Theorie des Algorithmischen Lernens
Sommersemester 2006

Teil 2.1: Lernen formaler Sprachen: Standarderkennungstypen

Version 1.4

Gliederung der LV

Teil 1: Motivation

1. Was ist Lernen
2. Das Szenario der Induktiven Inf erenz
3. Natürlichkeitsanforderungen

Teil 2: Lernen formaler Sprachen

1. Grundlegende Begriffe und Erkennungstypen
2. Die Rolle des Hypothesenraums
3. Lernen von Patternsprachen
4. Inkrementelles Lernen

Teil 3: Lernen endlicher Automaten

Teil 4: Lernen berechenbarer Funktionen

1. Grundlegende Begriffe und Erkennungstypen
2. Reflexion

Teil 5: Informationsextraktion

1. Island Wrappers
2. Query Scenarios

7 Parameters of Inductive Inference

1. objects to be learned — formal languages
2. examples (syntax) — strings / pairs of strings and classification
3. examples (semantics, i.e. connection to object to be learnt) — correct and complete in the limit (text / informant)
4. learning device — computable devices
5. hypothesis space (syntax of hypotheses) — natural numbers
6. semantics of hypotheses — index in some enumeration
7. success criteria — convergence in the limit

Terms

- $\mathbb{N} = \{0, 1, 2, \ldots\}$ natural numbers
- $\langle \cdot, \cdot \rangle$: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ Cantor's pairing function
  - $\langle x, y \rangle = ((x+y)^2 + 3x + y)/2$
  - π_1: projection to first argument, i.e. $\pi_1(\langle x, y \rangle) = x$,
  - π_2: projection to second argument, i.e. $\pi_2(\langle x, y \rangle) = y$
  - canonically extended to arbitrary number of arguments
- $\circ$: concatenation of sequences

Formal Languages

- ***alphabet*** Σ: finite set
- $\boldsymbol{\Sigma^*}$, $\boldsymbol{\Sigma^+}$
- ***language*** over Σ: set of words over Σ
  i.e. $L \subseteq \Sigma^*$
  - empty, finite, infinite
  - Chomsky hierarchy: regular, context-free, context-sensitive languages
  - complement $\overline{L}$
  - sometimes identify language L with its characteristic function
    - i.e. $L(x) = +$ iff $x \in L$ and $L(x) = -$ iff $x \notin L$
- $\boldsymbol{a^n}$ means $\underbrace{a \ldots a}_{n \text{ times}}$
  - $a^0 = \varepsilon$
- $\boldsymbol{|w|}$ length of w
- $\tau \sqsubseteq \tau'$: τ is a prefix of τ'

Indexable classes

Definition 2.1.1: (Angluin 1980)
A class of non-empty languages $\mathcal{L}$ is said to be an ***indexable class with uniformly decidable membership*** (***indexable class***, for short) provided there are

- an effective enumeration $(L_j)_{j\in\mathbb{N}}$ of all and only the concepts in $\mathcal{L}$ and
- a recursive function f

such that, for all $j \in \mathbb{N}$ and all $x \in \Sigma^*$, the following holds:

$$f(j,x) = \begin{cases} 1, & \text{if } \quad x \in L_j, \\ 0, & \text{otherwise.} \end{cases}$$

$\mathcal{IC}$: set of all indexable classes

Examples for indexable classes:

- context-sensitive languages, context-free languages, regular languages, and of all pattern languages
- can be extended to arbitrary concept classes
  - use arbitrary ***learning domain*** $\mathcal{X}$ instead of Σ^*; ***concepts*** are subsets of $\mathcal{X}$.
  - $\mathcal{X}$ = set of all n-bit Boolean vectors: monomials, k-CNF, k-DNF, and k-decision lists are indexable classes of recursive concepts

Pattern Languages

alphabet Σ and enumerable set X of ***variables***, $\Sigma \cap X = \emptyset$

a ***pattern*** is a string $\pi \in (\Sigma \cup X)^+$

a ***(non-erasing) substitution*** σ is a mapping from $X \to \Sigma^+$

- Canonically extend substitutions to patterns

$\boldsymbol{L(\pi)} = \{w \mid w \in \Sigma^+$ and there exists a substitution σ such that $\sigma(\pi) = w\}$

pattern language: language describable by a pattern

PAT: set of all pattern languages

PAT $\in \mathcal{IC}$

Text

Definition 2.1.2:
Let L be language and $t = (x_n)_{n \in \mathbb{N}}$ be an ***infinite*** sequence of elements from Σ^* such that

- $\{x_n \mid n \in \mathbb{N}\} = L$.

Then, t is said to be a ***positive presentation*** or, synonymously, a ***text*** for L.

- ***Text***(L): set of all texts for L.
- t_y: initial segment of length y of text t
  - ***SegText***(L): set of all finite initial segments of texts for L.
  - ***SegText***$(\mathcal{L})$ $= \bigcup_{L \in \mathcal{L}}$ *SegText*(L).
- t_y^+: set of all words contained in t_y

Informant

Definition 2.1.3:
Let L be language and $i = ((x_n, b_n))_{n\in\mathbb{N}}$ be any infinite sequence of elements from $\Sigma^* \times \{+,-\}$ such that

- $\{x_n \mid n \in \mathbb{N},\ b_n = +\} = L$, and
- $\{x_n \mid n \in \mathbb{N},\ b_n = -\} = \overline{L}$.

