On the Help of Bounded Shot Verifiers, Comparators and Standardisers for Learnability in Inductive Inference

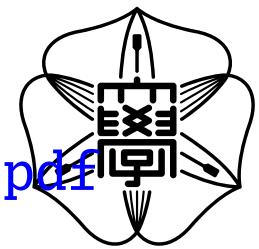
Ziyuan Gao, Sanjay Jain, Frank Stephan, and Thomas Zeugmann

National University of Singapore

Hokkaido University

ALT 2018, Lanzarote, Spain

http://proceedings.mlr.press/v83/gao18a/gao18a.pdf



Introduction I

We deal with the inductive inference of recursively enumerable languages from *positive data* (also called text). If L is a language then every sequence of strings $T = (s_i)_{i \in \mathbb{N}}$ and a *special symbol* # such that $L = \{s_i \mid i \in \mathbb{N}\} \setminus \{\#\}$ is called a *text* for L. Evidence about the target language is then given by growing initial segments of a text. On these segments the learner outputs a number (grammar), which is interpreted w.r.t. a hypothesis space.

Introduction I

We deal with the inductive inference of recursively enumerable languages from *positive data* (also called text). If L is a language then every sequence of strings $T = (s_i)_{i \in \mathbb{N}}$ and a *special symbol* # such that $L = \{s_i \mid i \in \mathbb{N}\} \setminus \{\#\}$ is called a *text* for L. Evidence about the target language is then given by growing initial segments of a text. On these segments the learner outputs a number (grammar), which is interpreted w.r.t. a hypothesis space.

We always assume a *Gödel numbering* $W_0, W_1, W_2, \ldots$ of all r.e. languages as *hypothesis space*. The sequence of hypotheses has then to *converge* to a correct hypothesis for the target language.

Introduction II

We distinguish *explanatory learning* (syntactical convergence), *behaviourally correct learning* (semantical convergence), and *finite learning* (just one hypothesis, also called 1-*shot learning*).

Introduction II

We distinguish *explanatory learning* (syntactical convergence), *behaviourally correct learning* (semantical convergence), and *finite learning* (just one hypothesis, also called *1-shot learning*).

Variations

Standardisation: evidence is provided by an index i (programme) for the target language L from the target class $\mathcal{L}$. Then there must be an index k such that $L = W_k$ and the *standardiser* has to map any i with $L = W_i$ to k.

Introduction II

We distinguish *explanatory learning* (syntactical convergence), *behaviourally correct learning* (semantical convergence), and *finite learning* (just one hypothesis, also called *1-shot learning*).

Variations

Standardisation: evidence is provided by an index i (programme) for the target language L from the target class $\mathcal{L}$. Then there must be an index k such that $L = W_k$ and the *standardiser* has to map any i with $L = W_i$ to k.

This can be done in the limit and/or with a restricted number of shots. If the standardiser is 1-shot, then we call it *finite standardisation*.

Introduction II

We distinguish *explanatory learning* (syntactical convergence), *behaviourally correct learning* (semantical convergence), and *finite learning* (just one hypothesis, also called *1-shot learning*).

Variations

Standardisation: evidence is provided by an index i (programme) for the target language L from the target class $\mathcal{L}$. Then there must be an index k such that $L = W_k$ and the *standardiser* has to map any i with $L = W_i$ to k.

This can be done in the limit and/or with a restricted number of shots. If the standardiser is 1-shot, then we call it *finite standardisation*.

Verifiability: evidence is provided by an index e and a text T for the language L. The *verifier* has to decide whether or not the index input is correct for L. So, the verifier outputs *yes*, *no*, or ?.

Introduction III

Comparability: The *comparator* receives two indices of languages from the *target class* $\mathcal{L}$ as input and has to *decide in the limit* whether or not these indices generate the *same language*. So, the comparator also outputs *yes*, *no*, or ?.

