Learning Concepts Definable in First-Order Logic with Counting

Steffen van Bergerem
RWTH Aachen University

LICS 2019

Introduction — Classification Problems

Database

+

```
SELECT EXISTS (
  SELECT * FROM ‘data‘
  WHERE color=‘red‘ AND distance>5
);
```

Query

→

True/False

Boolean

Introduction — Classification Problems

Database

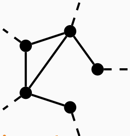
+

```
SELECT EXISTS (
  SELECT * FROM ‘data‘
  WHERE color=‘red‘ AND distance>5
);
```

Query

→ **True/False**

Boolean



Relational structure

+

$$\varphi = \exists x \exists y \; Exy$$

FO-formula

→ **True/False**

Boolean

Database

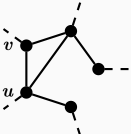
+

```
SELECT EXISTS (
  SELECT * FROM ‘data‘
  WHERE color=‘red‘ AND distance>5
);
```

Query

→ **True/False**

Boolean



Relational structure

+ $\varphi(x, y) = (x = y) \vee Exy$

FO-formula

→ **True/False**

Boolean

Introduction — Classification Problems

Database

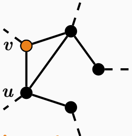
+

```
SELECT EXISTS (
  SELECT * FROM ‘data‘
  WHERE color=‘red‘ AND distance>5
);
```

Query

→ **True/False**

Boolean



Relational structure

+ $\varphi(x; y) = (x = y) \lor Exy$

FO-formula

→ **True/False**

Boolean

Introduction — Classification Problems

Database

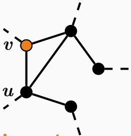
+

```
SELECT EXISTS (
  SELECT * FROM 'data'
  WHERE color='red' AND distance>5
);
```

Query

→ **True/False**

Boolean



Relational structure

+ $\varphi(x; y) = (x = y) \lor Exy$

FO-formula

→ **True/False**

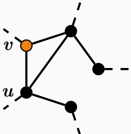
Boolean

fixed background structure

labeled examples

Introduction — Classification Problems

Database



Relational structure

fixed background structure

+

```
SELECT EXISTS (
  SELECT * FROM 'data'
  WHERE color='red' AND distance>5
);
```

Query

+

$$\varphi(x; y) = (x = y) \lor Exy$$

FO-formula

Task: learn model

→ **True/False**

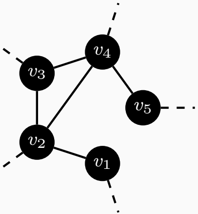
Boolean

→ **True/False**

Boolean

labeled examples

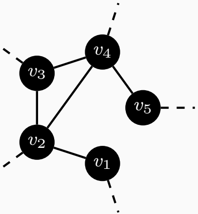
Introduction — FO Learning



Background structure

$((v_1, v_2), \text{False}), ((v_2, v_3), \text{True}), ((v_3, v_4), \text{True}),$
$((v_4, v_5), \text{False}), ((v_1, v_3), \text{True}), ((v_2, v_4), \text{True})$

Examples



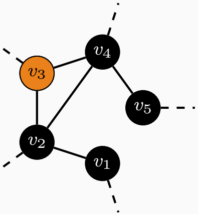
Background structure

$$((v_1, v_2), \text{False}), ((v_2, v_3), \text{True}), ((v_3, v_4), \text{True}),$$
$$((v_4, v_5), \text{False}), ((v_1, v_3), \text{True}), ((v_2, v_4), \text{True})$$

Examples

Task: learn a consistent model

$$\varphi(x_1, x_2; y) = \ ?$$



Background structure

$((v_1, v_2), \text{False}), ((v_2, v_3), \text{True}), ((v_3, v_4), \text{True}),$
$((v_4, v_5), \text{False}), ((v_1, v_3), \text{True}), ((v_2, v_4), \text{True})$

Examples

Task: learn a consistent model

$$\varphi(x_1, x_2; y) = \exists x_3 \, (Ex_1x_2 \wedge Ex_1x_3 \wedge Ex_2x_3) \vee (x_2 = y), \quad y = v_3$$

Grohe and Ritzert (2017):

There is a consistent model-learning algorithm for FO-formulas

that runs in sublinear time

on background structures of polylog. degree.