Then, i is said to be a ***complete presentation*** or, synonymously, an ***informant*** for L.

- ***Info***$(\boldsymbol{L})$: set of all informants for L.
- $\boldsymbol{i_y}$: initial segment of length y of informant i
  - ***SegInfo***$(\boldsymbol{L})$: set of all finite initial segments of informants for L.
  - ***SegInfo***$(\boldsymbol{\mathcal{L}}) = \bigcup_{L\in\mathcal{L}}$ *SegInfo*(L).
- ***content***$(\boldsymbol{i_y})$: set of all words contained in i_y
- $\boldsymbol{i_y^+}$ and $\boldsymbol{i_y^-}$: sets of all positive and all negative words contained in i_y, i.e.
  $i_y^+ = \{x_j \mid j \le y,\ b_j = +\}$ and $i_y^- = \{x_j \mid j \le y,\ b_j = -\}$

Special Types of Text/Informant

- assume lexicographic order of strings in Σ^*: $(w_j)_{j \in \mathbb{N}}$
- ***lexicographically ordered text***: all strings appear in lexicographic order exactly once
  - exist only for infinite languages
- ***canonical text*** $t = (x_n)_{n \in \mathbb{N}}$
  - search the lexicographic smallest $w \in L$
  - set $x_0 = w$
  - for any $j > 0$: if $w_j \in L$ set $x_j = w_j$, otherwise set $x_j = x_{j-1}$
- for informant both terms ***lexicographically ordered informant*** and ***canonical informant*** coincide

Inductive Inference Machines

An ***inductive inference machine*** (abbr. ***IIM***) for some indexable class $\mathcal{L}$ is

- a total computable mapping from *SegText*$(\mathcal{L})/$*SegInfo*$(\mathcal{L})$ to $\mathbb{N} \cup \{?\}$.

the numbers output by an IIM M are interpreted with respect to a ***hypothesis space*** $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$, i.e. when M outputs some j, hypothesizes h_j

the output "?" means "don't have enough information"

Convergence

Definition 2.1.4:
Let $h = (h_j)_{j \in \mathbb{N}}$ be an infinite sequence.
We say that h ***converges in the limit*** to x iff all but finitely many terms of it are equal to x.

- This means, there exists an m such that for every $n \geq m$ it holds $h_n = x$.
- Notion $\boldsymbol{\lim h = x}$

Learning in the Limit

Definition 2.1.5:
Let $\mathcal{L} \in \mathcal{IC}$, let $L \in \mathcal{L}$ be a language, and let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space.
An IIM M ***LimTxt*** $_{\mathcal{H}}$–identifies L iff,

- for every $t \in$ *Text*(L)
  - there is a $j \in \mathbb{N}$ with $h_j = L$

such that

- the sequence $(M(t_y))_{y\in\mathbb{N}}$ converges to j.

M ***LimTxt*** $_{\mathcal{H}}$–identifies $\mathcal{L}$ iff M *LimTxt* $_{\mathcal{H}}$–identifies each $L' \in \mathcal{L}$.

LimTxt denotes the collection of all classes $\mathcal{L}' \in \mathcal{IC}$ for which there are a hypothesis space $\mathcal{H}' = (h'_j)_{j\in\mathbb{N}}$ and an IIM M' that *LimTxt* $_{\mathcal{H}'}$–identifies $\mathcal{L}'$.

LimTxt $_{\mathcal{H}}(M)$: set of all languages that are *LimTxt* $_{\mathcal{H}}$–identified by M

Learning in the Limit

Definition 2.1.6:
Let $\mathcal{L} \in \mathcal{IC}$, let $L \in \mathcal{L}$ be a language, and let $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$ be a hypothesis space.
An IIM M ***LimInf***$_{\mathcal{H}}$–identifies L iff,

- for every $i \in$ *Info*(L)
  - there is a $j \in \mathbb{N}$ with $h_j = L$

such that

- the sequence $(M(i_y))_{y \in \mathbb{N}}$ converges to j.

M ***LimInf***$_{\mathcal{H}}$–identifies $\mathcal{L}$ iff M *LimInf*$_{\mathcal{H}}$–identifies each $L' \in \mathcal{L}$.

LimInf denotes the collection of all classes $\mathcal{L}' \in \mathcal{IC}$ for which there are a hypothesis space $\mathcal{H}' = (h'_j)_{j \in \mathbb{N}}$ and an IIM M' that *LimInf*$_{\mathcal{H}'}$–identifies $\mathcal{L}'$.

