TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Learning Structures

Ekaterina Fokina

TU Wien

joint with Nikolay Bazhenov and Luca San Mauro

Leeds-Ghent Virtual Logic Seminar

January 28, 2021

Learning structures

- Suppose we have a class of (countable) structures.

Learning structures

- Suppose we have a class of (countable) structures.
- Suppose we are stage by stage seeing one of the structures from the class: at each step a larger and larger finite piece of the structure.

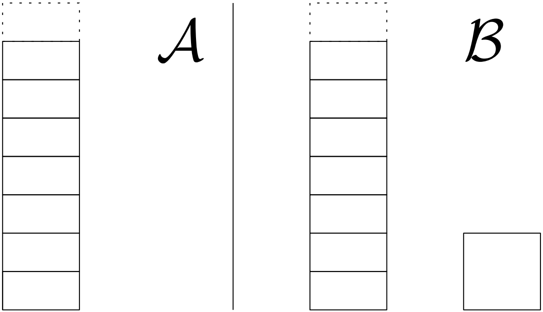
Learning structures

- Suppose we have a class of (countable) structures.
- Suppose we are stage by stage seeing one of the structures from the class: at each step a larger and larger finite piece of the structure.

Question

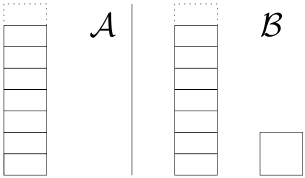
Can we, after finitely many steps, identify the structure (up to an isomorphism or other equivalence relations)?

Example 1



Graphics by Luca San Mauro

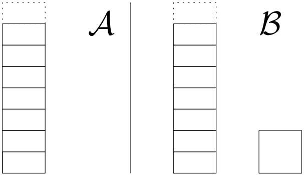
Example 1



$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

Graphics by Luca San Mauro

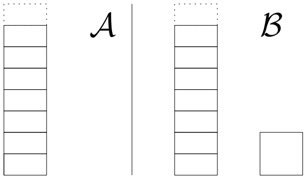
Example 1



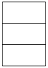
$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

Graphics by Luca San Mauro

Example 1

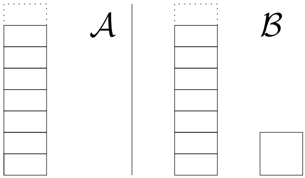


$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

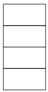


Graphics by Luca San Mauro

Example 1

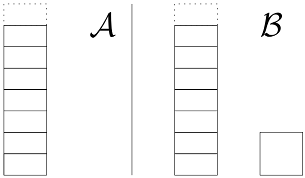


$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

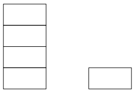


Graphics by Luca San Mauro

Example 1

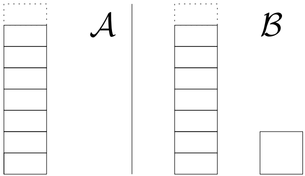


$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

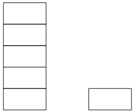


Graphics by Luca San Mauro

Example 1

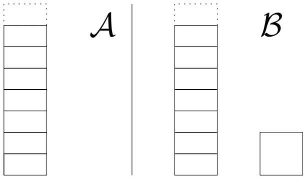


$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

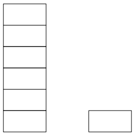


Graphics by Luca San Mauro

Example 1

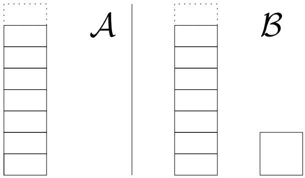


$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

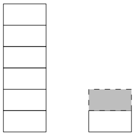


Graphics by Luca San Mauro

Example 1

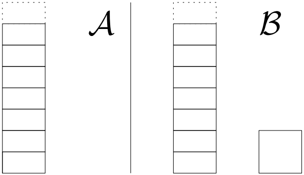


$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

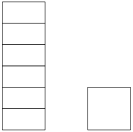


Graphics by Luca San Mauro

Example 1

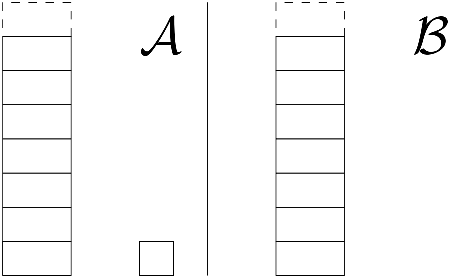


$\mathcal{S}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{B} \urcorner$



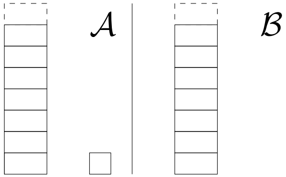
Graphics by Luca San Mauro

Example 2



Graphics by Luca San Mauro

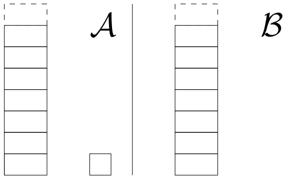
Example 2



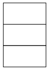
$\mathcal{S} \cong \mathcal{B}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

Graphics by Luca San Mauro

Example 2

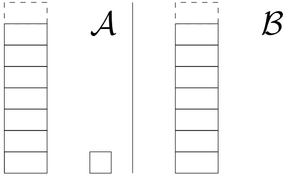


$\mathcal{S} \cong \mathcal{B} \quad \Big| \quad \mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$



Graphics by Luca San Mauro

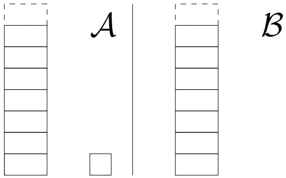
Example 2



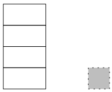
$\mathcal{S} \cong \mathcal{B}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{B} \urcorner$

Graphics by Luca San Mauro

Example 2

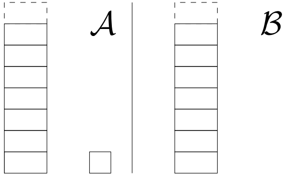


$\mathcal{S} \cong \mathcal{B}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{B} \urcorner$

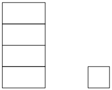


Graphics by Luca San Mauro

Example 2

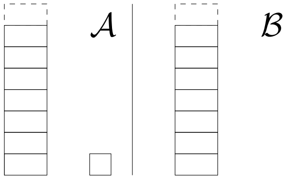


$\mathcal{S} \cong \mathcal{A} \;\Big|\; \mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{B} \urcorner$

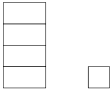


Graphics by Luca San Mauro

Example 2

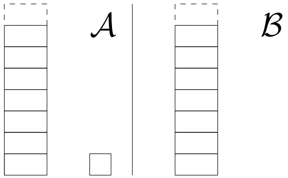


$\mathcal{S} \cong \mathcal{A} \;\Big|\; \mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{B} \urcorner$

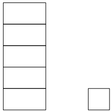


Graphics by Luca San Mauro

Example 2

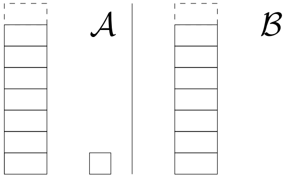


$$\mathcal{S} \cong \mathcal{A} \quad \Big| \quad \mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{B} \urcorner$$

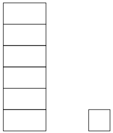


Graphics by Luca San Mauro

Example 2

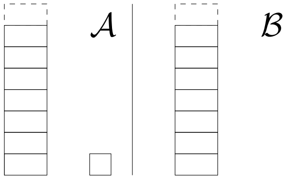


$\mathcal{S} \cong \mathcal{A}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

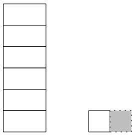


Graphics by Luca San Mauro

Example 2

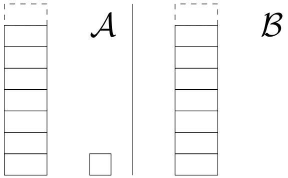


$\mathcal{S} \cong \mathcal{A}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$

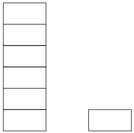


Graphics by Luca San Mauro

Example 2



$\mathcal{S} \cong \mathcal{B}$ | $\mathbf{M}(\mathcal{S}) = \ulcorner \mathcal{A} \urcorner$



Graphics by Luca San Mauro

Conclusion and overview

- How is the structure “revealed”?