Idea: Use brute-force, Gaifman normal forms and Gaifman locality.

Grohe and Ritzert (2017):

There is a consistent model-learning algorithm for FO-formulas that runs in sublinear time on background structures of polylog. degree.

Idea: Use brute-force, Gaifman normal forms and Gaifman locality.

Grohe and Ritzert (2017):
There is a consistent model-learning algorithm
for FO-formulas
that runs in sublinear time
on background structures of polylog. degree.

Idea: Use brute-force, Gaifman normal forms and Gaifman locality.

Introduction — From SQL to FOCN(P)

Database

+

```
SELECT EXISTS (
  SELECT * FROM 'data'
  WHERE color='red' AND distance>5
);
```

Query

→ **True/False**

Boolean

- We would like to learn something similar to SQL queries
- FO can be viewed as the logical core of SQL
- Aggregate functions are missing:
  Count, Sum, Average, Min, Max
- Kuske and Schweikardt (2017):
  FOCN(P), adds counting to FO

Introduction — From SQL to FOCN(P)

Database

+

```
SELECT EXISTS (
  SELECT * FROM 'data'
  WHERE color='red' AND distance>5
);
```

Query

→ **True/False**

Boolean

- We would like to learn something similar to SQL queries
- FO can be viewed as the logical core of SQL
- Aggregate functions are missing:
  Count, Sum, Average, Min, Max
- Kuske and Schweikardt (2017):
  FOCN(P), adds counting to FO

Introduction — From SQL to FOCN(P)

Database

+

```
SELECT EXISTS (
  SELECT * FROM 'data'
  WHERE color='red' AND distance>5
);
```

Query

→

True/False

Boolean

- We would like to learn something similar to SQL queries
- FO can be viewed as the logical core of SQL
- Aggregate functions are missing:
  Count, Sum, Average, Min, Max
- Kuske and Schweikardt (2017):
  FOCN(P), adds counting to FO

There is a consistent model-learning algorithm
for FO-formulas
that runs in sublinear time
on background structures of polylog. degree.

Idea: Use brute-force, Gaifman normal forms and Gaifman locality.

Goal:

There is a consistent model-learning algorithm
for **FOCN(P)-formulas**
that runs in sublinear time
on background structures of polylog. degree.

Idea: Use brute-force, Gaifman normal forms and Gaifman locality.

Goal:

There is a consistent model-learning algorithm for **FOCN(P)-formulas** that runs in sublinear time on background structures of polylog. degree.

Idea: Use brute-force, **Hanf** normal forms and **Hanf** locality.

Introduction to FOCN(P)

Introduction to FOCN(P)

Counting terms: $\#\bar{y}.\varphi(\bar{y})$, $i \in \mathbb{Z}$, $(t_1 + t_2)$, $(t_1 \cdot t_2)$, κ

FOCN(P)-formulas: rules from FO, $P(t_1, \ldots, t_{\mathsf{ar}(P)})$, $\exists \kappa\, \varphi$

Example (κ-regular graph)

$$\varphi_1 = \exists \kappa\, \forall x\, (\underbrace{\#(y).Exy}_{t_{\text{edges}}(x)} = \kappa)$$

Example

$$\varphi_2(x, \kappa) = \underbrace{\#(y).Exy}_{t_{\text{edges}}(x)} + \#(y, z).(Exy \wedge Eyz) = \kappa + 4$$

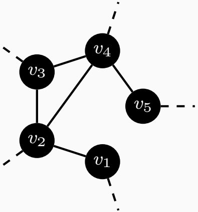
Learnability Results

Goal:

There is a consistent model-learning algorithm for **FOCN(P)-formulas** that runs in sublinear time on background structures of polylog. degree.

Idea: Use brute-force, **Hanf** normal forms and **Hanf** locality.