LimInf$_{\mathcal{H}}(M)$: set of all languages that are *LimInf*$_{\mathcal{H}}$–identified by M

Learning of indexable class

When we have to learn an indexable class $\mathcal{L} = (L_j)_{j\in\mathbb{N}}$, we can choose the hypothesis space as follows:

1. use $\mathcal{L} = (L_j)_{j\in\mathbb{N}}$ as hypothesis space: ***exact*** identification
2. use another enumeration of $\mathcal{L} = (L_j)_{j\in\mathbb{N}}$ as hypothesis space: ***class-preserving*** identification
3. use another indexable class $\mathcal{L}' = (L'_j)_{j\in\mathbb{N}}$ as hypothesis space that contains each L_j: ***class-comprising*** identification

$\longrightarrow$ currently we consider class-comprising learning

- the other variants will be discussed later

Learning in the Limit

Theorem 2.1.1:
LimInf $= \mathcal{IC}$

Proof.

Identification by enumeration

Theorem 2.1.2:
LimTxt $\subset$ *LimInf*

Proof.

Consider class $\mathcal{L}_{sf}$:

- $L_0 = \{a^n \mid n \in \mathbb{N}\}$
- $L_{i+1} = \{a, \ldots, a^{i+1}\}$ for all $i \in \mathbb{N}$

$\mathcal{L}_{sf} \notin$ *LimTxt*

Learning in the Limit

Theorem 2.1.3:
$PAT \in LimTxt$

Sketch of proof.

1. *PAT* is enumerable
2. consistency is decidable
3. for any example set, there are only *finitely* many consistent hypotheses (apart from variable renamings)
4. overgeneralisation can be avoided

Consistent Learning

Definition 2.1.7:
Let $\mathcal{L} \in \mathcal{IC}$, let $L \in \mathcal{L}$ be a language, and let $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$ be a hypothesis space. An IIM M ***ConsTxt***$_{\mathcal{H}}$ / ***ConsInf***$_{\mathcal{H}}$–identifies L iff,

- for every $s \in$ *Text*(L) / $s \in$ *Info*(L)
- there is a $j \in \mathbb{N}$ with $h_j = L$

such that

- the sequence $(M(s_y))_{y \in \mathbb{N}}$ converges to j and
- every hypothesis is consistent, i.e.
  - (text) for each $x \in s_y^+$ it holds $x \in h_{M(s_y)}$
  - (informant) for each $x \in s_y^+$ it holds $x \in h_{M(s_y)}$ and for each $x \in s_y^-$ it holds $x \notin h_{M(s_y)}$

ConsTxt$_{\mathcal{H}}(M)$, *ConsInf*$_{\mathcal{H}}(M)$, ***ConsTxt***, ***ConsInf*** are defined analogously to *LimTxt*…

Consistent Learning

Observation:

- consistency is uniformly decidable in indexable classes

$\longrightarrow$ every hypothesis can be made consistent

Corollary 2.1.4:
ConsInf $=$ *LimInf*
ConsTxt $=$ *LimTxt*

Proof.

Informant: Identification by enumeration works consistently

Text: Let M be an IIM. For any t_y, ***pad*** the hypothesis $M(t_y)$ with t_y, i.e. add all strings $w \in t_y^+$ to $h_{M(t_y)}$.

Consistency is no restriction for learning indexable classes!

Finite Learning

Definition 2.1.8:
Let $\mathcal{L} \in \mathcal{IC}$, let $L \in \mathcal{L}$ be a language, and let $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$ be a hypothesis space.
An IIM M ***FinTxt***$_{\mathcal{H}}$ / ***FinInf***$_{\mathcal{H}}$–identifies L iff,

- for every $s \in$ *Text*(L) / $s \in$ *Info*(L)
- there is a $j \in \mathbb{N}$ with $h_j = L$

such that

- there is exactly one index m in the sequence $(M(s_y))_{y \in \mathbb{N}}$ with $M(s_m) \in \mathbb{N}$
  (all other hypotheses are "?")

  and
- $M(s_m) = j$

Finite Learning

Corollary 2.1.5:

FinTxt $\subseteq$ *LimTxt*
FinInf $\subseteq$ *LimInf*

Proof.

Exercise.

Consider the following alternative definition:

Definition 2.1.9:

An IIM M ***FinTxt***$_{\mathcal{H}}$ / ***FinInf***$_{\mathcal{H}}$–identifies L iff,

- for every $s \in$ *Text*(L) / $s \in$ *Info*(L)
- there is a $j \in \mathbb{N}$ with $h_j = L$

such that

- the sequence $(M(s_y))_{y \in \mathbb{N}}$ converges to j and
- if $M(s_y) = M(s_{y+1})$ then $M(s_{y+1}) = M(s_{y+2})$

both are equivalent $\longrightarrow$ Exercise

Finite Learning

Theorem 2.1.6:
$\mathit{LimTxt} \setminus \mathit{FinInf} \neq \emptyset$

Proof.

Consider the set $\mathcal{L}_{fin}$ of all finite languages. Clearly $\mathcal{L}_{fin} \in \mathit{LimTxt} \setminus \mathit{FinInf}$.

Corollary 2.1.7:
$\mathit{FinTxt} \subset \mathit{LimTxt}$
$\mathit{FinInf} \subset \mathit{LimInf}$

Finite Learning: Characterization Info

Can we find a characterization for *Fin*-learnability?

