Learning Logically Defined Hypotheses

Martin Grohe
RWTH Aachen

Outline

I. A Declarative Model-Theoretic Framework for ML

II. First-Order Hypotheses on Low-Degree Structures
(joint work with Martin Ritzert)

III. Monadic Second-Order Hypotheses on Strings
(joint work with Christof Löding and Martin Ritzert)

IV. Concluding Remarks

A Declarative Model-Theoretic Framework for ML

Observations about today's ML practice

- Algorithmic focus: goal is to approximate an unknown function as well as possible (rather than understanding the function)

Observations about today's ML practice

- Algorithmic focus: goal is to approximate an unknown function as well as possible (rather than understanding the function)
- It is difficult for a non-expert user to decide which algorithm (with which "hyper-parameter" settings, network topology, ...) to use

Observations about today's ML practice

- Algorithmic focus: goal is to approximate an unknown function as well as possible (rather than understanding the function)
- It is difficult for a non-expert user to decide which algorithm (with which "hyper-parameter" settings, network topology, ...) to use
- Models are determined by the algorithm and often have little meaning beyond that.

Observations about today's ML practice

- Algorithmic focus: goal is to approximate an unknown function as well as possible (rather than understanding the function)
- It is difficult for a non-expert user to decide which algorithm (with which "hyper-parameter" settings, network topology, ...) to use
- Models are determined by the algorithm and often have little meaning beyond that.
- It is difficult to understand and explain what the models do.

Observations about today's ML practice

- Algorithmic focus: goal is to approximate an unknown function as well as possible (rather than understanding the function)
- It is difficult for a non-expert user to decide which algorithm (with which "hyper-parameter" settings, network topology, ...) to use
- Models are determined by the algorithm and often have little meaning beyond that.
- It is difficult to understand and explain what the models do.

Observations about today's ML practice

- Algorithmic focus: goal is to approximate an unknown function as well as possible (rather than understanding the function)
- It is difficult for a non-expert user to decide which algorithm (with which "hyper-parameter" settings, network topology, ...) to use
- Models are determined by the algorithm and often have little meaning beyond that.
- It is difficult to understand and explain what the models do.

Declarative approach

Try to separate model from solver as far as possible.

Idea of Model-Theoretic Framework

Background structure

Background knowledge represented by logical structure,

Idea of Model-Theoretic Framework

Background structure

Background knowledge represented by logical structure, which, for example, may capture

- arithmetical knowledge (e.g. field of real numbers, some Hilbert space)

Idea of Model-Theoretic Framework

Background structure

Background knowledge represented by logical structure, which, for example, may capture

- arithmetical knowledge (e.g. field of real numbers, some Hilbert space)
- structural knowledge (e.g. webgraph, relational data)

Idea of Model-Theoretic Framework

Background structure

Background knowledge represented by logical structure, which, for example, may capture

- arithmetical knowledge (e.g. field of real numbers, some Hilbert space)
- structural knowledge (e.g. webgraph, relational data)

Parametric model

Model described by formula of a suitable logic, which usually has certain free variables for parameters.

Example 1

Goal

Try to predict chances on academic job market based on publication data.

Example 1

Goal

Try to predict chances on academic job market based on publication data.

We view this as a Boolean classification problem: instances are applicants, and the question is whether they get a job or not.

Example 1

Goal

Try to predict chances on academic job market based on publication data.

We view this as a Boolean classification problem: instances are applicants, and the question is whether they get a job or not.

Data

A list of applicants, or rather certain pieces of information about the applicants, labelled by the information of whether they succeeded or not.

Example 1 (cont'd)

Scenario 1

Suppose for each person we only have the following information:

$$p = \text{number of publications}, \qquad t = \text{years since PhD}.$$

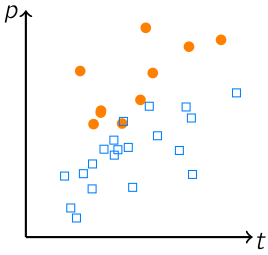
Example 1 (cont'd)

Scenario 1

Suppose for each person we only have the following information:

$$p = \text{number of publications}, \qquad t = \text{years since PhD}.$$

Data



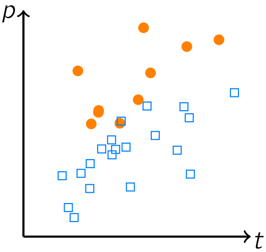
Example 1 (cont'd)

Scenario 1

Suppose for each person we only have the following information:

$$p = \text{number of publications}, \qquad t = \text{years since PhD}.$$

Data



Model

Linear model with two parameters a, b:

$$p \geq at + b.$$

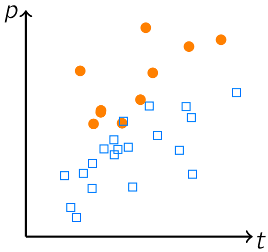
Example 1 (cont'd)

Scenario 1

Suppose for each person we only have the following information:

$$p = \text{number of publications}, \qquad t = \text{years since PhD.}$$

Data



Model

Linear model with two parameters a, b:

$$p \geq at + b.$$

Representation in our framework

- Background structures: ordered field of the reals
- Model: $\varphi(x_1, x_2\,; y_1, y_2) := (x_1 \geq y_1 \cdot x_2 + y_2)$

Example 1 (cont'd)

Scenario 2

We have a publication database of a schema that includes relations

- AUTHOR(auth-id, name, affill)
- PUB(pub-id, auth-id, title, journal, year, . . .)