Learnability Results



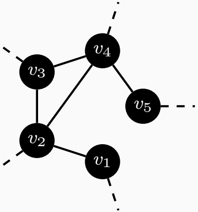
Background structure

$k = 2,\ \ell = 1$

Constants

Fixed	$\mathcal{B}$	relational background structure
	k	length of each tuple we should classify
	ℓ	number of parameters we should learn

Learnability Results



Background structure

$k = 2, \ \ell = 1$

Constants

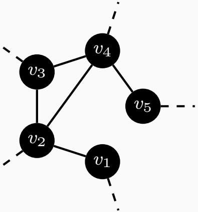
$((v_1, v_2), \text{False}), ((v_2, v_3), \text{True}), ((v_3, v_4), \text{True}),$
$((v_4, v_5), \text{False}), ((v_1, v_3), \text{True}), ((v_2, v_4), \text{True})$

Examples

Given $\mathcal{T} = ((\bar{u}_1, c_1), \ldots, (\bar{u}_t, c_t)), \quad u_i \in \left(U(\mathcal{B})\right)^k, \ c_i \in \{\text{True}, \text{False}\}$

training sequence of length t with tuples of length k

Learnability Results



Background structure

$k = 2,\ \ell = 1$

Constants

$((v_1, v_2), \text{False}), ((v_2, v_3), \text{True}), ((v_3, v_4), \text{True}),$
$((v_4, v_5), \text{False}), ((v_1, v_3), \text{True}), ((v_2, v_4), \text{True})$

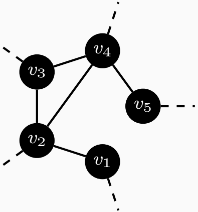
Examples

If there is a consistent model $(\varphi^*(\bar{x}; \bar{y}, \bar{\kappa}), \bar{v}^*, \bar{\lambda}^*)$,
φ^* FOCN(P)-formula

Return $\boldsymbol{\varphi(\bar{x}; \bar{y}, \bar{\kappa})}$ FOCN(P)-formula, $|x| = k,\ |y| = \ell$
$\boldsymbol{\bar{v}} \in \left(U(\mathcal{B})\right)^{\ell},\ \boldsymbol{\bar{\lambda}} \in \{1, \ldots, |U(\mathcal{B})|\}^{|\bar{\kappa}|}$ parameters

such that $[\![\varphi(\bar{x}; \bar{v}, \bar{\lambda})]\!]^{\mathcal{B}}$ is consistent with $\mathcal{T}$, i.e. $[\![\varphi(\bar{u}_i; \bar{v}, \bar{\lambda})]\!]^{\mathcal{B}} = c_i$

Learnability Results



Background structure

$k = 2,\ \ell = 1$

Constants

$((v_1, v_2), \text{False}), ((v_2, v_3), \text{True}), ((v_3, v_4), \text{True}),$
$((v_4, v_5), \text{False}), ((v_1, v_3), \text{True}), ((v_2, v_4), \text{True})$

Examples

If there is a consistent model $(\varphi^*(\bar{x}; \bar{y}, \bar{\kappa}), \bar{v}^*, \bar{\lambda}^*)$,
φ^* FOCN(P)-formula

Return $\boldsymbol{\varphi}(\bar{\boldsymbol{x}}; \bar{\boldsymbol{y}}, \bar{\kappa})$ FOCN(P)-formula, $|x| = k,\ |y| = \ell$
$\bar{\boldsymbol{v}} \in \left(U(\mathcal{B})\right)^{\ell},\ \bar{\boldsymbol{\lambda}} \in \{1, \ldots, |U(\mathcal{B})|\}^{|\bar{\kappa}|}$ parameters

such that $[\![\varphi(\bar{x}; \bar{v}, \bar{\lambda})]\!]^{\mathcal{B}}$ is consistent with $\mathcal{T}$, i.e. $[\![\varphi(\bar{u}_i; \bar{v}, \bar{\lambda})]\!]^{\mathcal{B}} = c_i$

Learnability Results

Theorem

Let $k, \ell \in \mathbb{N}$. There is an FOCN(P)-learning algorithm for the k-ary learning problem over some finite background structure $\mathcal{B}$ such that:

1. **If there is a consistent hypothesis** consisting of an FOCN(P)-formula with certain complexity bounds, a tuple of integers $\bar{\lambda}$ and a tuple in $\left(U(\mathcal{B})\right)^{\ell}$, then the algorithm returns a hypothesis.
2. **If the algorithm returns a hypothesis**, then the hypothesis consists of an FO-formula $\varphi(\bar{x}; \bar{y})$ with a certain locality bound and a tuple $\bar{v} \in \left(U(\mathcal{B})\right)^{\ell}$ and $[\![\varphi(\bar{x}; \bar{v})]\!]^{\mathcal{B}}$ is consistent with the training sequence.
3. It runs in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.
4. The hypothesis can be evaluated in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.