Conclusion and overview

- How is the structure "revealed"?
- In this talk we are mostly considering the case when we receive both the positive and the negative information about the structure (motivated by computable structures).

Conclusion and overview

- How is the structure "revealed"?
- In this talk we are mostly considering the case when we receive both the positive and the negative information about the structure (motivated by computable structures).
- Another approach: only the positive information about the structure is revealed (motivated by c.e. structures).

Conclusion and overview

- How is the structure "revealed"?
- In this talk we are mostly considering the case when we receive both the positive and the negative information about the structure (motivated by computable structures).
- Another approach: only the positive information about the structure is revealed (motivated by c.e. structures).
- What does it mean "to classify", or "to identify" the structure?

Conclusion and overview

- How is the structure "revealed"?
- In this talk we are mostly considering the case when we receive both the positive and the negative information about the structure (motivated by computable structures).
- Another approach: only the positive information about the structure is revealed (motivated by c.e. structures).
- What does it mean "to classify", or "to identify" the structure?
- To formalize these and other issues, we use the ideas from computable structure theory and computational learning theory.

Computational Learning Theory

Computational Learning Theory (CLT): deals with the question of how a learner, provided with more and more data about some environment, is eventually able to achieve systematic knowledge about it.

Computational Learning Theory

Computational Learning Theory (CLT): deals with the question of how a learner, provided with more and more data about some environment, is eventually able to achieve systematic knowledge about it.

- (Gold, 1967): language identification.

Computational Learning Theory

Computational Learning Theory (CLT): deals with the question of how a learner, provided with more and more data about some environment, is eventually able to achieve systematic knowledge about it.

- (Gold, 1967): language identification.

Most work in CLT concerns

- either learning of total functions (where the order in which the data is received matters)
- or learning of formal languages (where the order does not matter)

These paradigms model the data to be learned as an unstructured flow — but what if one deals with data having some structural content?

CLT and Structures

More recently researchers applied the machinery of CLT to algebraic structures:

CLT and Structures

More recently researchers applied the machinery of CLT to algebraic structures:

- Glymour, 1985
- Martin, Osherson, 1998
- Stephan, Ventsov, 2001: learning ring ideals of commutative rings.
- Merkle, Stephan, 2004 learning isolated branches on uniformly computable sequences of trees.
- Harizanov, Stephan, 2007: learning subspaces of V_∞.
- Gao, Stephan, Wu, Yamamoto, 2012: learning closed sets in matroids.
- F., Kötzing, San Mauro, 2018: learning equivalence structures.

CLT and Structures

More recently researchers applied the machinery of CLT to algebraic structures:

- Glymour, 1985
- Martin, Osherson, 1998
- Stephan, Ventsov, 2001: learning ring ideals of commutative rings.
- Merkle, Stephan, 2004 learning isolated branches on uniformly computable sequences of trees.
- Harizanov, Stephan, 2007: learning subspaces of V_∞.
- Gao, Stephan, Wu, Yamamoto, 2012: learning closed sets in matroids.
- F., Kötzing, San Mauro, 2018: learning equivalence structures.

Our goal is: to combine the technology of CLT with notions coming from computable structure theory to develop a general framework for learning the isomorphism type of algebraic structures.

Computable Structures

To learn the isomorphism type of a given structure, one should be able to name such an isomorphism type. This is why we focus on the learning of (copies of) computable structures: the name of the isomorphism type of a computable structure $\mathcal{A}$ will be just the index, w.r.t. to some given effective enumeration, of a Turing machine that computes the atomic diagram of $\mathcal{A}$.

Learning should be independent from the way in which data is presented. So, a successful learning procedure should work for all isomorphic copies of a given structure.

Computable structures

- We consider finite (relational) signatures L.

Computable structures

- We consider finite (relational) signatures L.
- We assume that the domain of any countably infinite structure is equal to the set ω of the natural numbers.

Computable structures

- We consider finite (relational) signatures L.
- We assume that the domain of any countably infinite structure is equal to the set ω of the natural numbers.
- We effectively identify, through a fixed Gödel numbering, any sentence about an L-structure with a natural number.

Computable structures

- We consider finite (relational) signatures L.
- We assume that the domain of any countably infinite structure is equal to the set ω of the natural numbers.
- We effectively identify, through a fixed Gödel numbering, any sentence about an L-structure with a natural number.
- To measure the complexity of a structure, we identify it to its atomic diagram: we say that a structure $\mathcal{M}$ is $\mathbf{d}$-*computable* if $D(\mathcal{M})$ is a $\mathbf{d}$-computable subset of ω, where $\mathbf{d}$ is a Turing degree.

Computable structures

- We consider finite (relational) signatures L.
- We assume that the domain of any countably infinite structure is equal to the set ω of the natural numbers.
- We effectively identify, through a fixed Gödel numbering, any sentence about an L-structure with a natural number.
- To measure the complexity of a structure, we identify it to its atomic diagram: we say that a structure $\mathcal{M}$ is $\mathbf{d}$-*computable* if $D(\mathcal{M})$ is a $\mathbf{d}$-computable subset of ω, where $\mathbf{d}$ is a Turing degree.
- A *presentation* of a countable algebraic structure is an arbitrary isomorphic copy $\mathcal{M}' \cong \mathcal{M}$ with the universe a subset of ω. We call a structure $\mathcal{M}$ *computably presentable* if it has a presentation $\mathcal{M}'$ which is computable.

Computable structures

- Any computable structure $\mathcal{A}$ in a relational signature can be presented as an increasing union of its finite substructures

$$\mathcal{A}^0 \subseteq \mathcal{A}^1 \subseteq \ldots \subseteq \mathcal{A}^i \subseteq \ldots,$$

where $\mathcal{A}^n$ is the restriction of $\mathcal{A}$ to the domain $\{0, 1, \ldots, n\}$ and $\mathcal{A} = \bigcup_i \mathcal{A}^i$.

Computable structures

- Any computable structure $\mathcal{A}$ in a relational signature can be presented as an increasing union of its finite substructures

$$\mathcal{A}^0 \subseteq \mathcal{A}^1 \subseteq \ldots \subseteq \mathcal{A}^i \subseteq \ldots,$$

  where $\mathcal{A}^n$ is the restriction of $\mathcal{A}$ to the domain $\{0, 1, \ldots, n\}$ and $\mathcal{A} = \bigcup_i \mathcal{A}^i$.
- By $\mathbb{K}_L$ we denote the class of all L-structures with domain ω.

