshold $n_0(\delta, k)$~~

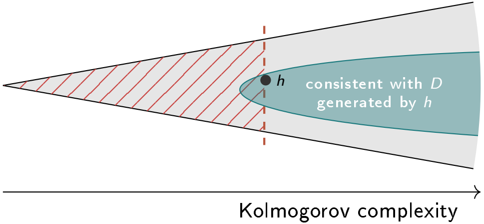
such that for

- any failure probability $\delta \in (0, 1)$,
- any hypothesis $h \in \mathcal{H}$ with Kolmogorov complexity $k = K(h)$, } hypothesis conditions

} data conditions

What conditions do D and P need to fulfil?

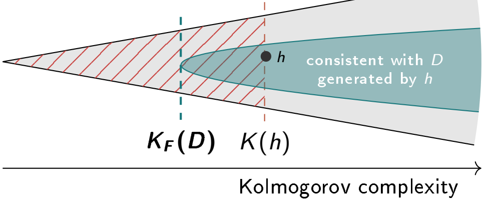
$$\Pr\Big[\underbrace{\boldsymbol{A(D) = h}}_{\text{h is the simplest consistent function}}\Big] \geq 1 - \delta.$$



Conditioning learnability on functional information

Define the functional information in D as

$$K_F(D) := \min_{p \in \{0,1\}^*} \{ l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D \}.$$

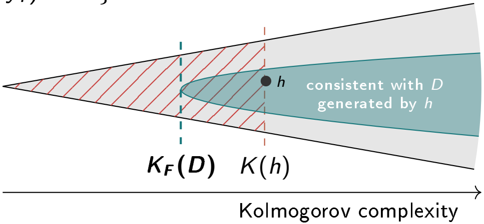


Conditioning learnability on functional information

Define the functional information in D as

$$K_F(D) := \min_{p \in \{0,1\}^*} \{ l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D \}.$$

This quantifies the information that datasets convey about the functions that could have generated them.

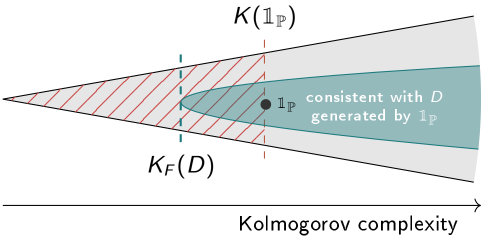


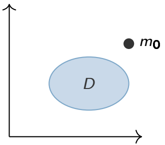
Look at the prior examples anew.

	True function	Dataset	Sample Size	$K_F(D)$
Ex. 1	f_y	D_y	1	$K_F(D_y) = K(f_y)$
Ex. 2	mod_2	D_0	∞	$K_F(D_0) \leq K(f_0) < K(\mathrm{mod}_2)$

Teaching prime numbers by enumerating them

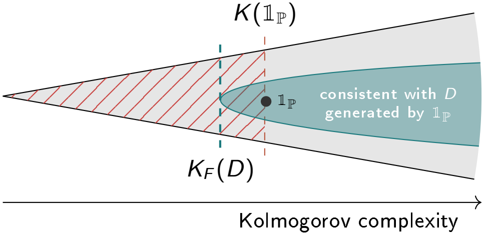
Consider the *prime number decision function* $\mathbb{1}_{\mathbb{P}}(n) = \mathbb{1}\{n \in \mathbb{P}\}$.
There exists an m_0 such that any dataset D that contains $(n, \mathbb{1}_{\mathbb{P}}(n))$ for all $n \leq m_0$ renders $\mathbb{1}_{\mathbb{P}}$ the simplest consistent function with D among all computable decision functions over $\mathbb{N}$.

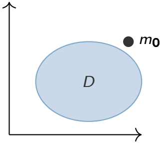




Teaching prime numbers by enumerating them

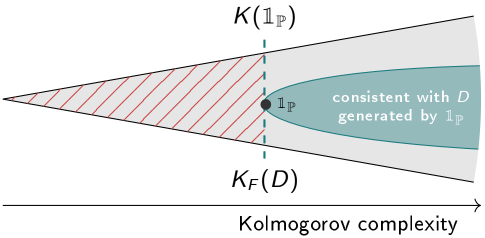
Consider the *prime number decision function* $\mathbb{1}_{\mathbb{P}}(n) = \mathbb{1}\{n \in \mathbb{P}\}$.
There exists an m_0 such that any dataset D that contains $\big(n, \mathbb{1}_{\mathbb{P}}(n)\big)$ for all $n \leq m_0$ renders $\mathbb{1}_{\mathbb{P}}$ the simplest consistent function with D among all computable decision functions over $\mathbb{N}$.