Proof idea: Use brute-force, Hanf normal forms and Hanf locality.

Learnability Results

Theorem

Let $k, \ell \in \mathbb{N}$. There is an FOCN(P)-learning algorithm for the k-ary learning problem over some finite background structure $\mathcal{B}$ such that:

1. **If there is a consistent hypothesis** consisting of an FOCN(P)-formula with certain complexity bounds, a tuple of integers $\bar{\lambda}$ and a tuple in $\left(U(\mathcal{B})\right)^{\ell}$, then the algorithm returns a hypothesis.
2. **If the algorithm returns a hypothesis**, then the hypothesis consists of an FO-formula $\varphi(\bar{x}; \bar{y})$ with a certain locality bound and a tuple $\bar{v} \in \left(U(\mathcal{B})\right)^{\ell}$ and $[\![\varphi(\bar{x}; \bar{v})]\!]^{\mathcal{B}}$ is consistent with the training sequence.
3. It runs in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.
4. The hypothesis can be evaluated in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.

Proof idea: Use brute-force, Hanf normal forms and Hanf locality.

Learnability Results

Theorem

Let $k, \ell \in \mathbb{N}$. There is an FOCN(P)-learning algorithm for the k-ary learning problem over some finite background structure $\mathcal{B}$ such that:

1. **If there is a consistent hypothesis** consisting of an FOCN(P)-formula with certain complexity bounds, a tuple of integers $\bar{\lambda}$ and a tuple in $\left(U(\mathcal{B})\right)^{\ell}$, then the algorithm returns a hypothesis.
2. **If the algorithm returns a hypothesis**, then the hypothesis consists of an FO-formula $\varphi(\bar{x}; \bar{y})$ with a certain locality bound and a tuple $\bar{v} \in \left(U(\mathcal{B})\right)^{\ell}$ and $[\![\varphi(\bar{x}; \bar{v})]\!]^{\mathcal{B}}$ is consistent with the training sequence.
3. It runs in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.
4. The hypothesis can be evaluated in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.

Proof idea: Use brute-force, Hanf normal forms and Hanf locality.

Learnability Results

Theorem

Let $k, \ell \in \mathbb{N}$. There is an FOCN(P)-learning algorithm for the k-ary learning problem over some finite background structure $\mathcal{B}$ such that:

1. **If there is a consistent hypothesis** consisting of an FOCN(P)-formula with certain complexity bounds, a tuple of integers $\bar{\lambda}$ and a tuple in $\left(U(\mathcal{B})\right)^{\ell}$, then the algorithm returns a hypothesis.
2. **If the algorithm returns a hypothesis**, then the hypothesis consists of an FO-formula $\varphi(\bar{x}; \bar{y})$ with a certain locality bound and a tuple $\bar{v} \in \left(U(\mathcal{B})\right)^{\ell}$ and $[\![\varphi(\bar{x}; \bar{v})]\!]^{\mathcal{B}}$ is consistent with the training sequence.
3. It runs in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.
4. The hypothesis can be evaluated in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.

Proof idea: Use brute-force, Hanf normal forms and Hanf locality.

Learnability Results

Theorem

Let $k, \ell \in \mathbb{N}$. There is an FOCN(P)-learning algorithm for the k-ary learning problem over some finite background structure $\mathcal{B}$ such that:

1. **If there is a consistent hypothesis** consisting of an FOCN(P)-formula with certain complexity bounds, a tuple of integers $\bar{\lambda}$ and a tuple in $\left(U(\mathcal{B})\right)^{\ell}$, then the algorithm returns a hypothesis.
2. **If the algorithm returns a hypothesis**, then the hypothesis consists of an FO-formula $\varphi(\bar{x}; \bar{y})$ with a certain locality bound and a tuple $\bar{v} \in \left(U(\mathcal{B})\right)^{\ell}$ and $[\![\varphi(\bar{x}; \bar{v})]\!]^{\mathcal{B}}$ is consistent with the training sequence.
3. It runs in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.
4. The hypothesis can be evaluated in time $(\log n + t)^{\mathcal{O}(1)} d^{\mathcal{O}((\log d)^c)}$ with only local access to $\mathcal{B}$.