Definition 2.1.10:
An indexable class $\mathcal{L} = (L_j)_{j \in \mathbb{N}}$ of languages is said to be ***discrete*** iff

- for every $j \in \mathbb{N}$ there exists a *finite* set $D_j \subseteq \Sigma^*$
  - such that for every $j' \in \mathbb{N}$ with $L_j \neq L_{j'}$ it holds that
    - there is an $x \in D_j$ with $L_j(x) \neq L_{j'}(x)$.

$\mathcal{L}$ is said to be ***effectively discrete*** iff there is a computable function $f : \mathbb{N} \to \wp(\Sigma^*)$ such that $f(j) = D_j$, for every $j \in \mathbb{N}$.

Theorem 2.1.8:
$\mathcal{L} \in$ *FinInf* iff $\mathcal{L}$ is effectively discrete.

Finite Learning: Characterization Info

Proof. Suffiency:

Use $\mathcal{L} = (L_j)_{j\in\mathbb{N}}$ as hypothesis space, i.e. set $h_j = L_j$, for all $j \in \mathbb{N}$.

$M(i_x)$:
If $x = 0$ or $M(t_{x-1}) =$ "?", goto (*). Otherwise output $M(i_{x-1})$.
(*) For $j = 0, 1, \ldots, x$, generate $D_j = f(j)$ and test whether $D_j \subseteq$ *content*(i_x) and $h_j(w) = i_x(w)$, for all $w \in D_j$.
If such a j has been found, output the minimal one. Otherwise output "?".

Verification. Let i be an informant for L_j.

1. M always outputs a hypothesis
2. there is an x such that $M(i_x) \in \mathbb{N}$
   - set $x = \max\{j, \hat{x}\}$, where $\hat{x}$ is the smallest x with $D_j \subseteq$ *content*(i_x)
3. $h_M(i_x) = L_j$ holds by properties of f

Finite Learning: Characterization Info

<u>Necessity:</u>

Let M be an IIM finitely learning $\mathcal{L}$. Define f as follows:

$f(j)$:
Search for the least $x \in \mathbb{N}$ such that $M(i_x) \in \mathbb{N}$, where i is the canonical informant for L_j. Output *content*(i_x).

Finite Learning: Characterization Text

Theorem 2.1.9:
$\mathcal{L} \in$ *FinTxt* iff there are an indexing $\mathcal{L} = (L_j)_{j\in\mathbb{N}}$ and a *recursively generable* family $(T_j)_{j\in\mathbb{N}}$ of *finite* sets such that

- for all $j \in \mathbb{N}$, $T_j \subseteq L_j$
- for all $j, z \in \mathbb{N}$, if $T_j \subseteq L_z$ then $L_j = L_z$

recursively generable: there is a total-computable function $f : \mathbb{N} \longrightarrow \wp(\Sigma^*)$ such that $f(j) = T_j$, for every $j \in \mathbb{N}$.

Finite Learning: Characterization Text

Proof. Suffiency:

$M(t_x)$:
If $x = 0$ or $M(t_{x-1}) =$ "?", goto (*). Otherwise output $M(t_{x-1})$.
(*) For $j = 0, 1, \ldots, x$, generate T_j and test whether $T_j \subseteq t_x^+$ and $t_x^+ \subseteq h_j$.
If such a j has been found, output the minimal one. Otherwise output "?".

Verification of correctness $\longrightarrow$ Exercise

Necessity:

Let $j \in \mathbb{N}$ and let t be the canonical text for L_j.

We let x be the smallest number such that $M(t_x) \in \mathbb{N}$. We set $T_j = t_x^+$.

Verification of correctness $\longrightarrow$ Exercise

Conservative Learning

Definition 2.1.11:
Let $\mathcal{L} \in \mathcal{IC}$, let $L \in \mathcal{L}$ be a language, and let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space. An IIM M ***ConsvTxt$_\mathcal{H}$*** / ***ConsvInf$_\mathcal{H}$***–identifies L iff,

- for every $s \in$ *Text*(L) / $s \in$ *Info*(L)
- there is a $j \in \mathbb{N}$ with $h_j = L$

such that

- the sequence $(M(s_y))_{y\in\mathbb{N}}$ converges to j and
- for any two consecutive hypotheses $k = M(s_y)$ and $j = M(s_{y+1})$:
  - if $k \in \mathbb{N}$ and $k \neq j$, then h_k is inconsistent with s_{y+1}

conservative learning must be done *without overgeneralisation* (a hypothesis j is ***overgeneralized*** if $h_j \supset L$)

Gold 67: *The problem with text is that, if you guess too large a language, the text will never tell you that you are wrong.*

Conservative Learning: Characterization Info

Corollary 2.1.10:
ConsvInf $=$ *LimInf*.

Proof.