Example 1 (cont'd)

Scenario 2

We have a publication database of a schema that includes relations

- AUTHOR(auth-id, name, affill)
- PUB(pub-id, auth-id, title, journal, year, . . .)

The database is our background structure.

Example 1 (cont'd)

Scenario 2

We have a publication database of a schema that includes relations

- AUTHOR(auth-id, name, affill)
- PUB(pub-id, auth-id, title, journal, year, ...)

The database is our background structure.

Our model may say something like the following (with parameters $a, b, c_1, \ldots, c_m, d_1, \ldots, d_n$):

- the candidate has at least a publications on average per year
- and at least b single-author publications
- and either a joint publication with an author from one of the universities $c_1, \ldots, c_m$
- or a publication in one of the journals $d_1, \ldots, d_n$.

Example 1 (cont'd)

Scenario 2

We have a publication database of a schema that includes relations

- AUTHOR(auth-id, name, affill)
- PUB(pub-id, auth-id, title, journal, year, . . .)

The database is our background structure.

Our model may say something like the following (with parameters $a, b, c_1, \ldots, c_m, d_1, \ldots, d_n$):

- the candidate has at least a publications on average per year
- and at least b single-author publications
- and either a joint publication with an author from one of the universities $c_1, \ldots, c_m$
- or a publication in one of the journals $d_1, \ldots, d_n$.

This can be expressed by a SQL query $\varphi(x\,; y_1, \ldots, y_{m+n+2})$.

Example 2

Goal

Learn a formula of monadic second-order logic (or a regular expression) that selects certain positions in a string.

Example 2

Goal

Learn a formula of monadic second-order logic (or a regular expression) that selects certain positions in a string.

Data

Fragment of the string with certain positions marked.

Example 2

Goal

Learn a formula of monadic second-order logic (or a regular expression) that selects certain positions in a string.

Data

Fragment of the string with certain positions marked.

```
\begin{frame}
  \frametitle{Formal Framework}

  For simplicity, we only consider \alert{Boolean classifi
problems} here.

  \begin{block}{Background structure}
    Finite or infinite structure $\red B$ with universe $U(

    \pause
    \alert{Instance space} is $U(B)^k$ for some $k$.
  \end{block}

   \medskip\pause
   \begin{block}{Parametric model}
    Formula $\red\phi(\bar x\smid\bar y)$ of some logic L.

    $\bar x=(x_1,\ldots,x_k)$ \alert{instance variables}.

    $\bar y=(y_1,\ldots,y_\ell)$ (for some $\ell$) \alert{p
variables}.
  \end{block}

  \medskip\pause
  \begin{block}{Hypotheses}
    For each parameter tuple $\bar v\in U(B)^\ell$ a functi
    ${\red\llbracket\phi(\bar x\smid\bar v)\rrbracket^B}\co
U(B)^k\to\{0,1\}$ by
    \[
    \llbracket \phi(\bar x\smid\bar v)\rrbracket^B(\bar u):
    \begin{cases}
      1&\text{if } B\models\phi(\bar u\smid\bar v),\\
```

Example 2

Goal

Learn a formula of monadic second-order logic (or a regular expression) that selects certain positions in a string.

Data

Fragment of the string with certain positions marked.

Model

Select all positions with letter 'B' in LaTeX math mode.

```
\begin{frame}
  \frametitle{Formal Framework}

  For simplicity, we only consider \alert{Boolean classifi
problems} here.

  \begin{block}{Background structure}
    Finite or infinite structure $\red B$ with universe $U(

    \pause
    \alert{Instance space} is $U(B)^k$ for some $k$.
  \end{block}

   \medskip\pause
   \begin{block}{Parametric model}
    Formula $\red\phi(\bar x\smid\bar y)$ of some logic L.

    $\bar x=(x_1,\ldots,x_k)$ \alert{instance variables}.

    $\bar y=(y_1,\ldots,y_\ell)$ (for some $\ell$) \alert{p
variables}.
  \end{block}

  \medskip\pause
  \begin{block}{Hypotheses}
    For each parameter tuple $\bar v\in U(B)^\ell$ a functi
    ${\red\llbracket\phi(\bar x\smid\bar v)\rrbracket^B}\co
U(B)^k\to\{0,1\}$ by
    \[
    \llbracket \phi(\bar x\smid\bar v)\rrbracket^B(\bar u):
    \begin{cases}
      1&\text{if } B\models\phi(\bar u\smid\bar v),\\
```

Formal Framework

For simplicity, we only consider Boolean classification problems.

Formal Framework

For simplicity, we only consider Boolean classification problems.

Background structure

Finite or infinite structure B with universe $U(B)$.