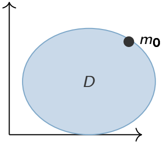




Teaching prime numbers by enumerating them

Consider the *prime number decision function* $\mathbb{1}_{\mathbb{P}}(n) = \mathbb{1}\{n \in \mathbb{P}\}$.
There exists an m_0 such that any dataset D that contains $\left(n, \mathbb{1}_{\mathbb{P}}(n)\right)$ for all $n \leq m_0$ renders $\mathbb{1}_{\mathbb{P}}$ the simplest consistent function with D among all computable decision functions over $\mathbb{N}$.

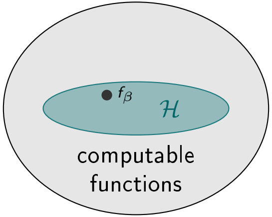




Learning parity functions with less samples

Let $\mathcal{H} = \left\{ f_\beta : \{0,1\}^d \to \{0,1\}, f(x) = \langle \beta, x \rangle \bmod 2 \mid \beta \in \{0,1\}^d \right\}$ be the class of parity functions over d-dimensional binary inputs.
Let $P = \mathrm{Ber}\left(\frac{1}{2}\right)^{\otimes d}$ be the uniform distribution over strings in $\{0,1\}^d$.

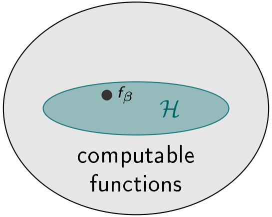
$$\Pr_{x \sim P}\left[f'_\beta(x) = f_\beta(x)\right] = \frac{1}{2}.$$



Learning parity functions with less samples

Let $\mathcal{H} = \{f_\beta : \{0,1\}^d \to \{0,1\}, f(x) = \langle \beta, x \rangle \bmod 2 \mid \beta \in \{0,1\}^d\}$ be the class of parity functions over d-dimensional binary inputs.
Let $P = \mathrm{Ber}\left(\frac{1}{2}\right)^{\otimes d}$ be the uniform distribution over strings in $\{0,1\}^d$.

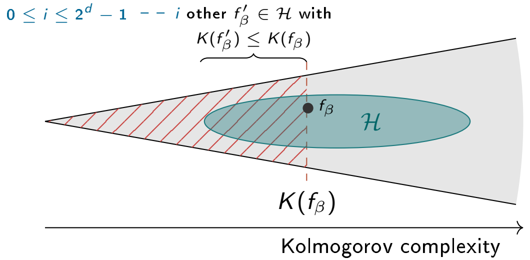
$\Pr_{x \sim P}[f'_\beta(x) = f_\beta(x)] = \frac{1}{2}$. $\Longrightarrow$ At least d samples necessary to render f_β the only consistent function.



Learning parity functions with less samples

Let $\mathcal{H} = \{f_\beta : \{0,1\}^d \to \{0,1\}, f(x) = \langle \beta, x \rangle \bmod 2 \mid \beta \in \{0,1\}^d\}$ be the class of parity functions over d-dimensional binary inputs.
Let $P = \text{Ber}(\frac{1}{2})^{\otimes d}$ be the uniform distribution over strings in $\{0,1\}^d$.

$$\Pr_{x \sim P}\left[f'_\beta(x) = f_\beta(x)\right] = \frac{1}{2}.$$



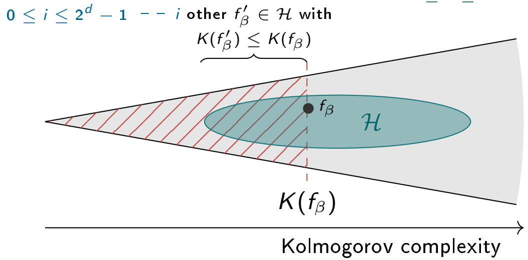
Learning parity functions with less samples

Let $\mathcal{H} = \{ f_\beta : \{0,1\}^d \to \{0,1\}, f(x) = \langle \beta, x \rangle \bmod 2 \mid \beta \in \{0,1\}^d \}$ be the class of parity functions over d-dimensional binary inputs.
Let $P = \mathrm{Ber}\left(\frac{1}{2}\right)^{\otimes d}$ be the uniform distribution over strings in $\{0,1\}^d$.