Computable structures

- Any computable structure $\mathcal{A}$ in a relational signature can be presented as an increasing union of its finite substructures

$$\mathcal{A}^0 \subseteq \mathcal{A}^1 \subseteq \ldots \subseteq \mathcal{A}^i \subseteq \ldots,$$

  where $\mathcal{A}^n$ is the restriction of $\mathcal{A}$ to the domain $\{0, 1, \ldots, n\}$ and $\mathcal{A} = \bigcup_i \mathcal{A}^i$.
- By $\mathbb{K}_L$ we denote the class of all L-structures with domain ω.
- We assume that every considered class of L-structures is closed under isomorphisms.

Formal Example I

Consider a family $\mathfrak{C}$ of two infinite, undirected graphs:

1. G_1 which contains only 2-cycles, and
2. G_2 containing only 3-cycles.

Formal Example I

Consider a family $\mathfrak{C}$ of two infinite, undirected graphs:

1. G_1 which contains only 2-cycles, and
2. G_2 containing only 3-cycles.

- **The learning domain**: the family $\mathfrak{C}^*$ of all possible presentations of G_1 and G_2.

Formal Example I

Consider a family $\mathfrak{C}$ of two infinite, undirected graphs:

1. G_1 which contains only 2-cycles, and
2. G_2 containing only 3-cycles.

- **The learning domain**: the family $\mathfrak{C}^*$ of all possible presentations of G_1 and G_2.
- **The hypothesis space**: the set $\{1, 2, ?\}$.

Formal Example I

Consider a family $\mathfrak{C}$ of two infinite, undirected graphs:

1. G_1 which contains only 2-cycles, and
2. G_2 containing only 3-cycles.

- **The learning domain**: the family $\mathfrak{C}^*$ of all possible presentations of G_1 and G_2.
- **The hypothesis space**: the set $\{1, 2, ?\}$.
- **The information source**: an informant I for a graph H in $\mathfrak{C}^*$ is an infinite list of pairs containing: all pairs (x, y) of natural numbers, as the first component; and either 0 or 1, as the second component, where this second component is 1 if and only if x and y are adjacent in H.

Formal Example I

Consider a family $\mathfrak{C}$ of two infinite, undirected graphs:

1. G_1 which contains only 2-cycles, and
2. G_2 containing only 3-cycles.

- **The learning domain**: the family $\mathfrak{C}^*$ of all possible presentations of G_1 and G_2.
- **The hypothesis space**: the set $\{1, 2, ?\}$.
- **The information source**: an informant I for a graph H in $\mathfrak{C}^*$ is an infinite list of pairs containing: all pairs (x, y) of natural numbers, as the first component; and either 0 or 1, as the second component, where this second component is 1 if and only if x and y are adjacent in H.
- **The learner**: a function (or an algorithm).

Formal Example I

Consider a family $\mathfrak{C}$ of two infinite, undirected graphs:

1. G_1 which contains only 2-cycles, and
2. G_2 containing only 3-cycles.

- **The learning domain**: the family $\mathfrak{C}^*$ of all possible presentations of G_1 and G_2.
- **The hypothesis space**: the set $\{1, 2, ?\}$.
- **The information source**: an informant I for a graph H in $\mathfrak{C}^*$ is an infinite list of pairs containing: all pairs (x, y) of natural numbers, as the first component; and either 0 or 1, as the second component, where this second component is 1 if and only if x and y are adjacent in H.
- **The learner**: a function (or an algorithm).
- **The prior knowledge**: the target graph is isomorphic to either G_1 or G_2.

Formal Example I

Consider a family $\mathfrak{C}$ of two infinite, undirected graphs:

1. G_1 which contains only 2-cycles, and
2. G_2 containing only 3-cycles.

- **The learning domain**: the family $\mathfrak{C}^*$ of all possible presentations of G_1 and G_2.
- **The hypothesis space**: the set $\{1, 2, ?\}$.
- **The information source**: an informant I for a graph H in $\mathfrak{C}^*$ is an infinite list of pairs containing: all pairs (x, y) of natural numbers, as the first component; and either 0 or 1, as the second component, where this second component is 1 if and only if x and y are adjacent in H.
- **The learner**: a function (or an algorithm).
- **The prior knowledge**: the target graph is isomorphic to either G_1 or G_2.
- **The criterion of success**: a learner that, receiving larger and larger pieces of any graph G in $\mathfrak{C}^*$, eventually stabilizes to a correct guess about whether G is isomorphic to G_1 or G_2.

Formal Example II

Consider an *infinite* family $\mathfrak{D}$:

- for each $i \geq 1$, the graph G_i contains infinitely many $(i+1)$-cycles.

Formal Example II

Consider an *infinite* family $\mathfrak{D}$:

- for each $i \geq 1$, the graph G_i contains infinitely many $(i+1)$-cycles.
- the learning domain is the family $\mathfrak{D}^*$ of *all* presentations of the graphs in $\mathfrak{D}$;

Formal Example II

Consider an *infinite* family $\mathfrak{D}$:

- for each $i \geq 1$, the graph G_i contains infinitely many $(i+1)$-cycles.
- the learning domain is the family $\mathfrak{D}^*$ of *all* presentations of the graphs in $\mathfrak{D}$;
- each informant I provides both positive and negative information about any given graph in $\mathfrak{D}^*$;

Formal Example II

Consider an *infinite* family $\mathfrak{D}$:

- for each $i \geq 1$, the graph G_i contains infinitely many $(i+1)$-cycles.
- the learning domain is the family $\mathfrak{D}^*$ of *all* presentations of the graphs in $\mathfrak{D}$;
- each informant I provides both positive and negative information about any given graph in $\mathfrak{D}^*$;
- every conjecture is an element of the set $\omega \cup \{?\}$;

Formal Example II

Consider an *infinite* family $\mathfrak{D}$:

- for each $i \geq 1$, the graph G_i contains infinitely many $(i+1)$-cycles.
- the learning domain is the family $\mathfrak{D}^*$ of *all* presentations of the graphs in $\mathfrak{D}$;
- each informant I provides both positive and negative information about any given graph in $\mathfrak{D}^*$;
- every conjecture is an element of the set $\omega \cup \{?\}$;
- a learner is a function (or an algorithm) that learns, up to isomorphism, any graph in $\mathfrak{D}^*$;

Formal Example II

Consider an *infinite* family $\mathfrak{D}$:

- for each $i \geq 1$, the graph G_i contains infinitely many $(i+1)$-cycles.
- the learning domain is the family $\mathfrak{D}^*$ of *all* presentations of the graphs in $\mathfrak{D}$;
- each informant I provides both positive and negative information about any given graph in $\mathfrak{D}^*$;
- every conjecture is an element of the set $\omega \cup \{?\}$;
- a learner is a function (or an algorithm) that learns, up to isomorphism, any graph in $\mathfrak{D}^*$;
- the prior knowledge consists of the knowledge that the target graph is isomorphic to some graph from the family $\mathfrak{D}$.

Formal Example II

Consider an *infinite* family $\mathfrak{D}$:

- for each $i \geq 1$, the graph G_i contains infinitely many $(i+1)$-cycles.
- the learning domain is the family $\mathfrak{D}^*$ of *all* presentations of the graphs in $\mathfrak{D}$;
- each informant I provides both positive and negative information about any given graph in $\mathfrak{D}^*$;
- every conjecture is an element of the set $\omega \cup \{?\}$;
- a learner is a function (or an algorithm) that learns, up to isomorphism, any graph in $\mathfrak{D}^*$;
- the prior knowledge consists of the knowledge that the target graph is isomorphic to some graph from the family $\mathfrak{D}$.

The only technical problem is how to specify the hypothesis space or, in other words:

How does one formally define the set of possible conjectures?