Formal Framework

For simplicity, we only consider Boolean classification problems.

Background structure

Finite or infinite structure B with universe $U(B)$.
Instance space is $U(B)^k$ for some k. We call k the dimension of the problem.

Formal Framework

For simplicity, we only consider Boolean classification problems.

Background structure

Finite or infinite structure B with universe $U(B)$.
Instance space is $U(B)^k$ for some k. We call k the dimension of the problem.

Parametric model

Formula $\varphi(\bar{x}\,;\bar{y})$ of some logic L.
$\bar{x} = (x_1, \ldots, x_k)$ instance variables.
$\bar{y} = (y_1, \ldots, y_\ell)$ (for some ℓ) parameter variables.

For simplicity, we only consider Boolean classification problems.

Background structure

Finite or infinite structure B with universe $U(B)$.
Instance space is $U(B)^k$ for some k. We call k the dimension of the problem.

Parametric model

Formula $\varphi(\bar{x}\,;\bar{y})$ of some logic L.
$\bar{x} = (x_1, \ldots, x_k)$ instance variables.
$\bar{y} = (y_1, \ldots, y_\ell)$ (for some ℓ) parameter variables.

Hypotheses

For each parameter tuple $\bar{v} \in U(B)^\ell$ a Boolean function $[\![\varphi(\bar{x}\,;\bar{v})]\!]^B \colon U(B)^k \to \{0,1\}$ defined by

$$[\![\varphi(\bar{x}\,;\bar{v})]\!]^B(\bar{u}) := \begin{cases} 1 & \text{if } B \models \varphi(\bar{u}\,;\bar{v}), \\ 0 & \text{otherwise.} \end{cases}$$

Remarks

- Background structure may capture both abstract knowledge and (potentially very large) data sets and relations between them

Remarks

- Background structure may capture both abstract knowledge and (potentially very large) data sets and relations between them
- Usually, only a small part of of the background structure can be inspected at runtime

Remarks

- Background structure may capture both abstract knowledge and (potentially very large) data sets and relations between them
- Usually, only a small part of of the background structure can be inspected at runtime
- At this point it is wide open what may constitute good logics for specifying models.

Remarks

- Background structure may capture both abstract knowledge and (potentially very large) data sets and relations between them
- Usually, only a small part of of the background structure can be inspected at runtime
- At this point it is wide open what may constitute good logics for specifying models.
- Approach probably best suited for applications where specifications in some kind of logic or formal language are common, such as verification or database systems.

Input

Learning algorithms have access to background structure B and receive as input a training sequence T of labelled examples:

$$(\bar{u}_1, \lambda_1), \ldots, (\bar{u}_t, \lambda_t) \in U(B)^k \times \{0, 1\}.$$

Input

Learning algorithms have access to background structure B and receive as input a training sequence T of labelled examples:

$$(\bar{u}_1, \lambda_1), \ldots, (\bar{u}_t, \lambda_t) \in U(B)^k \times \{0, 1\}.$$

Goal

Find hypothesis of the form $[\![\varphi(\bar{x}\,;\bar{v})]\!]^B$ that generalises well, that is, predicts true target values for instances $\bar{u} \in U(B)^k$ well.

The training error $\mathsf{err}_T(H)$ (a.k.a. empirical risk) of a hypothesis H on a training sequence T is the fraction of examples in T labelled wrong by H.

Learning as Minimisation

The training error $\mathsf{err}_T(H)$ (a.k.a. empirical risk) of a hypothesis H on a training sequence T is the fraction of examples in T labelled wrong by H.

Typically, a learning algorithm will try to minimise

$$\mathsf{err}_T(H) + \rho(H),$$

where H ranges over hypotheses from a hypothesis class $\mathcal{H}$ and a $\rho(H)$ is a regularisation term.

Learning as Minimisation

The training error $\mathsf{err}_T(H)$ (a.k.a. empirical risk) of a hypothesis H on a training sequence T is the fraction of examples in T labelled wrong by H.

Typically, a learning algorithm will try to minimise

$$\mathsf{err}_T(H) + \rho(H),$$

where H ranges over hypotheses from a hypothesis class $\mathcal{H}$ and a $\rho(H)$ is a regularisation term.

In our setting,

- $\mathcal{H}$ is a set of hypothesis of the form $[\![\varphi(\bar{x}\,;\bar{v})]\!]^B$.

Learning as Minimisation

The training error $\mathsf{err}_T(H)$ (a.k.a. empirical risk) of a hypothesis H on a training sequence T is the fraction of examples in T labelled wrong by H.

Typically, a learning algorithm will try to minimise

$$\mathsf{err}_T(H) + \rho(H),$$

where H ranges over hypotheses from a hypothesis class $\mathcal{H}$ and a $\rho(H)$ is a regularisation term.

In our setting,

- $\mathcal{H}$ is a set of hypothesis of the form $[\![\varphi(\bar{x}\,;\bar{v})]\!]^B$.
- $\rho(H)$ only depends on φ (typically function of quantifier rank).