Introduction III

Comparability: The *comparator* receives two indices of languages from the *target class* $\mathcal{L}$ as input and has to *decide in the limit* whether or not these indices generate the *same language*. So, the comparator also outputs *yes*, *no*, or ?.

For verifiers and comparators we mainly consider a bounded number of shots, e.g., 1-shot, 2-shot, or 3-shot. Note that the special symbol ? does *not* count, e.g., ?, ?, *yes* is a 1-shot sequence.
Also, in all models we ask for for one learner which can fulfil the task for all languages in the target class $\mathcal{L}$.

Introduction III

Comparability: The *comparator* receives two indices of languages from the *target class* $\mathcal{L}$ as input and has to *decide in the limit* whether or not these indices generate the *same language*. So, the comparator also outputs *yes*, *no*, or ?.

For verifiers and comparators we mainly consider a bounded number of shots, e.g., 1-shot, 2-shot, or 3-shot. Note that the special symbol ? does *not* count, e.g., ?, ?, *yes* is a 1-shot sequence.
Also, in all models we ask for for one learner which can fulfil the task for all languages in the target class $\mathcal{L}$.

The main question studied is to what extent verifiability, comparability, and standardisability are useful for the inductive inference of classes of recursively enumerable languages.

Introduction IV

For example, we are interested in learning whether or not a verifiable class is also explanatorily learnable, and if it is, whether or not the number of mind changes is preserved.

Introduction IV

For example, we are interested in learning whether or not a verifiable class is also explanatorily learnable, and if it is, whether or not the number of mind changes is preserved.

We distinguish between *r.e. classes, one-one r.e. classes* (of r.e. languages), and indexed families (of recursive languages). A class is said to be an *indexed family* if there is a recursive function f such that for all $i \in \mathbb{N}$ and all strings $x \in \mathbb{N}$, $f(i, x) = 1$ if $x \in W_i$ and $f(i, x) = 0$, otherwise.

Introduction IV

For example, we are interested in learning whether or not a verifiable class is also explanatorily learnable, and if it is, whether or not the number of mind changes is preserved.

We distinguish between *r.e. classes, one-one r.e. classes* (of r.e. languages), and indexed families (of recursive languages). A class is said to be an *indexed family* if there is a recursive function f such that for all $i \in \mathbb{N}$ and all strings $x \in \mathbb{N}$, $f(i, x) = 1$ if $x \in W_i$ and $f(i, x) = 0$, otherwise.

The paper provides an almost complete picture concerning

(1) comparison of learnability with 1-shot verifiability, 1-shot comparability, and 1-shot standardisability;

(2) $\sim$ with 2-shot verifiability, 1-shot comparability, and 2-shot standardisability;

(3) $\sim$ with 3 or more shot verifiability, comparability, and standardisability.

Results I

Let $K =_{df} \{e \mid e \in \mathbb{N},\ \varphi_e(e) \text{ is defined}\}$, where $\varphi_0, \varphi_1, \ldots$ is any fixed Gödel numbering of all partial recursive functions mapping $\mathbb{N}$ to $\mathbb{N}$. We call K the *halting set*.

Theorem 1

The class $\mathcal{K}$ consisting of K and all singleton languages $\{x\}$ with $x \notin K$ is finitely learnable but not finitely verifiable.

Results I

Let $K =_{df} \{e \mid e \in \mathbb{N},\ \varphi_e(e) \text{ is defined}\}$, where $\varphi_0, \varphi_1, \ldots$ is any fixed Gödel numbering of all partial recursive functions mapping $\mathbb{N}$ to $\mathbb{N}$. We call K the *halting set*.

Theorem 1

The class $\mathcal{K}$ consisting of K and all singleton languages $\{x\}$ with $x \notin K$ is finitely learnable but not finitely verifiable.

Proof. A finite verification algorithm on an index e for K and a text $x, x, x, x, \ldots$ would have to eventually output "no" iff $x \notin K$ and that would, combined with the enumeration procedure of K, lead to a decision procedure of K.