Enumerations I

First Solution:
For $m \in \omega$, the conjecture "m" means "$H \cong G_{m+1}$."

Enumerations I

First Solution:
For $m \in \omega$, the conjecture "m" means "$H \cong G_{m+1}$."

- This solution is similar to the so-called *exact learning*, considered in the setting of c.e. languages, where one assumes that the hypothesis space of the problem is precisely the class being learned with the corresponding indexing.

Enumerations I

First Solution:
For $m \in \omega$, the conjecture "m" means "$H \cong G_{m+1}$."

- This solution is similar to the so-called *exact learning*, considered in the setting of c.e. languages, where one assumes that the hypothesis space of the problem is precisely the class being learned with the corresponding indexing.
- Drawback: it can be computationally very hard to enumerate certain familiar families of computable structures, up to isomorphism.
- Goncharov and Knight: for the classes of computable Boolean algebras, linear orders, and abelian p-groups one cannot even hyperarithmetically enumerate their isomorphism types.

Enumerations II

Second Solution:
Fix a uniformly computable sequence $(\mathcal{M}_e)_{e\in\omega}$ of *all* computable undirected graphs. The conjecture "m" means "$H \cong \mathcal{M}_m$."

Enumerations II

Second Solution:
Fix a uniformly computable sequence $(\mathcal{M}_e)_{e\in\omega}$ of *all* computable undirected graphs. The conjecture "m" means "$H \cong \mathcal{M}_m$."
This solution is similar to the so-called *class-comprising learning*, where one assumes that the hypothesis space of the problem should only contain the class being learned.

Enumerations II

Second Solution:
Fix a uniformly computable sequence $(\mathcal{M}_e)_{e\in\omega}$ of *all* computable undirected graphs. The conjecture "m" means "$H \cong \mathcal{M}_m$."
This solution is similar to the so-called *class-comprising learning*, where one assumes that the hypothesis space of the problem should only contain the class being learned.

General framework: consider an arbitrary superclass $\mathfrak{K} \supseteq \mathfrak{D}$ which is *uniformly enumerable*, i.e. there is a uniformly computable sequence of structures $(\mathcal{N}_e)_{e\in\omega}$ such that:

1. Any structure from $\mathfrak{K}$ is isomorphic to some $\mathcal{N}_e$.
2. For every e, $\mathcal{N}_e$ belongs to $\mathfrak{K}$.

Then for a number $e \in \omega$, the conjecture "e" is interpreted as "the input structure is isomorphic to $\mathcal{N}_e$."

Our framework

Let L be a relational signature.

- Let $\mathfrak{K}_0$ be a class of L-structures. An *effective enumeration* of the class $\mathfrak{K}_0$ is a function $\nu\colon \omega \to \mathfrak{K}_0$ with the following properties:

Our framework

Let L be a relational signature.

- Let $\mathfrak{K}_0$ be a class of L-structures. An *effective enumeration* of the class $\mathfrak{K}_0$ is a function $\nu\colon \omega \to \mathfrak{K}_0$ with the following properties:
  1. The sequence of L-structures $(\nu(e))_{e\in\omega}$ is uniformly computable.

Our framework

Let L be a relational signature.

- Let $\mathfrak{K}_0$ be a class of L-structures. An *effective enumeration* of the class $\mathfrak{K}_0$ is a function $\nu\colon \omega \to \mathfrak{K}_0$ with the following properties:
  1. The sequence of L-structures $(\nu(e))_{e\in\omega}$ is uniformly computable.
  2. For any $\mathcal{A} \in \mathfrak{K}_0$, there is an index e such that the structures $\mathcal{A}$ and $\nu(e)$ are isomorphic.
- Let $\mathfrak{K}_0$ be a class of L-structures, and let ν be an effective enumeration of $\mathfrak{K}_0$. Suppose that $\mathfrak{K}$ is a subclass of $\mathfrak{K}_0$.

Our framework

Let L be a relational signature.

- Let $\mathfrak{K}_0$ be a class of L-structures. An *effective enumeration* of the class $\mathfrak{K}_0$ is a function $\nu\colon \omega \to \mathfrak{K}_0$ with the following properties:
  1. The sequence of L-structures $(\nu(e))_{e\in\omega}$ is uniformly computable.
  2. For any $\mathcal{A} \in \mathfrak{K}_0$, there is an index e such that the structures $\mathcal{A}$ and $\nu(e)$ are isomorphic.
- Let $\mathfrak{K}_0$ be a class of L-structures, and let ν be an effective enumeration of $\mathfrak{K}_0$. Suppose that $\mathfrak{K}$ is a subclass of $\mathfrak{K}_0$.

  We say that $\mathfrak{K}$ is $\mathbf{InfEx}_{\cong}[\nu]$-*learnable* if there is a learner M with the following property:

Our framework

Let L be a relational signature.

- Let $\mathfrak{K}_0$ be a class of L-structures. An *effective enumeration* of the class $\mathfrak{K}_0$ is a function $\nu\colon \omega \to \mathfrak{K}_0$ with the following properties:
  1. The sequence of L-structures $(\nu(e))_{e\in\omega}$ is uniformly computable.
  2. For any $\mathcal{A} \in \mathfrak{K}_0$, there is an index e such that the structures $\mathcal{A}$ and $\nu(e)$ are isomorphic.
- Let $\mathfrak{K}_0$ be a class of L-structures, and let ν be an effective enumeration of $\mathfrak{K}_0$. Suppose that $\mathfrak{K}$ is a subclass of $\mathfrak{K}_0$.

  We say that $\mathfrak{K}$ is $\mathbf{InfEx}_{\cong}[\nu]$-*learnable* if there is a learner M with the following property:
  If I is an informant for a structure $\mathcal{A} \in \mathfrak{K}$, then there are e and s_0 such that $\nu(e) \cong \mathcal{A}$ and $M(I[s]) = e$ for all $s \geq s_0$. In other words, in the limit, the learner M learns all isomorphism types from $\mathfrak{K}$.

Infinitary formulas

We are able to fully characterize which families of structures are learnable. To do so, we use the logic $\mathcal{L}_{\omega_1\omega}$, which allows to take conjunctions or disjunctions of infinite sets of formulas.
The class of infinitary Σ_α L-formulas:

Infinitary formulas

We are able to fully characterize which families of structures are learnable. To do so, we use the logic $\mathcal{L}_{\omega_1\omega}$, which allows to take conjunctions or disjunctions of infinite sets of formulas.

The class of infinitary Σ_α L-formulas:

(a) $\Sigma_0^{\inf}$ and $\Pi_0^{\inf}$ formulas are quantifier-free first-order L-formulas.

Infinitary formulas

We are able to fully characterize which families of structures are learnable. To do so, we use the logic $\mathcal{L}_{\omega_1\omega}$, which allows to take conjunctions or disjunctions of infinite sets of formulas.

The class of infinitary Σ_α L-formulas:

(a) $\Sigma_0^{\inf}$ and $\Pi_0^{\inf}$ formulas are quantifier-free first-order L-formulas.

(b) A $\Sigma_\alpha^{\inf}$ formula $\psi(x_0, \ldots, x_m)$ is an countable disjunction

$$\bigvee\!\!\!\bigvee_{i \in I} \exists \bar{y}_i \xi_i(\bar{x}, \bar{y}_i),$$

where each ξ_i is a $\Pi_{\beta_i}^{\inf}$ formula, for some $\beta_i < \alpha$.