Learning as Minimisation

The training error $\mathsf{err}_T(H)$ (a.k.a. empirical risk) of a hypothesis H on a training sequence T is the fraction of examples in T labelled wrong by H.

Typically, a learning algorithm will try to minimise

$$\mathsf{err}_T(H) + \rho(H),$$

where H ranges over hypotheses from a hypothesis class $\mathcal{H}$ and a $\rho(H)$ is a regularisation term.

In our setting,

- $\mathcal{H}$ is a set of hypothesis of the form $[\![\varphi(\bar{x}\,;\bar{v})]\!]^B$.
- $\rho(H)$ only depends on φ (typically function of quantifier rank).

Often we regard φ or at least its quantifier rank fixed. Then this amounts to empirical risk minimisation (ERM).

Remarks on VC-Dimension and PAC-Learning

- The classes of definable hypotheses we consider here tend to have bounded VC-dimension (G. and Turán 2004; Adler and Adler 2014).

Remarks on VC-Dimension and PAC-Learning

- The classes of definable hypotheses we consider here tend to have bounded VC-dimension (G. and Turán 2004; Adler and Adler 2014).
- This implies PAC-learnability (in an information theoretic sense).

Remarks on VC-Dimension and PAC-Learning

- The classes of definable hypotheses we consider here tend to have bounded VC-dimension (G. and Turán 2004; Adler and Adler 2014).
- This implies PAC-learnability (in an information theoretic sense).
- However, it comes without any guarantees on efficiency.

Computation Model

- We assume a standard RAM computation model with a uniform cost measure.

Computation Model

- We assume a standard RAM computation model with a uniform cost measure.
- For simplicity, in this talk we always assume the background structure to be finite.

Computation Model

- We assume a standard RAM computation model with a uniform cost measure.
- For simplicity, in this talk we always assume the background structure to be finite.
- However, we still assume the structure to be very large, and we want our learning algorithms to run in sublinear time in the size of the structure.

Computation Model

- We assume a standard RAM computation model with a uniform cost measure.
- For simplicity, in this talk we always assume the background structure to be finite.
- However, we still assume the structure to be very large, and we want our learning algorithms to run in sublinear time in the size of the structure.
- To be able to do meaningful computations in sublinear time, we usually need some form of local access to the structure.

Computation Model

- We assume a standard RAM computation model with a uniform cost measure.
- For simplicity, in this talk we always assume the background structure to be finite.
- However, we still assume the structure to be very large, and we want our learning algorithms to run in sublinear time in the size of the structure.
- To be able to do meaningful computations in sublinear time, we usually need some form of local access to the structure.

  For example, we should be able to access the neighbours of a vertex in a graph.

Complexity Considerations

- We strive for algorithms running in time polynomial in the size of the training data, regardless of the size of the background structure (or at most polylogarithmic in the size of the background structure).

Complexity Considerations

- We strive for algorithms running in time polynomial in the size of the training data, regardless of the size of the background structure (or at most polylogarithmic in the size of the background structure).
- With respect to the formula $\varphi(\bar{x}\,;\bar{y})$, we take a data complexity point of view (common in database theory): we ignore contribution of the formula to the running time, or equivalently, assume the dimension, the number of parameters, and the quantifier rank of φ to be fixed.

Complexity Considerations

- We strive for algorithms running in time polynomial in the size of the training data, regardless of the size of the background structure (or at most polylogarithmic in the size of the background structure).
- With respect to the formula $\varphi(\bar{x}\,;\bar{y})$, we take a data complexity point of view (common in database theory): we ignore contribution of the formula to the running time, or equivalently, assume the dimension, the number of parameters, and the quantifier rank of φ to be fixed.
- Then we can simply ignore the regularisation term (only depending on φ) and follow the ERM paradigm:
  we need to find a formula of quantifier rank at most q and a parameter tuple that minimise the training error.

First-Order Hypotheses on Low-Degree Structures

Theorem (G., Ritzert 2017)

There is a learner for FO running in time

$$(d + t)^{O(1)}$$

where

- $t = |T|$ *is the length of the training sequence*
- d *is the maximum degree of the background structure* B
- *the constant hidden in the* $O(1)$ *depends on* q, k, ℓ.

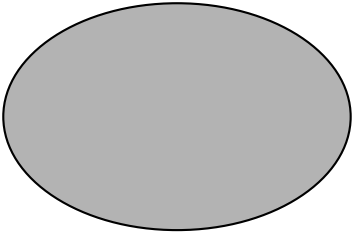
Proof

Idea

Exploit locality of FO (Gaifman's Theorem).

Idea

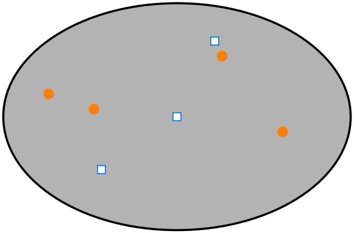
Exploit locality of FO (Gaifman's Theorem).