Proof idea: Use brute-force, Hanf normal forms and Hanf locality.

Theorem

1. If there is a consistent hypothesis consisting of an FOCN(P)-formula with certain complexity bounds, a tuple of integers $\bar{\lambda}$ and a tuple in $U(\mathcal{B})^{\ell}$, then the algorithm returns a hypothesis.

→ Try all FOCN(P)-formulas with certain complexity bounds, all structure parameters and number parameters.

Learnability Results — Proof idea

Theorem

1. If there is a consistent hypothesis consisting of an FOCN(P)-formula with certain complexity bounds, a tuple of integers $\bar{\lambda}$ and a tuple in $U(\mathcal{B})^{\ell}$, then the algorithm returns a hypothesis. ✓

→ Try all FOCN(P)-formulas with certain complexity bounds, all structure parameters and number parameters.

Learnability Results — Proof idea

Theorem

2. If the algorithm returns a hypothesis, then the hypothesis consists of an FO-formula $\varphi(\bar{x};\bar{y})$ with a certain locality bound and a tuple $\bar{v} \in U(\mathcal{B})^{\ell}$ and $[\![\varphi(\bar{x};\bar{v})]\!]^{\mathcal{B}}$ is consistent with the training sequence.

→ Try all FOCN(P)-formulas with certain complexity bounds, all structure parameters and number parameters.

Learnability Results — Proof idea

Check all FOCN(P)-formulas with certain complexity bounds
all structure parameters
all number parameters

Learnability Results — Proof idea

Check all FOCN(P)-formulas with certain complexity bounds
all structure parameters
all number parameters

Theorem (Kuske and Schweikardt 2017)

For every degree bound $d \in \mathbb{N}$ and every FOCN(P)-formula φ with certain complexity bounds there exists a d-equivalent formula ψ in Hanf normal form with a certain locality bound.

Learnability Results — Proof idea

Check all FOCN(P)-formulas in Hanf normal form
all structure parameters with certain locality bounds
all number parameters

Theorem (Kuske and Schweikardt 2017)

For every degree bound $d \in \mathbb{N}$ and every FOCN(P)-formula φ with certain complexity bounds there exists a d-equivalent formula ψ in Hanf normal form with a certain locality bound.

Learnability Results — Proof idea

Check all FOCN(P)-formulas in Hanf normal form
all structure parameters with certain locality bounds
all number parameters

Fact (Hanf normal form)

A formula is in Hanf normal form if it is a Boolean combination of sphere-formulas and numerical conditions.

Fact

Every sphere-formula is an FO-formula.

Fact

For a fixed background structure and a fixed number parameter, the numerical conditions become constant.

Learnability Results — Proof idea

Check all Boolean combinations of sphere-formulas
all structure parameters with certain locality bounds
~~all number parameters~~

Fact (Hanf normal form)

A formula is in Hanf normal form if it is a Boolean combination of sphere-formulas and numerical conditions.

Fact

Every sphere-formula is an FO-formula.

Fact

For a fixed background structure and a fixed number parameter, the numerical conditions become constant.

Learnability Results — Proof idea

Check all Boolean combinations of sphere-formulas with certain locality bounds

all structure parameters

~~all number parameters~~

Fact

A sphere-formula is an FO-formula that exactly characterizes the r-neighborhood of a tuple up to isomorphism.

$$\mathcal{B} \models \mathsf{sph}_{\mathcal{N}_r(\bar{u})}(\bar{v}) \iff \mathcal{N}_r(\bar{u}) \cong \mathcal{N}_r(\bar{v})$$

Learnability Results — Proof idea

Check all Boolean combinations of sphere-formulas

with certain locality bounds

all structure parameters

~~all number parameters~~

Fact

A sphere-formula is an FO-formula that exactly characterizes the r-neighborhood of a tuple up to isomorphism.

$$\mathcal{B} \models \mathrm{sph}_{\mathcal{N}_r(\bar{u})}(\bar{v}) \iff \mathcal{N}_r(\bar{u}) \cong \mathcal{N}_r(\bar{v})$$

Fact

There is a parameter $\bar{v}^*$ such that $\mathcal{N}_r(\bar{u}_i\bar{v}^*) \ncong \mathcal{N}_r(\bar{u}_j\bar{v}^*)$ for all positive examples $\bar{u}_i$ and negative examples $\bar{u}_j$.