Identification by enumeration works conservativley

Conservative Learning: Characterization Text

Theorem 2.1.11:
$\mathcal{L} \in$ *ConsvTxt* iff there are a hypothesis space $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ and a *recursively generable* family $(T_j)_{j\in\mathbb{N}}$ of *finite* sets such that

- $\mathcal{H}$ contains all languages from $\mathcal{L}$
- for all $j \in \mathbb{N}$, $T_j \subseteq h_j$
- for all $j, z \in \mathbb{N}$, if $T_j \subseteq h_z$ then $h_z \not\subset h_j$

important concept: the sets T_j are called ***Telltales***

Conservative Learning: Characterization Text

Example 1:

- set of all finite languages on $\Sigma = \{a, b, c\}$:
  $\longrightarrow$ telltale for L is L
- $L_j = \Sigma^* \setminus \{a^j\}$
  $\longrightarrow$ $T_j = \{a^{j+1}\}$
- $\mathcal{L}_{\textit{sf}}$: $L_0 = \{a^n \mid n \in \mathbb{N}\}$; $L_{i+1} = \{a, \ldots, a^{i+1}\}$
  $\longrightarrow$ There is no telltale for L_0

Conservative Learning: Characterization Text

Proof. <u>Suffiency:</u>

$M(t_x)$:
If $x = 0$ or $M(t_{x-1}) =$ "?", goto (B). Otherwise, goto (A).
(A) Let $j = M(t_{x-1})$. Test whether or not $t_x^+ \subseteq h_j$. In case it is, output j. Otherwise, goto (B).
(B) For $j = 0, 1, \ldots, x$, generate T_j and test whether or not $T_j \subseteq t_x^+ \subseteq h_j$.
If such a j has been found, output the minimal one. Otherwise output "?".

Verification. Let t be a text for some $L \in \mathcal{L}$.

1. M always outputs a hypothesis
2. M converges on t
   - let k be the minimal index of L in $\mathcal{H}$
   - there must be an $\hat{x}$ such that $T_k \subseteq t_{\hat{x}}^+$
   - after point $\max\{k, \hat{x}\}$, M outputs a number
     - which is $\leq k$
   - M only changes the hypothesis in case of inconsistencies

Conservative Learning: Characterization Text

3. if M converges (say to j), then $h_j = L$

   Suppose the converse

   case "$L \setminus h_j \neq \emptyset$":

   the string w with $w \in L \setminus h_j$ will appear and M will change its hypothesis - a contradiction

   case "$h_j \setminus L \neq \emptyset$":

   - may assume $L \subset h_j$ (otherwise we are in the former case)
   - for $x \geq \hat{x}$, $T_j \subseteq t_x^+$
   - since $t_x^+ \subseteq L$ this implies $T_j \subseteq L$
   - by property 3 of T_j this implies $L \not\subset h_j$ - a contradiction

Remark:
In fact, M not only *ConsvTxt*$_{\mathcal{H}}$-identifies $\mathcal{L}$, but the potentially larger set $\mathcal{H}$.

Conservative Learning: Characterization Text

Necessity:

Let M *ConsvTxt*$_{\mathcal{H}}$-identify $\mathcal{L} = (L_j)_{j \in \mathbb{N}}$.

We first use an auxiliary construction $\tilde{\mathcal{H}} = (\tilde{h}_j)_{j \in \mathbb{N}}$ and $(\tilde{T}_j)_{j \in \mathbb{N}}$:

For each $k, x \in \mathbb{N}$, set $\tilde{h}_{\langle k,x \rangle} = h_k$. Define $(\tilde{T}_j)_{j \in \mathbb{N}}$ as follows:

- Determine k, x with $j = \langle k, x \rangle$.
  Let t be the canonical text for h_k.
- Determine the least $r \leq x$ such that $t_x = t'_x$, where t' is the canonical text for L_r.
  If no such r exists, set $\tilde{T}_j = \emptyset$.
- Determine the least $y \leq x$ such that $M(t_y) = k$.
  If y has been found, set $\tilde{T}_j = t_y^+$, otherwise set $\tilde{T}_j = \emptyset$.

Conservative Learning: Characterization Text

$\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ and $(T_j)_{j\in\mathbb{N}}$ are now destilled from $\tilde{\mathcal{H}} = (\tilde{h}_j)_{j\in\mathbb{N}}$ and $(\tilde{T}_j)_{j\in\mathbb{N}}$ by simply deleting all entries j with $\tilde{T}_j = \emptyset$.

Analysis

- $(T_j)_{j\in\mathbb{N}}$ is a recursively generable family of finite sets
- condition 1 holds: for each $L \in \mathcal{L}$, there is an index j with $L = h_j$
  - there is a k with $h_k = L$ and M converges to k when feeding the canonical text t for L (say convergence happens at t_y)
  - there is a smallest index r with $L = L_r$
  - then, for $x = \max\{y, r\}$, $\tilde{T}_{\langle k,x\rangle} = t_y^+ \neq \emptyset$ and $\tilde{h}_{\langle k,x\rangle} = L$
- condition 2 holds by definition
- *verification of condition 3 is more involved, we skip it here (can be found in [2])*

qed

Set-Driven Learning

Definition 2.1.12:
Let M be an IIM.
M works ***rearrangement-independent*** iff for every texts t, t' and every $y \in \mathbb{N}$, $t_y^+ = t'^+_y$ implies $M(t_y) = M(t'_y)$.