Proof

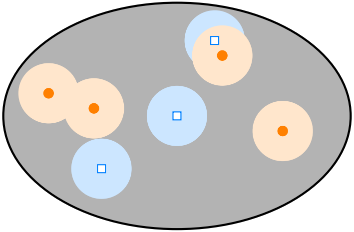
Idea

Exploit locality of FO (Gaifman's Theorem).



Idea

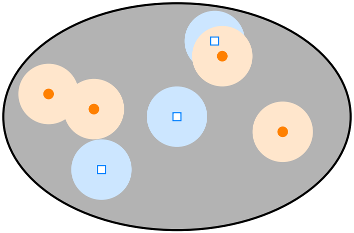
Exploit locality of FO (Gaifman's Theorem).



Proof

Idea

Exploit locality of FO (Gaifman's Theorem).

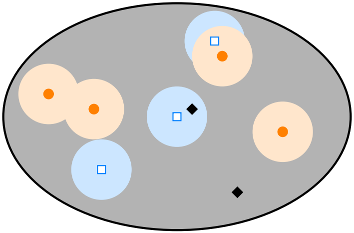


Key Lemma

Parameters far away from all training examples are irrelevant.

Proof

Idea
Exploit locality of FO (Gaifman's Theorem).

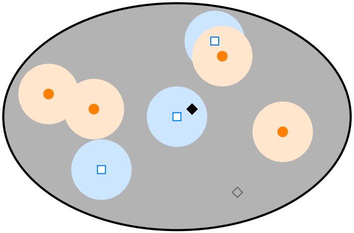


Key Lemma
Parameters far away from all training examples are irrelevant.

Proof

Idea

Exploit locality of FO (Gaifman's Theorem).



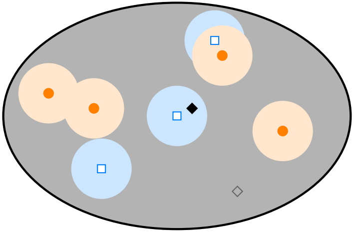
Key Lemma

Parameters far away from all training examples are irrelevant.

Proof

Idea

Exploit locality of FO (Gaifman's Theorem).



Key Lemma

Parameters far away from all training examples are irrelevant.

Algorithm

Search through all local formulas of desired quantifier rank and all parameter settings close to training points and check which hypothesis has the smallest training error.

Monadic Second-Order Hypotheses on Strings

Strings as Background Structures

String $a_1 \dots a_n$ over alphabet Σ viewed as structure with

- universe $\{1, \dots, n\}$,
- binary order relation $\leq$ on positions,
- for each $a \in \Sigma$ a unary relation R_a that contains all positions i such that $a_i = a$.

Strings as Background Structures

String $a_1 \ldots a_n$ over alphabet Σ viewed as structure with

- universe $\{1, \ldots, n\}$,
- binary order relation $\leq$ on positions,
- for each $a \in \Sigma$ a unary relation R_a that contains all positions i such that $a_i = a$.

Example

baaaacabcaaaaaaaaabaaaaabaacaaaaaabaaaaaacaaaaabbacccaacbcba

Strings as Background Structures

String $a_1 \ldots a_n$ over alphabet Σ viewed as structure with

- universe $\{1, \ldots, n\}$,
- binary order relation $\leq$ on positions,
- for each $a \in \Sigma$ a unary relation R_a that contains all positions i such that $a_i = a$.

Example

$baaaacabcaaaaaaaaabaaaaabaacaaaaaabaaaaaacaaaaabbaccca acbcba$

Strings as Background Structures

String $a_1 \dots a_n$ over alphabet Σ viewed as structure with

- universe $\{1, \dots, n\}$,
- binary order relation $\leq$ on positions,
- for each $a \in \Sigma$ a unary relation R_a that contains all positions i such that $a_i = a$.

Example

baaaacabcaaaaaaaaabaaaaabaacaaaaaabaaaaaacaaaaabbaccc aacbcba

Formula

$$\varphi(x\,;y) = R_a(x) \wedge \exists z\Big(z < x \wedge \forall z'\big(z < z' < x \to R_a(z')\big) \wedge \big((R_b(z) \wedge z < y) \vee (R_c(z) \wedge z \geq y)\big)\Big)$$

with parameter $v = 35$ consistent with training examples.

Learning with Local Access

Local access in a string means that for each position we can retrieve the previous and the next position.

Learning with Local Access

Local access in a string means that for each position we can retrieve the previous and the next position.

Theorem (G., Löding, Ritzert 2017)

1. *There are learners running in time* $t^{O(1)}$ *for quantifier-free formulas and 1-dimensional existential formulas over strings.*

Learning with Local Access

Local access in a string means that for each position we can retrieve the previous and the next position.

Theorem (G., Löding, Ritzert 2017)

1. *There are learners running in time $t^{O(1)}$ for quantifier-free formulas and 1-dimensional existential formulas over strings.*

2. *There is no sublinear learning algorithm for $\exists\forall$-formulas or 2-dimensional existential formulas over strings.*

Monadic Second-Order Logic

Monadic Second-Order Logic (MSO) is the extension of first-order logic FO that allows quantification not only over the elements of a structure, but also over sets of elements.