Infinitary formulas

We are able to fully characterize which families of structures are learnable. To do so, we use the logic $\mathcal{L}_{\omega_1\omega}$, which allows to take conjunctions or disjunctions of infinite sets of formulas.

The class of infinitary Σ_α L-formulas:

(a) $\Sigma_0^{\inf}$ and $\Pi_0^{\inf}$ formulas are quantifier-free first-order L-formulas.

(b) A $\Sigma_\alpha^{\inf}$ formula $\psi(x_0, \ldots, x_m)$ is an countable disjunction

$$\bigvee\!\!\!\bigvee_{i\in I} \exists \bar{y}_i \xi_i(\bar{x}, \bar{y}_i),$$

where each ξ_i is a $\Pi_{\beta_i}^{\inf}$ formula, for some $\beta_i < \alpha$.

(c) A $\Pi_\alpha^{\inf}$ formula $\psi(\bar{x})$ is a countable conjunction

$$\bigwedge\!\!\!\bigwedge_{i\in I} \forall \bar{y}_i \xi_i(\bar{x}, \bar{y}_i),$$

where each ξ_i is a $\Sigma_{\beta_i}^{\inf}$ formula, for some $\beta_i < \alpha$.

Today we will only need $\Sigma_\alpha^{\inf}$ formulas for $\alpha \leq 2$.

Infinitary formulas

We are able to fully characterize which families of structures are learnable. To do so, we use the logic $\mathcal{L}_{\omega_1\omega}$, which allows to take conjunctions or disjunctions of infinite sets of formulas.

The class of X-computable infinitary Σ_α L-formulas:

(a) $\Sigma^c_0(X)$ and $\Pi^c_0(X)$ formulas are quantifier-free first-order L-formulas.

(b) A $\Sigma^c_\alpha(X)$ formula $\psi(x_0, \dots, x_m)$ is an X-computably enumerable (X-c.e.) disjunction

$$\bigvee\!\!\!\!\bigvee_{i \in I} \exists \bar{y}_i \xi_i(\bar{x}, \bar{y}_i),$$

where each ξ_i is a $\Pi^c_{\beta_i}(X)$ formula, for some $\beta_i < \alpha$.

(c) A $\Pi^c_\alpha(X)$ formula $\psi(\bar{x})$ is an X-c.e. conjunction

$$\bigwedge\!\!\!\!\bigwedge_{i \in I} \forall \bar{y}_i \xi_i(\bar{x}, \bar{y}_i),$$

where each ξ_i is a $\Sigma^c_{\beta_i}(X)$ formula, for some $\beta_i < \alpha$.

Today we will only need $\Sigma^c_\alpha(X)$ formulas for $\alpha \leq 2$.

Main Theorem

Suppose that $\mathfrak{K}_0$ is a class of L-structures, and ν is an effective enumeration of the class $\mathfrak{K}_0$.

Theorem

Let $\mathfrak{K} = \{\mathcal{B}_i : i \in \omega\}$ be a family of structures such that $\mathfrak{K} \subseteq \mathfrak{K}_0$, and the structures $\mathcal{B}_i$ are infinite and pairwise non-isomorphic. Then the following conditions are equivalent:

Main Theorem

Suppose that $\mathfrak{K}_0$ is a class of L-structures, and ν is an effective enumeration of the class $\mathfrak{K}_0$.

Theorem

Let $\mathfrak{K} = \{\mathcal{B}_i : i \in \omega\}$ be a family of structures such that $\mathfrak{K} \subseteq \mathfrak{K}_0$, and the structures $\mathcal{B}_i$ are infinite and pairwise non-isomorphic. Then the following conditions are equivalent:

1. *The class $\mathfrak{K}$ is $\mathbf{InfEx}_{\cong}[\nu]$-learnable;*
2. *There is a sequence of $\Sigma_2^{\inf}$ sentences $\{\psi_i : i \in \omega\}$ such that for all i and j, we have $\mathcal{B}_j \models \psi_i$ if and only if $i = j$.*

Main Theorem

Suppose that $\mathfrak{K}_0$ is a class of L-structures, and ν is an effective enumeration of the class $\mathfrak{K}_0$.

Theorem

Let $\mathfrak{K} = \{\mathcal{B}_i : i \in \omega\}$ be a family of structures such that $\mathfrak{K} \subseteq \mathfrak{K}_0$, and the structures $\mathcal{B}_i$ are infinite and pairwise non-isomorphic. Then the following conditions are equivalent:

1. *The class $\mathfrak{K}$ is* $\mathbf{InfEx}_{\cong}[\nu]$*-learnable;*
2. *There is a sequence of Σ_2^{inf} sentences $\{\psi_i : i \in \omega\}$ such that for all i and j, we have $\mathcal{B}_j \models \psi_i$ if and only if $i = j$.*

The statement is similar to a result due to Martin and Osherson. Yet, our proof is novel and based on Pullback Theorem in the context of Turing computable embeddings introduced by Knight, S. Miller, and Vanden Boom. This provides us with an upper bound for the Turing complexity of the learners which we apply later.

Bounding the complexity

Corollary

Let $X \subseteq \omega$ be an oracle. Let $\mathfrak{K}_0$ be a class of countably infinite L-structures, and ν be an effective enumeration of $\mathfrak{K}_0$.

Bounding the complexity

Corollary

Let $X \subseteq \omega$ be an oracle. Let $\mathfrak{K}_0$ be a class of countably infinite L-structures, and ν be an effective enumeration of $\mathfrak{K}_0$. Assume that either $I = \omega$, or I is a finite initial segment of ω. Consider a subclass $\mathfrak{K} = \{\mathcal{B}_i : i \in I\}$ inside $\mathfrak{K}_0$. Suppose that:

Bounding the complexity

Corollary

Let $X \subseteq \omega$ be an oracle. Let $\mathfrak{K}_0$ be a class of countably infinite L-structures, and ν be an effective enumeration of $\mathfrak{K}_0$. Assume that either $I = \omega$, or I is a finite initial segment of ω. Consider a subclass $\mathfrak{K} = \{\mathcal{B}_i : i \in I\}$ inside $\mathfrak{K}_0$. Suppose that:

(i) *There is uniformly X-computable sequence of $\Sigma^c_2(X)$ sentences $(\psi_i)_{i \in I}$ such that:*

$$\mathcal{B}_j \models \psi_i \;\Leftrightarrow\; i = j.$$

Bounding the complexity

Corollary

Let $X \subseteq \omega$ be an oracle. Let $\mathfrak{K}_0$ be a class of countably infinite L-structures, and ν be an effective enumeration of $\mathfrak{K}_0$. Assume that either $I = \omega$, or I is a finite initial segment of ω. Consider a subclass $\mathfrak{K} = \{\mathcal{B}_i : i \in I\}$ inside $\mathfrak{K}_0$. Suppose that:

(i) *There is uniformly X-computable sequence of $\Sigma^c_2(X)$ sentences $(\psi_i)_{i \in I}$ such that:*

$$\mathcal{B}_j \models \psi_i \;\Leftrightarrow\; i = j.$$

(ii) *There is an X-computable sequence $(e_i)_{i \in I}$ such that $\nu(e_i) \cong \mathcal{B}_i$ for all i. Note that if the set I is finite, then one can always choose this sequence in a computable way.*

Bounding the complexity

Corollary

Let $X \subseteq \omega$ be an oracle. Let $\mathfrak{K}_0$ be a class of countably infinite L-structures, and ν be an effective enumeration of $\mathfrak{K}_0$. Assume that either $I = \omega$, or I is a finite initial segment of ω. Consider a subclass $\mathfrak{K} = \{\mathcal{B}_i : i \in I\}$ inside $\mathfrak{K}_0$. Suppose that:

(i) *There is uniformly X-computable sequence of $\Sigma^c_2(X)$ sentences $(\psi_i)_{i \in I}$ such that:*

$$\mathcal{B}_j \models \psi_i \;\Leftrightarrow\; i = j.$$

(ii) *There is an X-computable sequence $(e_i)_{i \in I}$ such that $\nu(e_i) \cong \mathcal{B}_i$ for all i. Note that if the set I is finite, then one can always choose this sequence in a computable way.*

Then the class $\mathfrak{K}$ is $\mathbf{InfEx}_{\cong}[\nu]$-learnable via an X-computable learner.