Definition 2.1.13:
Let M be an IIM.
M works ***set-driven*** iff for every texts t, t' and every $y, y' \in \mathbb{N}$, $t_y^+ = t'^+_{y'}$ implies $M(t_y) = M(t'_{y'})$.
Corresponding identification type: *sd-LimTxt*

set-driven IIMs only consider the *content*, where rearrangement-independent IIMs also can take the step number into account

Set-Driven Learning

Theorem 2.1.12:
sd-LimTxt $=$ *ConsvTxt*

Sketch of proof.

ConsvTxt $\subseteq$ *sd-LimTxt*:

$\mathcal{L} \in$ *ConsvTxt* implies existence of $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$ and recursively generable telltale-sets $(T_j)_{j \in \mathbb{N}}$.

$M(t_x)$:
For $j = 0, 1, \ldots,$ *card*(t_x^+), generate T_j and test whether or not $T_j \subseteq t_x^+ \subseteq h_j$.
If such a j has been found, output the minimal one. Otherwise output a hypothesis for t_x^+.

Set-Driven Learning

Analysis:

- M works set-driven
- M correctly learns $\mathcal{L}$:

  case "L is finite": Consider the hypothesis computed when L is completely contained in t_x, i.e. $t_x^+ = L$
  - the hypothesis is computed by the "otherwise"-statement: correct by definition
  - a j has been found with $T_j \subseteq t_x^+ \subseteq h_j$: $T_j \subseteq L \subseteq h_j$ implies $L = h_j$

  case "L is infinite": argumentation "as usual"

Remark: with a slight modification, M can be made conservative: use $\cup_{n \leq j} T_n \cap h_j$ instead of T_j

---

ConsvTxt $\subseteq$ *sd-LimTxt*:

skipped (see [2]) *qed*

Learning in the Limit: Characterization Text

Theorem 2.1.13:
$\mathcal{L} \in$ *LimTxt* iff there is an indexing $(L_j)_{j \in \mathbb{N}}$ of $\mathcal{L}$ and a *recursively enumerable* family $(T_j)_{j \in \mathbb{N}}$ of *finite* sets such that

- for all $j \in \mathbb{N}$, $T_j \subseteq L_j$
- for all $j, z \in \mathbb{N}$, if $T_j \subseteq L_z$ then $L_z \not\subset L_j$

recursively enumerable means there exists a recursive function $f : \mathbb{N} \times \mathbb{N} \to \wp(\Sigma^*)$ such that $\bigcup_{n \in \mathbb{N}} f(j, n) = T_j$, for every $j \in \mathbb{N}$.

Learning in the Limit: Characterization Text

Proof. Suffiency:

Notation: T_j^x: result of running the generation of T_j for x time steps

$M(t_x)$:
Search for the least $j \leq x$ with $T_j^x \subseteq t_x^+ \subseteq L_j$.
If j has been found, output it; otherwise output "?".

Verification. Let t be a text for some $L \in \mathcal{L}$.

1. M always outputs a hypothesis
2. M converges on t
   - let k be the minimal index of L in $\mathcal{L}$
   - let l be the time after which $T_0, T_1, \ldots T_k$ are completely enumerated
   - let x be the time so that all elements of $T_0, T_1, \ldots T_k$ (if they belong to L) are contained in t_x, i.e. $\left(L \cap \bigcup_{j=0,\ldots,k} T_j\right) \subseteq t_x$
   - after point $\hat{x} = \max\{k, l, x\}$, M outputs a number $\leq k$
     - once a value j has been rejected by M, it will never be output

Learning in the Limit: Characterization Text

3. if M converges (say to j), then $L_j = L$

   Suppose the converse

   case "$L \setminus L_j \neq \emptyset$":

   the string w with $w \in L \setminus L_j$ will appear and M will change its hypothesis - a contradiction

   case "$L_j \setminus L \neq \emptyset$":

   - may assume $L \subset L_j$ (otherwise we are in the former case)
   - for $x \geq \hat{x}$, $T_j \subseteq t_x^+$
   - since $t_x^+ \subseteq L$ this implies $T_j \subseteq L$
   - by property 3 of T_j this implies $L \not\subset L_j$ - a contradiction

Learning in the Limit: Characterization Text

Necessity:

Generator for T_j:
Let $s_0, s_1, \ldots$ be the canonical text for L_j and $(\sigma_j)_{j\in\mathbb{N}}$ be an enumeration of *SegText*(L_j) (i.e. of all finite sequences of strings from L_j).

Stage 0: Set $\tau = s_0$ and $T_j = \tau^+$.

Stage $n > 0$: Search for the least j such that $M(\tau) \neq M(\tau \circ \sigma_j)$.
If such a j has been found, set $\tau = \tau \circ \sigma_j \circ s_n$ and $T_j = \tau^+$.

($f(j,n)$ can be defined by letting the generator for T_j run n steps and output the current value of T_j.)