Monadic Second-Order Logic

Monadic Second-Order Logic (MSO) is the extension of first-order logic FO that allows quantification not only over the elements of a structure, but also over sets of elements.

Theorem (Büchi, Elgot, Trakhtenbrot)

A language $L \subseteq \Sigma^*$ is regular if and only if the corresponding class of string structures is definable in MSO.

Monadic Second-Order Logic

Monadic Second-Order Logic (MSO) is the extension of first-order logic FO that allows quantification not only over the elements of a structure, but also over sets of elements.

Theorem (Büchi, Elgot, Trakhtenbrot)
A language $L \subseteq \Sigma^*$ is regular if and only if the corresponding class of string structures is definable in MSO.

Goal
Learning algorithms for MSO-definable hypotheses.

Monadic Second-Order Logic

Monadic Second-Order Logic (MSO) is the extension of first-order logic FO that allows quantification not only over the elements of a structure, but also over sets of elements.

Theorem (Büchi, Elgot, Trakhtenbrot)

A language $L \subseteq \Sigma^*$ is regular if and only if the corresponding class of string structures is definable in MSO.

Goal

Learning algorithms for MSO-definable hypotheses.

Bummer

Previous theorem shows that learning MSO (even full FO) is not possible in sublinear time.

Local Access is too weak

If we can only access the neighbours of a position, we may end up seeing nothing relevant.

Local Access is too weak

If we can only access the neighbours of a position, we may end up seeing nothing relevant.

Example

$$\ldots b aaaaaaaaaaaaaaaaaaaa\underline{a}aaaaaaaaaaaaaaaaaaa c \ldots$$

Local Access is too weak

If we can only access the neighbours of a position, we may end up seeing nothing relevant.

Example

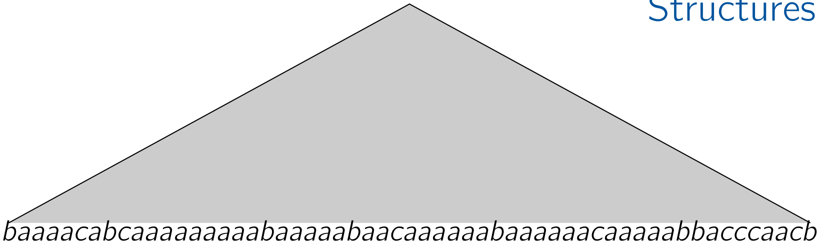
$$\ldots baaaaaaaaaaaaaaaaaaaa\underline{a}aaaaaaaaaaaaaaaaaaac \ldots$$

Solution: Index on Background Structure

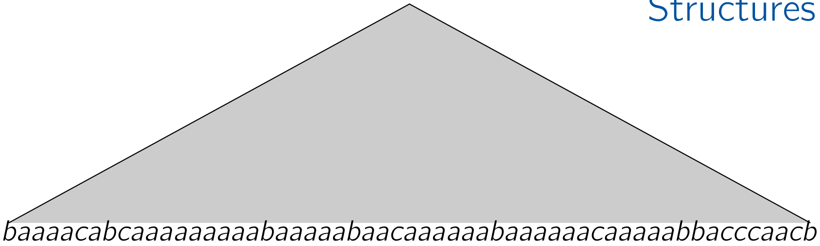
We can resolve this by building an index data structure over the background string.

We do this is a pre-processing phase where we only have access to the background structure, but not yet the training examples.

Factorisation Trees as Index Data Structures



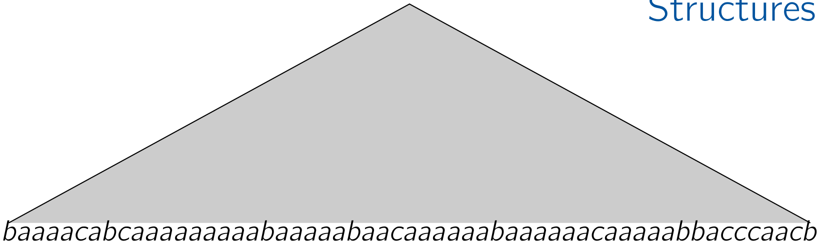
Factorisation Trees as Index Data Structures



A factorisation tree for a string B is an (ordered, unranked) tree whose

- leaves are labelled by the letters of the string,

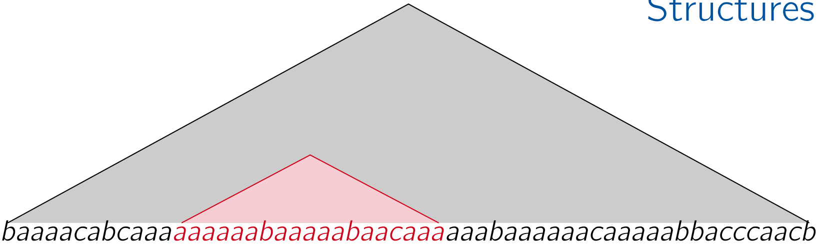
Factorisation Trees as Index Data Structures



A factorisation tree for a string B is an (ordered, unranked) tree whose

- leaves are labelled by the letters of the string,
- inner nodes are labelled by the MSO-type (of quantifier rank q) of the string "below" these nodes.