Application I: distributive lattices

Let $L_{\mathrm{lat}} := \{\vee, \wedge\}$.

Application I: distributive lattices

Let $L_{\mathrm{lat}} := \{\vee, \wedge\}$. Selivanov built a uniformly computable family $\{\mathcal{D}_i : i \in \omega\}$ of finite distributive lattices with the following property:

Application I: distributive lattices

Let $L_{\mathrm{lat}} := \{\vee, \wedge\}$. Selivanov built a uniformly computable family $\{\mathcal{D}_i : i \in \omega\}$ of finite distributive lattices with the following property: If $i \neq j$, then there is no isomorphic embedding from $\mathcal{D}_i$ into $\mathcal{D}_j$.

Application I: distributive lattices

Let $L_{\mathrm{lat}} := \{\vee, \wedge\}$. Selivanov built a uniformly computable family $\{\mathcal{D}_i : i \in \omega\}$ of finite distributive lattices with the following property:

If $i \neq j$, then there is no isomorphic embedding from $\mathcal{D}_i$ into $\mathcal{D}_j$.

For $i \in \omega$, define a countably infinite poset $\mathcal{B}_i$ as a direct sum of the lattice $\mathcal{D}_i$ and the linear order ω.

Application I: distributive lattices

Let $L_{\mathrm{lat}} := \{\vee, \wedge\}$. Selivanov built a uniformly computable family $\{\mathcal{D}_i : i \in \omega\}$ of finite distributive lattices with the following property:

If $i \neq j$, then there is no isomorphic embedding from $\mathcal{D}_i$ into $\mathcal{D}_j$.

For $i \in \omega$, define a countably infinite poset $\mathcal{B}_i$ as a direct sum of the lattice $\mathcal{D}_i$ and the linear order ω. Let $\mathfrak{K}_{lat} = \{\mathcal{B}_i : i \in \omega\}$. One can build a Friedberg effective enumeration ν_{lat} of $\mathfrak{K}_{lat}$.

Application I: distributive lattices

Let $L_{\mathrm{lat}} := \{\vee, \wedge\}$. Selivanov built a uniformly computable family $\{\mathcal{D}_i : i \in \omega\}$ of finite distributive lattices with the following property:

If $i \neq j$, then there is no isomorphic embedding from $\mathcal{D}_i$ into $\mathcal{D}_j$.

For $i \in \omega$, define a countably infinite poset $\mathcal{B}_i$ as a direct sum of the lattice $\mathcal{D}_i$ and the linear order ω. Let $\mathfrak{K}_{lat} = \{\mathcal{B}_i : i \in \omega\}$. One can build a Friedberg effective enumeration ν_{lat} of $\mathfrak{K}_{lat}$.

Proposition

The class $\mathfrak{K}_{lat}$ is $\mathbf{InfEx}_{\cong}[\nu_{lat}]$-learnable by a computable learner.

Application I: distributive lattices

Let $L_{\mathrm{lat}} := \{\vee, \wedge\}$. Selivanov built a uniformly computable family $\{\mathcal{D}_i : i \in \omega\}$ of finite distributive lattices with the following property:

If $i \neq j$, then there is no isomorphic embedding from $\mathcal{D}_i$ into $\mathcal{D}_j$.

For $i \in \omega$, define a countably infinite poset $\mathcal{B}_i$ as a direct sum of the lattice $\mathcal{D}_i$ and the linear order ω. Let $\mathfrak{K}_{lat} = \{\mathcal{B}_i : i \in \omega\}$. One can build a Friedberg effective enumeration ν_{lat} of $\mathfrak{K}_{lat}$.

Proposition

The class $\mathfrak{K}_{lat}$ is $\mathbf{InfEx}_{\cong}[\nu_{lat}]$-learnable by a computable learner.

Corollary

Suppose that ν is an arbitrary effective enumeration of the class $\mathbb{K}_{L_{\mathrm{lat}}}$. Then the following holds:

Application I: distributive lattices

Let $L_{\mathrm{lat}} := \{\vee, \wedge\}$. Selivanov built a uniformly computable family $\{\mathcal{D}_i : i \in \omega\}$ of finite distributive lattices with the following property:

If $i \neq j$, then there is no isomorphic embedding from $\mathcal{D}_i$ into $\mathcal{D}_j$.

For $i \in \omega$, define a countably infinite poset $\mathcal{B}_i$ as a direct sum of the lattice $\mathcal{D}_i$ and the linear order ω. Let $\mathfrak{K}_{lat} = \{\mathcal{B}_i : i \in \omega\}$. One can build a Friedberg effective enumeration ν_{lat} of $\mathfrak{K}_{lat}$.

Proposition

The class $\mathfrak{K}_{lat}$ is $\mathbf{InfEx}_{\cong}[\nu_{lat}]$-learnable by a computable learner.

Corollary

Suppose that ν is an arbitrary effective enumeration of the class $\mathbb{K}_{L_{\mathrm{lat}}}$. Then the following holds:

(a) *The class $\mathfrak{K}_{lat}$ is $\mathbf{InfEx}_{\cong}[\nu]$-learnable. Note that here the complexity of the learner depends only on the complexity of the sequence $(e_i)_{i \in \omega}$ from Corollary.*

(b) *$\mathfrak{K}_{lat}$ is $\mathbf{InfEx}_{\cong}[\nu \oplus \nu_{lat}]$-learnable by a computable learner.*

Application II: Abelian groups

Let $L_{ag} := \{+, 0\}$.

Application II: Abelian groups

Let $L_{ag} := \{+, 0\}$. For a number $i \in \omega$, define the group

$$\mathcal{A}_i := \bigoplus_{j \in \omega} \mathbb{Z}(p^{i+1}).$$

Application II: Abelian groups

Let $L_{ag} := \{+, 0\}$. For a number $i \in \omega$, define the group

$$\mathcal{A}_i := \bigoplus_{j \in \omega} \mathbb{Z}(p^{i+1}).$$

We set $\mathfrak{K}_{ag} := \{\mathcal{A}_i : i \in \omega\}$, and we construct a Friedberg effective enumeration ν_{ag} as follows: just define $\nu_{ag}(i)$ as a natural computable copy of $\mathcal{A}_i$.

Application II: Abelian groups

Let $L_{ag} := \{+, 0\}$. For a number $i \in \omega$, define the group

$$\mathcal{A}_i := \bigoplus_{j \in \omega} \mathbb{Z}(p^{i+1}).$$

We set $\mathfrak{K}_{ag} := \{\mathcal{A}_i : i \in \omega\}$, and we construct a Friedberg effective enumeration ν_{ag} as follows: just define $\nu_{ag}(i)$ as a natural computable copy of $\mathcal{A}_i$.

Proposition

The class $\mathfrak{K}_{ag}$ is $\mathbf{InfEx}_{\cong}[\nu_{ag}]$-learnable by a computable learner.