Learnability Results — Proof idea

Check $\varphi^*(\bar{x}\,;\bar{y}) = \bigvee_{i\in[t],\ c_i=\text{True}} \text{sph}_{\mathcal{N}_r(\bar{u}_i\bar{v}^*)}(\bar{x}\bar{y})$

all structure parameters

~~all number parameters~~

Fact

A sphere-formula is an FO-formula that exactly characterizes the r-neighborhood of a tuple up to isomorphism.

$$\mathcal{B} \models \text{sph}_{\mathcal{N}_r(\bar{u})}(\bar{v}) \iff \mathcal{N}_r(\bar{u}) \cong \mathcal{N}_r(\bar{v})$$

Fact

There is a parameter $\bar{v}^*$ such that $\mathcal{N}_r(\bar{u}_i\bar{v}^*) \ncong \mathcal{N}_r(\bar{u}_j\bar{v}^*)$ for all positive examples $\bar{u}_i$ and negative examples $\bar{u}_j$.

Learnability Results — Proof idea

Check $\varphi^*(\bar{x}\,;\bar{y}) = \bigvee_{i\in[t],\, c_i=\text{True}} \text{sph}_{\mathcal{N}_r(\bar{u}_i\bar{v}^*)}(\bar{x}\bar{y})$

all structure parameters

~~all number parameters~~

Use locality to reduce number of possible parameters.

Input:

Training sequence $T = ((\bar{u}_1, c_1)), \dots, (\bar{u}_t, c_t)) \in \mathcal{T}$, $d = \Delta\mathcal{B}$,
local access to background structure $\mathcal{B}$

for all $\bar{v}^* \in (N_{r'}(T))^\ell$ **do**
  $\varphi^*(\bar{x}\,;\bar{y}) \leftarrow \bigvee_{i\in[t],\, c_i=\text{True}} \mathsf{sph}_{\mathcal{N}_r(\bar{u}_i\bar{v}^*)}(\bar{x}\bar{y})$
  $consistent \leftarrow$ **true**
  for $i \in [t]$ with $c_i =$ **false do**
    for $j \in [t]$ with $c_j =$ **true do**
      if $\mathcal{N}_r(\bar{u}_i\bar{v}^*) \cong \mathcal{N}_r(\bar{u}_j\bar{v}^*)$ **then**
        $consistent \leftarrow$ **false**
  if $consistent$ **then**
    return $(\varphi^*(\bar{x}\,;\bar{y}),\ \bar{v}^*)$
reject

Learnability Results — Algorithm

Input:

Training sequence $T = ((\bar{u}_1, c_1)), \ldots, (\bar{u}_t, c_t)) \in \mathcal{T}$, $d = \Delta\mathcal{B}$,
local access to background structure $\mathcal{B}$

for all $\bar{v}^* \in (N_{r'}(T))^{\ell}$ **do**
  $\varphi^*(\bar{x}\,;\bar{y}) \leftarrow \bigvee_{i \in [t],\, c_i = \text{True}} \mathsf{sph}_{\mathcal{N}_r(\bar{u}_i \bar{v}^*)}(\bar{x}\bar{y})$
  $consistent \leftarrow$ **true**
  for $i \in [t]$ with $c_i =$ **false do**
    for $j \in [t]$ with $c_j =$ **true do**
      if $\mathcal{N}_r(\bar{u}_i \bar{v}^*) \cong \mathcal{N}_r(\bar{u}_j \bar{v}^*)$ **then**
        $consistent \leftarrow$ **false**
  if $consistent$ **then**
    return $(\varphi^*(\bar{x}\,;\bar{y}),\ \bar{v}^*)$
reject

Corollary

There is a consistent model-learning algorithm for FOCN(P)-formulas that runs in sublinear time on background structures of polylog. degree.

Non-Learnability Results

Non-Learnability Results

Theorem

There is no consistent sublinear formula-learning algorithm for FO-formulas with only local access on background structures of unbounded degree.

Proof idea: Sublinear-time algorithms cannot see the whole structure.