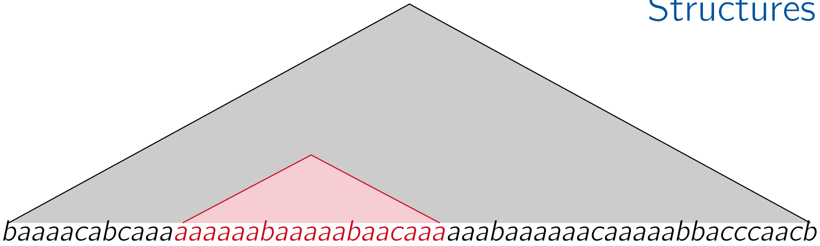
Factorisation Trees as Index Data Structures



A factorisation tree for a string B is an (ordered, unranked) tree whose

- leaves are labelled by the letters of the string,
- inner nodes are labelled by the MSO-type (of quantifier rank q) of the string “below” these nodes.

Factorisation Trees as Index Data Structures



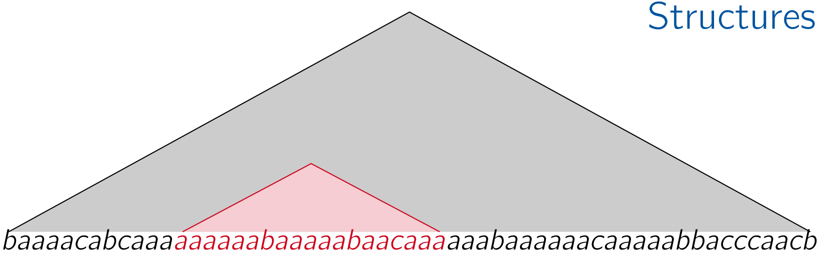
A factorisation tree for a string B is an (ordered, unranked) tree whose

- leaves are labelled by the letters of the string,
- inner nodes are labelled by the MSO-type (of quantifier rank q) of the string "below" these nodes.

Simons Factorisation Trees (Simon 1982)

We can construct a factorisation tree of constant height for a given string in linear time

Factorisation Trees as Index Data Structures



A factorisation tree for a string B is an (ordered, unranked) tree whose

- leaves are labelled by the letters of the string,
- inner nodes are labelled by the MSO-type (of quantifier rank q) of the string "below" these nodes.

Simons Factorisation Trees (Simon 1982)

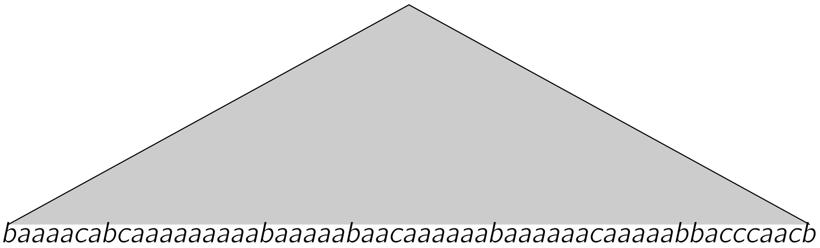
We can construct a factorisation tree of constant height for a given string in linear time (where the constant depends non-elementarily on the quantifier rank q).

Learning MSO

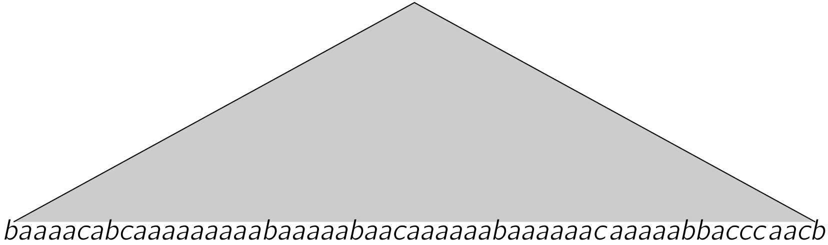
Theorem (G., Löding, Ritzert 2017)

There is a learner for MSO over strings with pre-processing time $O(n)$ *and learning time* $t^{O(1)}$.

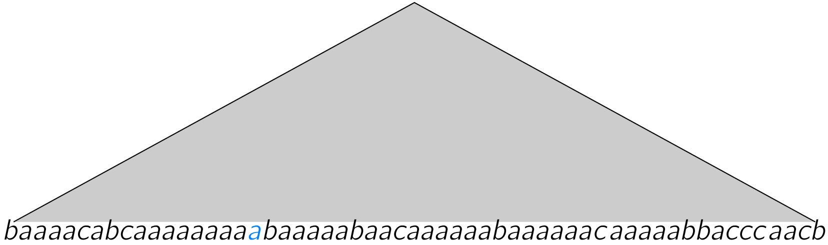
In the pre-processing phase, our algorithm builds a Simon factorisation tree for the background string B.