Application II: Abelian groups

Let $L_{ag} := \{+, 0\}$. For a number $i \in \omega$, define the group

$$\mathcal{A}_i := \bigoplus_{j \in \omega} \mathbb{Z}(p^{i+1}).$$

We set $\mathfrak{K}_{ag} := \{\mathcal{A}_i : i \in \omega\}$, and we construct a Friedberg effective enumeration ν_{ag} as follows: just define $\nu_{ag}(i)$ as a natural computable copy of $\mathcal{A}_i$.

Proposition

The class $\mathfrak{K}_{ag}$ is $\mathbf{InfEx}_{\cong}[\nu_{ag}]$-learnable by a computable learner.

Corollary

Suppose that ν is an arbitrary effective enumeration of the class $\mathbb{K}_{L_{ag}}$. Then the following holds:

Application II: Abelian groups

Let $L_{ag} := \{+, 0\}$. For a number $i \in \omega$, define the group

$$\mathcal{A}_i := \bigoplus_{j \in \omega} \mathbb{Z}(p^{i+1}).$$

We set $\mathfrak{K}_{ag} := \{\mathcal{A}_i : i \in \omega\}$, and we construct a Friedberg effective enumeration ν_{ag} as follows: just define $\nu_{ag}(i)$ as a natural computable copy of $\mathcal{A}_i$.

Proposition

The class $\mathfrak{K}_{ag}$ is $\mathbf{InfEx}_{\cong}[\nu_{ag}]$-learnable by a computable learner.

Corollary

Suppose that ν is an arbitrary effective enumeration of the class $\mathbb{K}_{L_{ag}}$. Then the following holds:

(a) *The class $\mathfrak{K}_{ag}$ is $\mathbf{InfEx}_{\cong}[\nu]$-learnable.*

(b) *$\mathfrak{K}_{ag}$ is $\mathbf{InfEx}_{\cong}[\nu \oplus \nu_{ag}]$-learnable by a computable learner.*

Application III: Boolean algebras

Proposition

Let $\mathfrak{K}$ be some class of infinite Boolean algebras, and let ν be an effective enumeration of $\mathfrak{K}$. Suppose that $\mathfrak{C}$ is a subclass of $\mathfrak{K}$ such that $\mathfrak{C}$ contains at least two non-isomorphic members. Then the class $\mathfrak{C}$ is not $\mathbf{InfEx}_{\cong}[\nu]$*-learnable.*

Application III: Boolean algebras

Proposition

Let $\mathfrak{K}$ be some class of infinite Boolean algebras, and let ν be an effective enumeration of $\mathfrak{K}$. Suppose that $\mathfrak{C}$ is a subclass of $\mathfrak{K}$ such that $\mathfrak{C}$ contains at least two non-isomorphic members. Then the class $\mathfrak{C}$ is not $\mathbf{InfEx}_{\cong}[\nu]$-learnable.

Idea of Proof: Suppose $\mathcal{A} \ncong \mathcal{B}$ are two BAs from $\mathfrak{C}$.

Application III: Boolean algebras

Proposition

Let $\mathfrak{K}$ be some class of infinite Boolean algebras, and let ν be an effective enumeration of $\mathfrak{K}$. Suppose that $\mathfrak{C}$ is a subclass of $\mathfrak{K}$ such that $\mathfrak{C}$ contains at least two non-isomorphic members. Then the class $\mathfrak{C}$ is not $\mathbf{InfEx}_{\cong}[\nu]$*-learnable.*

Idea of Proof: Suppose $\mathcal{A} \not\cong \mathcal{B}$ are two BAs from $\mathfrak{C}$. Then

$$\Sigma_2^{\inf}\text{-}Th(\mathcal{A}) \subseteq \Sigma_2^{\inf}\text{-}Th(\mathcal{B}) \text{ or } \Sigma_2^{\inf}\text{-}Th(\mathcal{B}) \subseteq \Sigma_2^{\inf}\text{-}Th(\mathcal{A}).$$

Therefore, the class $\mathfrak{C}$ is not $\mathbf{InfEx}_{\cong}[\nu]$-learnable.

Application IV: linear orders

Proposition

Let $n \geq 2$ be a natural number. Then there is a class of computable infinite linear orders $\mathfrak{C}$ with the following properties:

(a) *$\mathfrak{C}$ contains precisely n isomorphism types.*

(b) *Suppose that $\mathfrak{K}$ is a superclass of $\mathfrak{C}$, and ν is an effective enumeration of $\mathfrak{K}$. Then the class $\mathfrak{C}$ is $\mathbf{InfEx}_{\cong}[\nu]$-learnable.*

Application IV: linear orders

Proposition

Let $n \geq 2$ be a natural number. Then there is a class of computable infinite linear orders $\mathfrak{C}$ with the following properties:

(a) *$\mathfrak{C}$ contains precisely n isomorphism types.*

(b) *Suppose that $\mathfrak{K}$ is a superclass of $\mathfrak{C}$, and ν is an effective enumeration of $\mathfrak{K}$. Then the class $\mathfrak{C}$ is $\mathbf{InfEx}_{\cong}[\nu]$-learnable.*

Theorem

Let $\mathfrak{K}$ be some class of infinite linear orders, and let ν be an effective enumeration of $\mathfrak{K}$. Suppose that $\mathfrak{C}$ is a subclass of $\mathfrak{K}$ such that $\mathfrak{C}$ contains infinitely many pairwise non-isomorphic members. Then the class $\mathfrak{C}$ is not $\mathbf{InfEx}_{\cong}[\nu]$-learnable.

Equivalence structures

Let $\mathbb{E}$ be the class of equivalence structures, and ν its enumeration.

Equivalence structures

Let $\mathbb{E}$ be the class of equivalence structures, and ν its enumeration.

Theorem (F., Kötzing, San Mauro, 2018)

If $\mathfrak{A}$ is a family of equivalence structures, then

$$\mathfrak{A} \textit{ is } \mathbf{InfEx}_{\cong}[\nu]\textit{-learnable } \Leftrightarrow \mathfrak{A} \textit{ is finitely separable.}$$

Equivalence structures

Let $\mathbb{E}$ be the class of equivalence structures, and ν its enumeration.

Theorem (F., Kötzing, San Mauro, 2018)

If $\mathfrak{A}$ is a family of equivalence structures, then

$$\mathfrak{A} \text{ is } \mathbf{InfEx}_{\cong}[\nu]\text{-learnable} \Leftrightarrow \mathfrak{A} \text{ is finitely separable.}$$

Corollary

1. *If $\mathfrak{A} \notin \mathbf{InfEx}_{\cong}[\nu]$ and $\mathfrak{A}/_{\cong}$ is finite, then there exists $(\mathcal{A}, B) \subseteq \mathfrak{A}$ such that $(\mathcal{A}, B) \notin \mathbf{InfEx}_{\cong}[\nu]$.*

Equivalence structures

Let $\mathbb{E}$ be the class of equivalence structures, and ν its enumeration.