Learning in the Limit: Characterization Text

Analysis

- obvious: algorithm enumerates only strings from L_j, i.e. $T_j \subseteq L_j$ holds
- to show: T_j is finite
  - assume the contrary, i.e. T_j contains infinitely many elements

  $\longrightarrow$ every stage is left

  $\longrightarrow$ in the limit, the τ form a text for L_j (lets call it t)
    - t contains only strings from L_j
    - all strings from L_j are contained in t, since $s_0, s_1, \ldots$ is the canonical text for L_j

  $\longrightarrow$ but: M changes its hypothesis infitely often!
  - Basic Idea: hunting for a ***stabilizing sequence***:

> **Definition 2.1.14**:
> A finite sequence τ is a ***stabilizing sequence*** for L w.r.t. M iff
> * $\tau^+ \subseteq L$
> * $\forall \tau' \in$ *SegText*(L): if $\tau \sqsubseteq \tau'$, then $M(\tau) = M(\tau')$.

Learning in the Limit: Characterization Text

remains to show: for all $j, z \in \mathbb{N}$, if $T_j \subseteq L_z$ then $L_z \not\subset L_j$

- assume the contrary, i.e. there are $j, z \in \mathbb{N}$ with $T_j \subseteq L_z$ and $L_z \subset L_j$
- let τ be the one computed in the last stage which terminated
- let t' be a text for L_z starting with τ
- consider $t'_{|\tau|}, t'_{|\tau|+1}, t'_{|\tau|+2} \cdots$

$\longrightarrow$ by construction, $M(t'_{|\tau|}) = M(t'_{|\tau|+1}) = M(t'_{|\tau|+2}) = \cdots = M(\tau)$

$\longrightarrow$ M converges on t and t' to the same hypothesis, but both are texts for two different languages

$\longrightarrow$ M fails to identify at least one of L_j and L_z!

qed

Stabilizing Sequences

the last proof also shows the following insight

Lemma 2.1.14:
For any IIM M *LimTxt*-learning L, there is a stabilizing sequence for L w.r.t. M.

in fact, it proves an even stronger insight:

Lemma 2.1.15:
For any IIM M *LimTxt*-learning L and any $\tau \in$ *SegText*(L), there is a stabilizing sequence τ' for L w.r.t. M extending τ (i.e. $\tau \sqsubseteq \tau'$).

Conservative vs. Learning in the Limit

Theorem 2.1.16:
$ConsvTxt \subset LimTxt$

Excursion: Blum Complexity Measures

φ: acceptable numbering (Gödelnumbering) of all computable 1ary functions on $\mathbb{N}$

- for all $f \in \mathcal{P}$ there exists a $j \in \mathbb{N}$ such that $\varphi(j, x) = f(x)$, for all $x \in \mathbb{N}$
- ... *(some additional constraints)*

Notations:

- $\boldsymbol{\varphi_j(x)}$ instead of $\varphi(j, x)$
- $\boldsymbol{\varphi_j}$: the function f with $f(x) = \varphi(j, x)$
- $\varphi_j(x)\downarrow$: computation of $\varphi_j(x)$ terminates
- $\varphi_j(x)\uparrow$: computation of $\varphi_j(x)$ does not terminate

Definition 2.1.15:
A ***Blum complexity measure*** ϕ is a 2ary computable function $\mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ with the following properties:

- $\phi_j(x)\downarrow$ iff $\varphi_j(x)\downarrow$
- for each $j, x, k \in \mathbb{N}$ it is decidable whether $\phi_j(x) \leq k$ holds

Excursion: Blum Complexity Measures

Example 2:

the following methods are Blum complexity measures:

- time in seconds of computation on some fixed machine
- time in clock cycles of computation on some fixed machine
- number of branches executed when running a program

Conservative vs. Learning in the Limit

Proof.

define $\mathcal{L}_{\mathit{consv}}$ as follows:

- for all $k \in \mathbb{N}$, $L_k = \{a^k b^z \mid z \in \mathbb{N}\}$
- for all k with $\varphi_k(k) \downarrow$ and all $j \in \mathbb{N}$ with $j \leq \phi_k(k)$, $L_{k,j} = \{a^k b^z \mid z \leq j\}$

Exercise: Specify an IIM that learns $\mathcal{L}_{\mathit{consv}}$ in the limit.

Conservative vs. Learning in the Limit

$\mathcal{L}_{\mathit{consv}} \in \mathit{LimTxt}$

Define hypothesis space $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$ and telltale sets as follows:

$$h_{\langle k,x\rangle} = \begin{cases} L_{k,j} & : \text{if } x = \phi_k(k) + j \text{ for some } j \leq \phi_k(k) \\ L_k & : \text{otherwise} \end{cases}$$

$$T_{\langle k,x\rangle} = \begin{cases} L_{k,j} & : \text{if } x = \phi_k(k) + j \text{ for some } j \leq \phi_k(k) \\ \{a^k b, a^k b^{\phi_k(k)+1}\} & : \text{if } \varphi_k(k)\downarrow \text{ and } (x < \phi_k(k) \text{ or } x > 2\phi_k(k)) \\ \{a^k b\} & : \text{otherwise} \end{cases}$$

Exercise: argue why $T_{\langle k,x\rangle}$ is enumerable

verification, that $T_{\langle k,x\rangle}$ is really a telltale set for $L_{\langle k,x\rangle}$:

case "1. $\varphi_k(k)\uparrow$": for all x, $h_{\langle k,x\rangle} = L_k$ and $T_{\langle k,x\rangle} = \{a^k b\} \ldots$

case "2. $\varphi_k(k)\downarrow$":