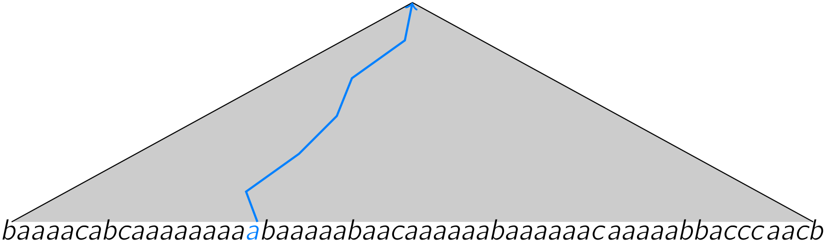
One by one, the training examples are incorporated into the factorisation tree.



One by one, the training examples are incorporated into the factorisation tree.



One by one, the training examples are incorporated into the factorisation tree.



To process a new example, we need to follow a path to the root an re-structure the tree along the way. The height of the tree may increase by an additive constant.

Learning Phase 1

One by one, the training examples are incorporated into the factorisation tree.

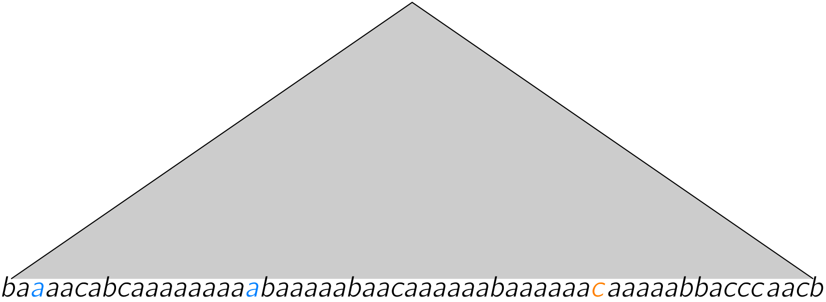
To process a new example, we need to follow a path to the root an re-structure the tree along the way. The height of the tree may increase by an additive constant.

Learning Phase 1

One by one, the training examples are incorporated into the factorisation tree.

To process a new example, we need to follow a path to the root an re-structure the tree along the way. The height of the tree may increase by an additive constant.

One by one, the training examples are incorporated into the factorisation tree.



To process a new example, we need to follow a path to the root an re-structure the tree along the way. The height of the tree may increase by an additive constant.

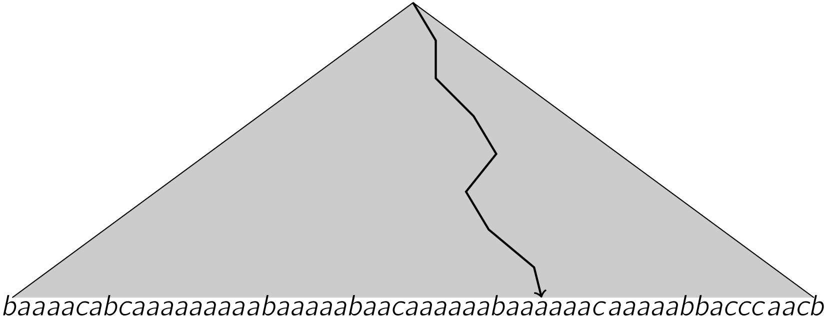
One by one, the training examples are incorporated into the factorisation tree.

To process a new example, we need to follow a path to the root an re-structure the tree along the way. The height of the tree may increase by an additive constant.

Learning Phase 2

To find a suitable choice of parameters, one has to process the tree in a top-down manner along branches from the root to the leaves (one branch per parameter).

To find a suitable choice of parameters, one has to process the tree in a top-down manner along branches from the root to the leaves (one branch per parameter).



Where do we go from here?

Open Problems

- Many technical questions are wide open: further classes of structures, other complexity measures, new logics…

Open Problems

- Many technical questions are wide open: further classes of structures, other complexity measures, new logics...
- What are suitable logics anyway?

Open Problems

- Many technical questions are wide open: further classes of structures, other complexity measures, new logics...
- What are suitable logics anyway?
- Go beyond Boolean classification.

Open Problems

- Many technical questions are wide open: further classes of structures, other complexity measures, new logics. . .
- What are suitable logics anyway?
- Go beyond Boolean classification.
- Can we design practical learning algorithms for our framework?

Vision

Design an data analysis system much like a databases system, providing an interface to “predictive queries” and for querying complex ML models (like ANNs).

References

- Martin Grohe and Gyorgy Turán.
  Learnability and Definability in Trees and Similar Structures.
  Theory of Computing Systems 37(1):193-220, 2004.
- Martin Grohe and Martin Ritzert.
  Learning first-order definable concepts over structures of small degree,
  arXiv:1701.05487 [cs.LG].
  Conference version in *Proceedings of the 32nd IEEE Symposium on Logic in Computer Science*, 2017.
- Martin Grohe, Christof Löding, and Martin Ritzert.
  Learning MSO-Definable Hypotheses on Strings,
  arXiv:1708.08081 [cs.LG].
  Conference version in *Proceedings of the 28th International Conference on Algorithmic Learning Theory*, 2017.