Results I

Let $K =_{df} \{e \mid e \in \mathbb{N},\ \varphi_e(e) \text{ is defined}\}$, where $\varphi_0, \varphi_1, \ldots$ is any fixed Gödel numbering of all partial recursive functions mapping $\mathbb{N}$ to $\mathbb{N}$. We call K the *halting set*.

Theorem 1

The class $\mathcal{K}$ consisting of K and all singleton languages $\{x\}$ with $x \notin K$ is finitely learnable but not finitely verifiable.

Proof. A finite verification algorithm on an index e for K and a text $x, x, x, x, \ldots$ would have to eventually output "no" iff $x \notin K$ and that would, combined with the enumeration procedure of K, lead to a decision procedure of K.
Let $\psi(x, y) = 0$, if $y = x$; $\psi(x, y) = \varphi_x(x) \cdot \varphi_y(y)$, otherwise. Now, if $x \notin K$, then $\psi(x, y)$ diverges for all $y \neq x$, and thus $\text{domain}(\psi(x, \cdot)) = \{x\}$. If $x \in K$, then for all $y \neq x$, $\psi(x, y)$ converges iff $y \in K$, and thus $\text{domain}(\psi(x, \cdot)) = \{x\} \cup K = K$.

Results II

Now, the finite learner outputs the canonical index of domain of $\psi(x, \cdot)$ in the acceptable numbering $W_0, W_1, W_2, \ldots$ on input of any text T, if the first non-# symbol in the text T is x (that is, for some i, $T(i) = x$, and $T(i') = \#$ for all $i' < i$). It is easy to verify that the above learner finitely learns the class $\mathcal{K}$. ∎

Remark. The second part of the proof of Theorem 1 also shows that $\mathcal{K}$ is an r.e. class.

Results II

Now, the finite learner outputs the canonical index of domain of $\psi(x, \cdot)$ in the acceptable numbering $W_0, W_1, W_2, \ldots$ on input of any text T, if the first non-# symbol in the text T is x (that is, for some i, $T(i) = x$, and $T(i') = \#$ for all $i' < i$). It is easy to verify that the above learner finitely learns the class $\mathcal{K}$. ∎

Remark. The second part of the proof of Theorem 1 also shows that $\mathcal{K}$ is an r.e. class.

Theorem 2

The uniformly r.e. class $\mathcal{K}$ is neither finitely comparable nor finitely standardisable.

Results II

Now, the finite learner outputs the canonical index of domain of $\psi(x, \cdot)$ in the acceptable numbering $W_0, W_1, W_2, \ldots$ on input of any text T, if the first non-# symbol in the text T is x (that is, for some i, $T(i) = x$, and $T(i') = \#$ for all $i' < i$). It is easy to verify that the above learner finitely learns the class $\mathcal{K}$. ∎

Remark. The second part of the proof of Theorem 1 also shows that $\mathcal{K}$ is an r.e. class.

Theorem 2

The uniformly r.e. class $\mathcal{K}$ is neither finitely comparable nor finitely standardisable.

Proof. Suppose the converse, and let $k \in \mathbb{N}$ be any fixed index of K, i.e., $W_k = K$. Then one can design an algorithm deciding K.

Results III

This algorithm uses the numbering ψ constructed in the second part of the proof of Theorem 1. Since the numbering of the sets $W_0, W_1, W_2, \ldots$ is acceptable, there is a recursive function $c \in \mathcal{R}$ such that $\psi_x = \varphi_{c(x)}$ for all $x \in \mathbb{N}$. Hence $W_{c(x)}$ is equal to K iff $x \in K$ and equal to $\{x\}$ iff $x \notin K$.

Results III

This algorithm uses the numbering ψ constructed in the second part of the proof of Theorem 1. Since the numbering of the sets $W_0, W_1, W_2, \ldots$ is acceptable, there is a recursive function $c \in \mathcal{R}$ such that $\psi_x = \varphi_{c(x)}$ for all $x \in \mathbb{N}$. Hence $W_{c(x)}$ is equal to K iff $x \in K$ and equal to $\{x\}$ iff $x \notin K$.