$\Pr_{x \sim P}\left[f'_\beta(x) = f_\beta(x)\right] = \frac{1}{2}.$

k samples render f_β the *simplest* consistent function with probability at least $1 - i \cdot \left(\frac{1}{2^k}\right)$.

$0 \leq k \leq d$



Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{ l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D \}$$

concatenate samples

$$K_{JF}(D) = K\left([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n]\right)$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound		
▷ Inconsistency implies lower bound		
▷ Monotonicity for supersets		
▷ Invariance under sample permutation		

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D\}$$

concatenate samples

$$K_{JF}(D) = K\left([y_1, \dots, y_n] \mid [x_1, \dots, x_n]\right)$$

	$K_F(D)$	$K_{JF}(D)$
▽ Consistency implies upper bound	If f is consistent with D, then $K_F(D) \le K(f)$.	
▷ Inconsistency implies lower bound		
▷ Monotonicity for supersets		
▷ Invariance under sample permutation		

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{ l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D \}$$

concatenate samples

$$K_{JF}(D) = K\left([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n]\right)$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound	✓	
▽ Inconsistency implies lower bound	If any f with $K(f) < k$ is inconsistent with D, then $K_F(D) \geq k$.	
▷ Monotonicity for supersets		
▷ Invariance under sample permutation		

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D\}$$

concatenate samples

$$K_{JF}(D) = K([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n])$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound	✓	
▷ Inconsistency implies lower bound	✓	
▽ Monotonicity for supersets	Any $D' \supset D$ adds constraints, hence $K_F(D) \leq K_F(D')$.	
▷ Invariance under sample permutation		

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D\}$$

concatenate samples

$$K_{JF}(D) = K\left([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n]\right)$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound	✓	
▷ Inconsistency implies lower bound	✓	
▷ Monotonicity for supersets	✓	
▽ Invariance under sample permutation	Constraints are unordered.	

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{ l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D \}$$

concatenate samples

$$K_{JF}(D) = K\left([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n]\right)$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound	✓	
▷ Inconsistency implies lower bound	✓	
▷ Monotonicity for supersets	✓	
▷ Invariance under sample permutation	✓	

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D\}$$

concatenate samples

$$K_{JF}(D) = K\big([y_1, \dots, y_n] \mid [x_1, \dots, x_n]\big)$$

	$K_F(D)$	$K_{JF}(D)$
▽ Consistency implies upper bound	✓	If f is consistent with D, then $K_F(D) \leq K(f) + c$.
▷ Inconsistency implies lower bound	✓	
▷ Monotonicity for supersets	✓	
▷ Invariance under sample permutation	✓	

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{ l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D \}$$

concatenate samples

$$K_{JF}(D) = K\left([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n]\right)$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound	✓	$K_F(D) \leq K(f) + c$
▽ Inconsistency implies lower bound	✓	Notwithstanding $f(x_i) \neq y_i$, potentially $f\left([x_1, \ldots, x_n]\right) = [y_1, \ldots, y_n]$.
▷ Monotonicity for supersets	✓	
▷ Invariance under sample permutation	✓	

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D\}$$

concatenate samples

$$K_{JF}(D) = K([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n])$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound	✓	$K_F(D) \leq K(f) + c$
▷ Inconsistency implies lower bound	✓	✗
▽ Monotonicity for supersets	✓	add label as another instance $D = \{(x, y)\} \quad D' = \{(x, y), (y, 0)\}$
▷ Invariance under sample permutation	✓	

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{ l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D \}$$

concatenate samples

$$K_{JF}(D) = K([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n])$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound	✓	$K_F(D) \leq K(f) + c$
▷ Inconsistency implies lower bound	✓	✗
▷ Monotonicity for supersets	✓	✗
▽ Invariance under sample permutation	✓	$\underbrace{01101\ldots111011}_{\text{incompressible}} \xrightarrow{\pi} \underbrace{00\ldots011\ldots1}_{\text{compressible}}$

Weakening functional information spoils desirable properties

Could we weaken the constraint-based formulation of $K_F(D)$?

$$K_F(D) = \min_{p \in \{0,1\}^*} \{l(p) \mid U(px_i) = y_i \text{ for all } (x_i, y_i) \in D\}$$

concatenate samples

$$K_{JF}(D) = K\left([y_1, \ldots, y_n] \mid [x_1, \ldots, x_n]\right)$$

	$K_F(D)$	$K_{JF}(D)$
▷ Consistency implies upper bound	✓	$K_F(D) \leq K(f) + c$
▷ Inconsistency implies lower bound	✓	✗
▷ Monotonicity for supersets	✓	✗
▷ Invariance under sample permutation	✓	✗