→ Hide important parts of the structure from the algorithm.

Non-Learnability Results

Theorem

There is no consistent sublinear formula-learning algorithm for FO-formulas with only local access on background structures of unbounded degree.

Proof idea: Sublinear-time algorithms cannot see the whole structure.

➔ Hide important parts of the structure from the algorithm.

Non-Learnability Results

If the exponential-time hypothesis (ETH) holds:

Theorem

There is no consistent parameter-learning algorithm for first-order formulas φ of quantifier rank at most q on background structures $\mathcal{B}$ with no degree restriction running in time $|\mathcal{B}|^{o(q)}$, i.e. that, given φ and a sequence of training examples T, returns a tuple $\bar{v}$ such that $[\![\varphi(\bar{x}\,;\bar{v})]\!]^{\mathcal{B}}$ is consistent with all training examples.

Proof idea: Solve q-Clique by learning the parameter.

Non-Learnability Results

If the exponential-time hypothesis (ETH) holds:

Theorem

There is no consistent parameter-learning algorithm for first-order formulas φ of quantifier rank at most q on background structures $\mathcal{B}$ with no degree restriction running in time $|\mathcal{B}|^{o(q)}$, i.e. that, given φ and a sequence of training examples T, returns a tuple $\bar{v}$ such that $[\![\varphi(\bar{x}\,;\bar{v})]\!]^{\mathcal{B}}$ is consistent with all training examples.

Proof idea: Solve q-Clique by learning the parameter.

Conclusion

Conclusion

FOCN(P) extends FO and allows counting.

Results

There is ...

- a consistent sublinear model-learning algorithm for FOCN(P)-formulas on structures of polylog. degree.
- no consistent sublinear model-learning algorithm for FO-formulas with only local access on structures of unbounded degree.
- no consistent parameter-learning algorithm for FO-formulas running in time $|\mathcal{B}|^{o(q)}$ on structures of unbounded degree.

Open Questions

- Other aggregators from SQL? (Sum, Average, Min, Max)
- Better lower bounds for unbounded degree?

Conclusion

FOCN(P) extends FO and allows counting.

Results

There is ...

- a consistent sublinear model-learning algorithm for FOCN(P)-formulas on structures of polylog. degree.
- no consistent sublinear model-learning algorithm for FO-formulas with only local access on structures of unbounded degree.
- no consistent parameter-learning algorithm for FO-formulas running in time $|\mathcal{B}|^{o(q)}$ on structures of unbounded degree.

Open Questions

- Other aggregators from SQL? (Sum, Average, Min, Max)
- Better lower bounds for unbounded degree?

Conclusion

FOCN(P) extends FO and allows counting.

Results

There is ...

- a consistent sublinear model-learning algorithm for FOCN(P)-formulas on structures of polylog. degree.
- no consistent sublinear model-learning algorithm for FO-formulas with only local access on structures of unbounded degree.
- no consistent parameter-learning algorithm for FO-formulas running in time $|\mathcal{B}|^{o(q)}$ on structures of unbounded degree.

Open Questions

- Other aggregators from SQL? (Sum, Average, Min, Max)
- Better lower bounds for unbounded degree?

Conclusion

FOCN(P) extends FO and allows counting.

Results

There is ...

- a consistent sublinear model-learning algorithm for FOCN(P)-formulas on structures of polylog. degree.
- no consistent sublinear model-learning algorithm for FO-formulas with only local access on structures of unbounded degree.
- no consistent parameter-learning algorithm for FO-formulas running in time $|\mathcal{B}|^{o(q)}$ on structures of unbounded degree.

Open Questions

- Other aggregators from SQL? (Sum, Average, Min, Max)
- Better lower bounds for unbounded degree?

Conclusion

FOCN(P) extends FO and allows counting.

Results

There is ...

- a consistent sublinear model-learning algorithm for FOCN(P)-formulas on structures of polylog. degree.
- no consistent sublinear model-learning algorithm for FO-formulas with only local access on structures of unbounded degree.
- no consistent parameter-learning algorithm for FO-formulas running in time $|\mathcal{B}|^{o(q)}$ on structures of unbounded degree.

Open Questions

- Other aggregators from SQL? (Sum, Average, Min, Max)
- Better lower bounds for unbounded degree?