Consequently, for every $x \in \mathbb{N}$ one runs the finite comparator on input $c(x)$ and k. Note that by construction, $W_k, W_{c(x)} \in \mathcal{K}$ for all $x \in \mathbb{N}$, and thus the finite comparator must be defined on all these inputs. Also, it must return "yes" iff $W_{c(x)} = K$ and "no" otherwise; a contradiction to the undecidability of the set K.

Results III

This algorithm uses the numbering ψ constructed in the second part of the proof of Theorem 1. Since the numbering of the sets $W_0, W_1, W_2, \ldots$ is acceptable, there is a recursive function $c \in \mathcal{R}$ such that $\psi_x = \varphi_{c(x)}$ for all $x \in \mathbb{N}$. Hence $W_{c(x)}$ is equal to K iff $x \in K$ and equal to $\{x\}$ iff $x \notin K$.

Consequently, for every $x \in \mathbb{N}$ one runs the finite comparator on input $c(x)$ and k. Note that by construction, $W_k, W_{c(x)} \in \mathcal{K}$ for all $x \in \mathbb{N}$, and thus the finite comparator must be defined on all these inputs. Also, it must return "yes" iff $W_{c(x)} = K$ and "no" otherwise; a contradiction to the undecidability of the set K.

The second part is shown *mutatis mutandis*. First, we run the finite standardiser on input k to find out to which index k is finitely standardised, say to s.

Results IV

In order to decide K one executes the finite standardiser on input $c(x)$ for any given $x \in \mathbb{N}$. If it returns s, then $x \in K$ and otherwise $x \notin K$. Hence, if there would exist a finite standardiser then K would be decidable, a contradiction. ∎

Results IV

In order to decide K one executes the finite standardiser on input $c(x)$ for any given $x \in \mathbb{N}$. If it returns s, then $x \in K$ and otherwise $x \notin K$. Hence, if there would exist a finite standardiser then K would be decidable, a contradiction. ∎

Remark. Theorem 2 shows that finite learnability does *not* imply finite standardisability if learning classes of r.e. languages is considered.
In contrast, for function learning, every finitely learnable function class is also finitely standardisable as shown by Freivalds and Wiehagen (1979).

Results IV

In order to decide K one executes the finite standardiser on input $c(x)$ for any given $x \in \mathbb{N}$. If it returns s, then $x \in K$ and otherwise $x \notin K$. Hence, if there would exist a finite standardiser then K would be decidable, a contradiction. ∎

Remark. Theorem 2 shows that finite learnability does *not* imply finite standardisability if learning classes of r.e. languages is considered.
In contrast, for function learning, every finitely learnable function class is also finitely standardisable as shown by Freivalds and Wiehagen (1979).

Next, we consider any finite number of shots, say n, and compare standardisability and comparability. The following theorem can be shown:

Results V

Theorem 3

Every $\mathfrak{n}$*-shot standardisable class* $\mathfrak{L}$ *is* $(2\mathfrak{n}-1)$*-shot comparable.*

Results V

Theorem 3

Every n*-shot standardisable class* $\mathcal{L}$ *is* $(2n-1)$*-shot comparable.*

Proof. Let d and e be two indices such that both W_d and W_e are in $\mathcal{L}$. Then the algorithm runs two instances of the standardiser in parallel on the two inputs d and e, respectively, and waits until each of them has produced an output. Then, if the current outputs of the two instances are equal, then the algorithm outputs "yes", and if the two outputs are different, then it outputs "no".
Hence, every mind change of the comparator requires that at least one of the standardisers makes another shot and so one can bound the number of shots of the comparator by $1 + 2 \cdot (n-1) = 2n-1$. ∎

Results VI

A class $\mathcal{L}$ is said to be *inclusion-free* if there are no two languages A and B in the class such that $A \subset B$.