Compression algorithms cannot approximate Kolmogorov complexity

Kolmogorov Complexity is incomputable. But is there at least a viable approximation A that satisfies

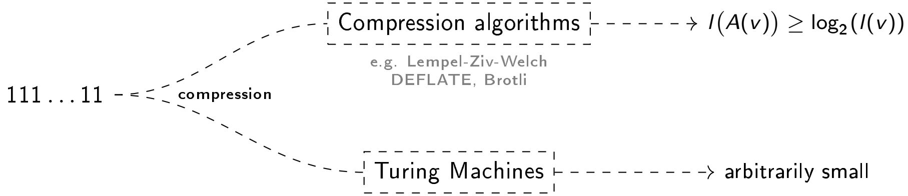
$$A(v) \geq \exp_2^{(k)}(a \cdot A(w) + b) \Rightarrow K(v) \geq K(w) \qquad \text{for some } a, b, k?$$

Compression algorithms cannot approximate Kolmogorov complexity

Kolmogorov Complexity is incomputable. But is there at least a viable approximation A that satisfies

$$A(v) \geq \exp_2^{(k)}(a \cdot A(w) + b) \Rightarrow K(v) \geq K(w) \qquad \text{for some } a, b, k?$$

Compression algorithms were employed in practice [LV⁺08, p. 696].
But their compression ratio is limited.

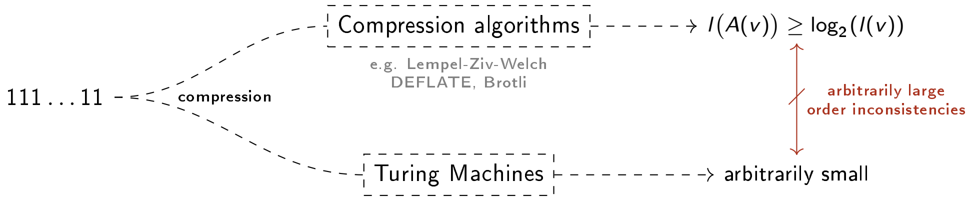


Compression algorithms cannot approximate Kolmogorov complexity

Kolmogorov Complexity is incomputable. But is there at least a viable approximation A that satisfies

$$A(v) \geq \exp_2^{(k)}(a \cdot A(w) + b) \Rightarrow K(v) \geq K(w) \qquad \text{for some } a, b, k?$$

Compression algorithms were employed in practice [LV⁺08, p. 696].
But their compression ratio is limited.



Key takeaways and future research

Key takeaways:

- Recursion is a powerful yet simple mechanism that feed-forward neural networks alone cannot express.

Key takeaways and future research

Key takeaways:

- Recursion is a powerful yet simple mechanism that feed-forward neural networks alone cannot express.
- Out-of-distribution generalization guarantees could draw upon a simplicity bias.

Key takeaways and future research

Key takeaways:

- Recursion is a powerful yet simple mechanism that feed-forward neural networks alone cannot express.
- Out-of-distribution generalization guarantees could draw upon a simplicity bias.
- Yet compression algorithms can not yield approximate guarantees on Kolmogorov complexity.

Key takeaways and future research

Key takeaways:

- Recursion is a powerful yet simple mechanism that feed-forward neural networks alone cannot express.
- Out-of-distribution generalization guarantees could draw upon a simplicity bias.
- Yet compression algorithms can not yield approximate guarantees on Kolmogorov complexity.

Future research:

- Bestow learning algorithms with viable simplicity heuristics.

Key takeaways and future research

Key takeaways:

- Recursion is a powerful yet simple mechanism that feed-forward neural networks alone cannot express.
- Out-of-distribution generalization guarantees could draw upon a simplicity bias.
- Yet compression algorithms can not yield approximate guarantees on Kolmogorov complexity.

Future research:

- Bestow learning algorithms with viable simplicity heuristics.
- How to efficiently learn recursive algorithms over discrete inputs?

References I

[ACZ+21] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Jean-Christophe Gagnon-Audet, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish.
Invariance principle meets information bottleneck for out-of-distribution generalization.
Advances in Neural Information Processing Systems, 34:3438–3450, 2021.

[LV+08] Ming Li, Paul Vitányi, et al.
An introduction to Kolmogorov complexity and its applications, volume 3.
Springer, 2008.