Theorem (F., Kötzing, San Mauro, 2018)

If $\mathfrak{A}$ is a family of equivalence structures, then

$$\mathfrak{A} \textit{ is } \mathbf{InfEx}_{\cong}[\nu]\textit{-learnable } \Leftrightarrow \mathfrak{A} \textit{ is finitely separable.}$$

Corollary

1. *If $\mathfrak{A} \notin \mathbf{InfEx}_{\cong}[\nu]$ and $\mathfrak{A}/_{\cong}$ is finite, then there exists $(\mathcal{A}, B) \subseteq \mathfrak{A}$ such that $(\mathcal{A}, B) \notin \mathbf{InfEx}_{\cong}[\nu]$.*
2. *There is $\mathfrak{A} \notin \mathbf{InfEx}_{\cong}[\nu])$ such that, for all $\mathfrak{B} \subseteq \mathfrak{A}$, if $\mathfrak{B}/_{\cong}$ is finite, then $\mathfrak{B} \in \mathbf{InfEx}_{\cong}[\nu]$.*

Equivalence structures

Let $\mathbb{E}$ be the class of equivalence structures, and ν its enumeration.

Theorem (F., Kötzing, San Mauro, 2018)

If $\mathfrak{A}$ is a family of equivalence structures, then

$$\mathfrak{A} \text{ is } \mathbf{InfEx}_{\cong}[\nu]\text{-learnable } \Leftrightarrow \mathfrak{A} \text{ is finitely separable.}$$

Corollary

1. *If $\mathfrak{A} \notin \mathbf{InfEx}_{\cong}[\nu]$ and $\mathfrak{A}/_{\cong}$ is finite, then there exists $(\mathcal{A}, B) \subseteq \mathfrak{A}$ such that $(\mathcal{A}, B) \notin \mathbf{InfEx}_{\cong}[\nu]$.*
2. *There is $\mathfrak{A} \notin \mathbf{InfEx}_{\cong}[\nu])$ such that, for all $\mathfrak{B} \subseteq \mathfrak{A}$, if $\mathfrak{B}/_{\cong}$ is finite, then $\mathfrak{B} \in \mathbf{InfEx}_{\cong}[\nu]$.*

Theorem

$$\mathbf{0}\text{-}\mathbf{InfEx}_{\cong}[\nu] \subsetneq \mathbf{InfEx}_{\cong}[\nu]$$

Equivalence structures

Let $\mathbb{E}$ be the class of equivalence structures, and ν its enumeration.

Theorem (F., Kötzing, San Mauro, 2018)

If $\mathfrak{A}$ is a family of equivalence structures, then

$$\mathfrak{A} \text{ is } \mathbf{InfEx}_{\cong}[\nu]\text{-learnable } \Leftrightarrow \mathfrak{A} \text{ is finitely separable.}$$

Corollary

1. *If $\mathfrak{A} \notin \mathbf{InfEx}_{\cong}[\nu]$ and $\mathfrak{A}/_{\cong}$ is finite, then there exists $(\mathcal{A}, B) \subseteq \mathfrak{A}$ such that $(\mathcal{A}, B) \notin \mathbf{InfEx}_{\cong}[\nu]$.*
2. *There is $\mathfrak{A} \notin \mathbf{InfEx}_{\cong}[\nu]$) such that, for all $\mathfrak{B} \subseteq \mathfrak{A}$, if $\mathfrak{B}/_{\cong}$ is finite, then $\mathfrak{B} \in \mathbf{InfEx}_{\cong}[\nu]$.*

Theorem

$$\mathbf{0}\text{-}\mathbf{InfEx}_{\cong}[\nu] \subsetneq \mathbf{InfEx}_{\cong}[\nu] = \mathbf{0}''\text{-}\mathbf{InfEx}_{\cong}[\nu]$$

Other learnability classes, I

We obtain different learnability classes by replacing the main ingredients of $\mathbf{InfEx}_{\cong}$ with natural alternatives:

Other learnability classes, I

We obtain different learnability classes by replacing the main ingredients of $\mathbf{InfEx}_{\cong}$ with natural alternatives:

1. $\mathbf{Inf} \mapsto \mathbf{Txt}$: in $\mathbf{Txt}$-learning (short for *text*) the learner receives only positive information of the structure to be learnt.

Other learnability classes, I

We obtain different learnability classes by replacing the main ingredients of $\mathbf{InfEx}_{\cong}$ with natural alternatives:

1. $\mathbf{Inf} \mapsto \mathbf{Txt}$: in **Txt**-learning (short for *text*) the learner receives only positive information of the structure to be learnt.
2. $\cong \mapsto E$, where E is some nice equivalence relations relation between elements of $\mathbb{K}$, such as bi-embeddability ($\approx$), computable isomorphism ($\cong^{\mathbf{0}}$), computable bi-embeddability ($\approx^{\mathbf{0}}$) – and so forth.

Other learnability classes, I

We obtain different learnability classes by replacing the main ingredients of $\mathbf{InfEx}_{\cong}$ with natural alternatives:

1. $\mathbf{Inf} \mapsto \mathbf{Txt}$: in **Txt**-learning (short for *text*) the learner receives only positive information of the structure to be learnt.
2. $\cong \mapsto E$, where E is some nice equivalence relations relation between elements of $\mathbb{K}$, such as bi-embeddability ($\approx$), computable isomorphism ($\cong^{\mathbf{0}}$), computable bi-embeddability ($\approx^{\mathbf{0}}$) – and so forth.
3. $\mathbf{Ex} \mapsto \mathbf{BC}$: in **BC**-learning (short for *behaviourally correct*) the learner is allowed to change its mind infinitely many times as far as almost all its conjectures lie in the same E-class (with E defined as in 2.).
4. Yet another dimension to consider is the complexity of the learner.

Other learnability classes, II

$$
\begin{tikzcd}
\mathbf{InfEx}_{\cong\mathbf{0}} \arrow[r, "\subsetneq"] \arrow[d, "\subsetneq"'] & \mathbf{InfEx}_{\cong} \arrow[d, "\subsetneq"] \\
\mathbf{InfEx}_{\approx\mathbf{0}} \arrow[r, "\subsetneq"] & \mathbf{InfEx}_{\approx}
\end{tikzcd}
$$

Other learnability classes, II

$$\begin{tikzcd}
\mathbf{TxtEx}_{\cong^{\mathbf{0}}} \arrow[rr, "\subsetneq"] \arrow[dd, "\subsetneq"'] & & \mathbf{TxtEx}_{\cong} \arrow[dd, "\subsetneq"] \\
& & \\
\mathbf{TxtEx}_{\approx^{\mathbf{0}}} \arrow[rr, "\subsetneq"] & & \mathbf{TxtEx}_{\approx}
\end{tikzcd}$$

References

[1] C. Glymour. Inductive inference in the limit. Erkenntnis, 22:23–31, 1985.

[2] E. Martin and D. Osherson. Elements of scientific inquiry. MIT Press, 1998.

[3] F. Stephan and Yu. Ventsov. Learning algebraic structures from text. Theoret. Comput. Sci., 268(2):221–273, 2001.

[4] W. Merkle and F. Stephan. Trees and learning. J. Comput. System Sci., 68(1):134–156, 2004.

[5] V. Harizanov and F. Stephan. On the learnability of vector spaces. J. Comput. System Sci., 73(1):109–122, 2007.

[6] Z. Gao, F. Stephan, G. Wu, and A. Yamamoto. Learning Families of Closed Sets in Matroids. Computation, Physics and Beyond. WTCS 2012. LNCS, 7160: 120–139, 2012.

[7] E. Fokina, T. Kötzing, and L. San Mauro. Limit learning equivalence structures. Proceedings of the 30th International Conference on Algorithmic Learning Theory, volume 98 of PMLR, 383–403, Chicago, Illinois, 22–24 Mar 2019.

[8] N. Bazhenov, E. Fokina, and L. San Mauro. Learning families of algebraic structures from informant, Information and Computation, 275, 2020.