Theorem 4

Any class which is 2-shot comparable must be inclusion-free. Thus, any class which is finitely standardisable must be inclusion-free.

Results VI

A class $\mathcal{L}$ is said to be *inclusion-free* if there are no two languages A and B in the class such that $A \subset B$.

Theorem 4

Any class which is 2-shot comparable must be inclusion-free. Thus, any class which is finitely standardisable must be inclusion-free.

Proof. If the class contains two sets A and B with $A \subset B$, and has a 2-shot comparator F, then using Smullyan's double recursion theorem, one can construct two grammars i and j such that $W_i = W_j = A$, if the comparator never outputs "yes" on input (i, j). If the comparator outputs "yes" on input (i, j) at some point, and then never outputs "no" after that, then $W_i = A$ and $W_j = B$. Otherwise, $W_i = W_j = B$. So it follows that the comparator is wrong on input (i, j). ∎

Results VII

Theorem 5

Let $\mathcal{L}$ be any r.e. class. Then we have

(1) *If $\mathcal{L}$ is finitely verifiable, then $\mathcal{L}$ is also finitely learnable.*

(2) *If $\mathcal{L}$ is finitely comparable then $\mathcal{L}$ is also finitely learnable.*

(3) *If $\mathcal{L}$ is finitely standardisable then $\mathcal{L}$ is also finitely learnable.*

Results VII

Theorem 5

Let $\mathcal{L}$ be any r.e. class. Then we have

(1) *If $\mathcal{L}$ is finitely verifiable, then $\mathcal{L}$ is also finitely learnable.*

(2) *If $\mathcal{L}$ is finitely comparable then $\mathcal{L}$ is also finitely learnable.*

(3) *If $\mathcal{L}$ is finitely standardisable then $\mathcal{L}$ is also finitely learnable.*

Proof. If a class $\mathcal{L}$ has a recursively enumerable list $e_0, e_1, e_2, \ldots$ of indices, then a finite verifier can be turned into a finite learner by dovetailing the enumeration of the indices $e_0, e_1, \ldots$ and by simulating the verifier on e_0 versus T, e_1 versus T, ... until one of them outputs "yes". The finite learner then conjectures the first index e_k, where the simulation gives the answer "yes", i.e., Assertion (1) is shown.

Results VII

Theorem 5

Let $\mathcal{L}$ be any r.e. class. Then we have

(1) *If $\mathcal{L}$ is finitely verifiable, then $\mathcal{L}$ is also finitely learnable.*

(2) *If $\mathcal{L}$ is finitely comparable then $\mathcal{L}$ is also finitely learnable.*

(3) *If $\mathcal{L}$ is finitely standardisable then $\mathcal{L}$ is also finitely learnable.*

Proof. If a class $\mathcal{L}$ has a recursively enumerable list $e_0, e_1, e_2, \ldots$ of indices, then a finite verifier can be turned into a finite learner by dovetailing the enumeration of the indices $e_0, e_1, \ldots$ and by simulating the verifier on e_0 versus T, e_1 versus T, ... until one of them outputs "yes". The finite learner then conjectures the first index e_k, where the simulation gives the answer "yes", i.e., Assertion (1) is shown.

Assertion (2) is more difficult (see the paper). Assertion (3) is then a direct consequence of Assertion (2) and Theorem 3. ∎

Results VIII

Remark. The requirement that $\mathcal{L}$ is an r.e. class in Theorem 5 is important, since we have the following:

Theorem 6

There is a class $\mathcal{F}$ of r.e. languages such that $\mathcal{L}$ is finitely standardisable but neither finitely learnable nor finitely verifiable.

Results VIII

Remark. The requirement that $\mathcal{L}$ is an r.e. class in Theorem 5 is important, since we have the following:

Theorem 6

There is a class $\mathcal{F}$ of r.e. languages such that $\mathcal{L}$ is finitely standardisable but neither finitely learnable nor finitely verifiable.