- if $h_{\langle k,x\rangle} = L_k$, $T_{\langle k,x\rangle} = \{a^k b, a^k b^{\phi_k(k)+1}\}$, which is not contained in any $h_{\langle k,x'\rangle}$ with $h_{\langle k,x'\rangle} \neq L_k$
- if $h_{\langle k,x\rangle} = L_{k,x}$, $T_{\langle k,x\rangle} = L_{k,x}$, therefore $T_{\langle k,x\rangle} \subseteq L$ implies $h_{\langle k,x\rangle} \subseteq L$

Conservative vs. Learning in the Limit

$\mathcal{L}_{consv} \notin ConsvTxt$

Assume the contrary, i.e. let M *ConsvTxt*$_\mathcal{H}$-identify $\mathcal{L}_{consv}$. Let $(T_j)_{j\in\mathbb{N}}$ be a recursive family of finite telltale sets.

Then, the following procedure decides the *halting problem*:

On input k do:
Search for a j with the following property:

- $\{a^k b^r \mid r \leq m_j\} \cup \{a^k b^{m_j+1}\} \subseteq h_j$, where $m_j = \max\{r \mid a^k b^r \in T_j\}$

If $\phi_k(k) \leq m_j$ output 1, otherwise output 0.

Verification:

- L_k is contained in $\mathcal{H}$ (lets say $L_k = h_j$)
- so, $T_j \cup \{a^k b^{m_j+1}\} \subseteq L_k = h_j$ holds (whatever T_j is)
- hence, the procedure terminates, we next have to verify its correctness

Conservative vs. Learning in the Limit

The procedure outputs 1 iff $\varphi_k(k) \downarrow$:

case "procedure returns 1": obviously correct

case "procedure returns 0": Suppose that $\varphi_k(k) \downarrow$, lets say $\phi_k(k) = y$.

- Let j and m_j be the values found in the procedure.
- since the procedure returns 0, $m_j < \phi_k(k)$ holds
- consider the language $L_{k,m_j} = \{a^k b^z \mid z \leq m_j\}$
  - $L_{k,m_j} \in \mathcal{L}$
  - by construction, $T_j \subseteq L_{k,m_j} \subset h_j$, a contradiction

qed

The missing Relation

Theorem 2.1.17:
FinInf $\subset$ *ConsvTxt*

Proof.

Let M be an IIM *FinInf*$_{\mathcal{H}}$-identifying $\mathcal{L}$.

We define recursive telltale sets as follows.

For $j \in \mathbb{N}$, let$i$ be the canonical informant for h_j. We set $T_j = i_y^+$, where y is such that $M(i_y) \in \mathbb{N}$.

(If $T_j = \emptyset$ by this construction, we repair it and set $T_j = \{w\}$ for some $w \in h_j$).

Verification:

- obviously $T_j \subseteq h_j$, for all $j \in \mathbb{N}$
- for all $j, z \in \mathbb{N}$, if $T_j \subseteq h_z$ then $h_z \not\subset h_j$:
  - assume the contrary, i.e. let $T_j \subseteq h_z \subset h_j$
  - consider the canonical informants for h_j and h_z $\rightarrow$ are identical up to y
  - M fails to finitely identify h_z

Exercise: Provide a set $\mathcal{L} \in$ *ConsvTxt* $\setminus$ *FinInf*.

Summary

$$\mathit{ConsvInf} = \mathit{LimInf} = \mathcal{IC}$$

$$\cup$$

$$\mathit{LimTxt}$$

$$\cup$$

$$\mathit{ConsvTxt} = \textit{ri-LimTxt} = \textit{sd-LimTxt}$$

$$\cup$$

$$\mathit{FinInf}$$

$$\cup$$

$$\mathit{FinTxt}$$

Literature

[1] E Mark Gold: Language Identification in the Limit. *Information and Control* 14, pp. 447–474, 1967.

[2] Steffen Lange: Algorithmic Learning of Recursive Languages. Mensch-und-Buch-Verlag 2000.

[3] Thomas Zeugmann & Steffen Lange: A Guided Tour Across the Boundaries of Learning Recursive Languages. *In: Jantke & Lange (eds.) Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence 961, pp. 190–258, Springer-Verlag 1995.

Changelog

- V1.4:
  - Folie 51 $L_{k,m_{j+1}} \longrightarrow L_{k,m_j}$
- V1.3:
  - Folie 35/36 zusammengefaßt, Theorem gelöscht
  - Folie 38 $\cup \longrightarrow \cap$
  - Folie 40 $\mathcal{H} \longrightarrow \mathcal{L}$
  - Folie 49 ff. Fehler korrigiert
- V1.2:
  - Folie 10 *SegInfo*$(\mathcal{L})$ hinzugefügt
  - Folie 13 $t \longrightarrow i$
  - Folie 22 finite hinzugefügt
  - Folie 23 $\max\{\mathit{card}(D_j), \hat{x}\} \longrightarrow \max\{j, \hat{x}\}$
  - Folien 35-38 eingeschoben, ab Folie 51 neu
- V1.1:
  - Folie 4: Prefix hinzugefügt
  - ab Folie 14 neu