Proof. Here we consider a class of functions, i.e., any enumeration of the graph of a function f is then a text for f. We define $f_0(x) := 0$ for all $x \in \mathbb{N}$, and for $n \geqslant 1$ we set $f_n(x) := 0$ for all $x \in \mathbb{N} \setminus \{n\}$ and $f_n(n) := 1$. Furthermore, we use $\min_\varphi f$ to denote the least index i such that $\varphi_i = f$. Next, consider the class $\mathcal{F} := \{f_n \mid n \in \mathbb{N},\ n \leqslant \min_\varphi f_n\}$. Then $\mathcal{F}$ is finitely standardisable. Here we need the condition that $n \leqslant \min_\varphi f_n$.

Results IX

To see that $\mathcal{F}$ is not finitely verifiable let $e \in \mathbb{N}$ be an index for f_0, and let T be a text for f_0 such that $T = ((0, f_0(0)), (1, f_0(1)), (2, f_0(2)) \ldots)$. Then, on input e and T, a finite verifier would have to eventually output "yes", since otherwise it could not verify that T is a text for the function f_0. Let this happen when the verifier has seen $T[m]$. Consequently, for every $n > m$ and a text T' in the same order $(0, f_n(0)), (1, f_n(1)), (2, f_n(2)) \ldots$ for f_n it must, on input e and T', also output "yes", a contradiction to the fact that T' is not a text for f_0.

Results IX

To see that $\mathcal{F}$ is not finitely verifiable let $e \in \mathbb{N}$ be an index for f_0, and let T be a text for f_0 such that $T = ((0, f_0(0)), (1, f_0(1)), (2, f_0(2)) \ldots)$. Then, on input e and T, a finite verifier would have to eventually output "yes", since otherwise it could not verify that T is a text for the function f_0. Let this happen when the verifier has seen $T[m]$. Consequently, for every $n > m$ and a text T' in the same order $(0, f_n(0)), (1, f_n(1)), (2, f_n(2)) \ldots$ for f_n it must, on input e and T', also output "yes", a contradiction to the fact that T' is not a text for f_0.

Mutatis mutandis one shows that $\mathcal{F}$ is not finitely learnable. ∎

Results X

Remark. Note that f_0 is an *accumulation point* of the class $\mathcal{F}$ (since $\mathcal{F}$ is infinite). That is, for every $n \in \mathbb{N}$ there is is a function $f \in \mathcal{F}$ such that $f(x) = f_0(x)$ for all $0 \leqslant x \leqslant n$ but $f \neq f_0$. So, if a class is finitely learnable or finitely verifiable, then it cannot contain an accumulation point.

Corollary 7

The collection of finitely learnable classes is incomparable to the collections of finitely standardisable classes and to the collection of finitely verifiable classes.

Results X

Remark. Note that f_0 is an *accumulation point* of the class $\mathcal{F}$ (since $\mathcal{F}$ is infinite). That is, for every $n \in \mathbb{N}$ there is is a function $f \in \mathcal{F}$ such that $f(x) = f_0(x)$ for all $0 \leqslant x \leqslant n$ but $f \neq f_0$. So, if a class is finitely learnable or finitely verifiable, then it cannot contain an accumulation point.

Corollary 7

The collection of finitely learnable classes is incomparable to the collections of finitely standardisable classes and to the collection of finitely verifiable classes.

Theorem 8

If a class $\mathcal{L}$ is finitely standardisable then $\mathcal{L}$ is explanatorily learnable.

Results XI

Theorem 9

If a class has a 2-shot standardiser and has a one-one r.e. numbering, then it is explanatorily learnable.

If a class $\mathcal{L}$ has a 2-shot comparator and a one-one r.e. numbering, then $\mathcal{L}$ is explanatorily learnable.

Proof. difficult, see the paper.

Results XI

Theorem 9

If a class has a 2-shot standardiser and has a one-one r.e. numbering, then it is explanatorily learnable.

If a class $\mathcal{L}$ has a 2-shot comparator and a one-one r.e. numbering, then $\mathcal{L}$ is explanatorily learnable.

Proof. difficult, see the paper.

Theorem 10

There is 2-shot standardisable indexed family which is not conservatively learnable.

Remark. A learner is said to be *conservative* if it performs exclusively justified mind changes. In particular, it can never *overgeneralise*.

Results XII

There are additional results in the paper to complete the overall picture, e.g., we have the following:

Theorem 11

There is an r.e. 2-shot comparable class which is not explanatorily learnable.

Results XII

There are additional results in the paper to complete the overall picture, e.g., we have the following:

Theorem 11

There is an r.e. 2-shot comparable class which is not explanatorily learnable.

Theorem 12

If a uniformly r.e. class $\mathcal{L}$ is 2-shot verifiable, then it is conservatively learnable.

Results XII

There are additional results in the paper to complete the overall picture, e.g., we have the following:

Theorem 11

There is an r.e. 2-shot comparable class which is not explanatorily learnable.

Theorem 12

If a uniformly r.e. class $\mathcal{L}$ is 2-shot verifiable, then it is conservatively learnable.

Note that there are more results in the paper which are omitted here due to the available time.

Conclusions I

Learning models have been introduced, where at least one input is an index, and the second input is a text or another index, resulting in *verifiability* and *comparability*, respectively. We obtained modifications of *standardisability*, a learning model which has been around for quite some time.

Conclusions I

Learning models have been introduced, where at least one input is an index, and the second input is a text or another index, resulting in *verifiability* and *comparability*, respectively. We obtained modifications of *standardisability*, a learning model which has been around for quite some time.

Variations: restricting the number of shots the learner is allowed to make.

Conclusions I

Learning models have been introduced, where at least one input is an index, and the second input is a text or another index, resulting in *verifiability* and *comparability*, respectively. We obtained modifications of *standardisability*, a learning model which has been around for quite some time.

Variations: restricting the number of shots the learner is allowed to make.

Comparisons to standard models (e.g., explanatory inference, behaviourally correct learning).
In depth study of the learning capabilities of these learning models in dependence on the classes to be learnt: arbitrary classes of r.e. languages, uniformly r.e. classes of r.e. languages, 1–1 r.e. classes of r.e. languages, and indexed families.

Conclusions II

In particular, we showed the following:

(1) comparators, standardisers, and verifiers are different with respect to their help for the inductive inference of the respective classes, i.e., verifiability *always implies explanatory learning*, but conservative learning is out of reach in general. Only 2-shot verifiers can always be used to achieve conservative learning.

Conclusions II

In particular, we showed the following:

(1) comparators, standardisers, and verifiers are different with respect to their help for the inductive inference of the respective classes, i.e., verifiability *always implies explanatory learning*, but conservative learning is out of reach in general. Only 2-shot verifiers can always be used to achieve conservative learning.

(2) 2-shot standardisability implies explanatory learning but not conservative learning if the target classes are indexed families or 1–1 r.e. classes. For indexed families any 2-shot comparator and any 2-shot verifier can be used to obtain a conservative learner. In contrast, a 3-shot standardiser cannot be used to obtain even an explanatory learner.

Conclusions II

In particular, we showed the following:

(1) comparators, standardisers, and verifiers are different with respect to their help for the inductive inference of the respective classes, i.e., verifiability *always implies explanatory learning*, but conservative learning is out of reach in general. Only 2-shot verifiers can always be used to achieve conservative learning.

(2) 2-shot standardisability implies explanatory learning but not conservative learning if the target classes are indexed families or 1–1 r.e. classes. For indexed families any 2-shot comparator and any 2-shot verifier can be used to obtain a conservative learner. In contrast, a 3-shot standardiser cannot be used to obtain even an explanatory learner.

(3) Studies revealed some topological constraints for 2-comparability and finite standardisability, i.e., the respective classes must be *inclusion-free*.

Thank you!