Theorie des Algorithmischen Lernens
Sommersemester 2007

Teil 4: Lernen rekursiver Funktionen

Version 1.0

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. zu lernende Objekte — totale berechenbare Funktionen
2. Beispiele (Syntax) — Paare von Ein-/Ausgabewerten
3. Beispiele (Semantik) — im Limes korrekte und vollständige Beschreibung der Funktion
4. Lernverfahren — berechenbare Funktion
5. Hypothesenraum (Syntax) — Natürliche Zahlen
6. Semantik von Hypothesen — Programm in festgelegter Programmiersprache
7. Erfolgskriterium — Konvergenz im Limes

Begriffe

$\mathbb{N}$ ... Menge der natürlichen Zahlen $\{0,1,2,\ldots\}$

$\mathcal{P}$ ... Menge aller berechenbaren (partiell-rekursiven) Funktionen

$\mathcal{P}^n$ ... Menge aller berechenbaren (partiell-rekursiven) n-stelligen Funktionen

$\mathcal{R}$ ... Menge aller überall definierten berechenbaren (allgemein-rekursiven, oder kurz ***rekursiven***) Funktionen

Gödelnumerierungen

Definition 4.1:
Eine Funktion $\varphi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ heißt ***Gödelnumerierung*** der einstelligen berechenbaren Funktionen gdw.

1. φ ist berechenbar, d.h. $\varphi \in \mathcal{P}^2$.
2. φ ist universell für $\mathcal{P}^1$, d.h. $\forall f \in \mathcal{P}^1 \; \exists j \in \mathbb{N} \; \forall x \in \mathbb{N} : \varphi(j, x) = f(x)$.
3. Jede Aufzählung berechenbarer Funktionen kann nach φ übersetzt werden, d.h. $\forall \psi \in \mathcal{P}^2 \; \exists c \in \mathcal{R} \; \forall x \in \mathbb{N} : \psi(i, x) = \varphi(c(i), x)$.

Schreibweise: $\varphi_i(x)$ statt $\varphi(i, x)$

Information

Definition 4.2:
Eine ***Informationsfolge*** σ für eine allgemein-rekursiven Funktion f ist eine Folge $\{(x_i, y_i)\}_{i \in \mathbb{N}}$ so daß folgendes gilt:

- $f(x_i) = y_i$
- jedes $x \in \mathbb{N}$ kommt in σ vor, d.h. $\{x_i \mid i \in \mathbb{N}\} = \mathbb{N}$

- Wir beschränken uns zunächst auf ***Standardreihenfolge***, d.h. $\sigma = (0, f(0)), (1, f(1)), (2, f(2)), \ldots$.
  - auch als ***Graph*** bezeichnet
- Anfangsstücke wie üblich
  - spezielle Notation für Graphen: $\boldsymbol{f[n]} = (0, f(0)), \ldots, (n, f(n))$
    - abgekürzt durch $f(0), f(1), \ldots, f(n)$

Lernen im Limes

analog zu formalen Sprachen

Definition 4.3:
Ein Lernverfahren $M \in \mathcal{P}$ ***lernt eine Funktion*** $f \in \mathcal{R}$ ***im Limes*** genau dann wenn:

1. M berechnet für jedes Anfangsstückvon f eine Hypothese,
   d.h. $\forall n \in \mathbb{N} : M(f[n]) \downarrow$
2. M konvergiert gegen eine korrekte Hypothese,
   d.h. $\exists h \in \mathbb{N} : \varphi_h = f$ und $\exists m \in \mathbb{N}\ \forall n \geq m : M(f[n]) = h$.

M lernt eine Klasse $U \subseteq \mathcal{R}$ im Limes gdw. es jede Funktion aus der Klasse lernt (Bezeichnung: $U \subseteq \mathit{LIM}(M)$).
Die Menge aller Klassen allgemein-rekursiver Funktionen, die von einem berechenbaren Lernverfahren gelernt werden, heißt *LIM*. Also,
$\mathit{LIM} = \{U \subseteq \mathcal{R} \mid \exists M \in \mathcal{P} : U \subseteq \mathit{LIM}(M)\}$.

Anmerkung: Wenn man formal ganz sauber sein will, muß man noch eine Kodierung von Anfangsstücke nach $\mathbb{N}$ definieren, da unsere Lernverfahren aus $\mathcal{P}$ sind, d.h. über nat. Zahlen arbeiten.

Finites Lernen

analog zu formalen Sprachen

Definition 4.4:
Ein Lernverfahren M lernt eine Funktion $f \in \mathcal{R}$ ***finit*** im Limes genau dann wenn es ein $d \in \mathcal{R}$ gibt, so daß:

1. M berechnet für jedes Anfangsstück von f eine Hypothese,
   d.h. $\forall n \in \mathbb{N} : M(f[n]) \downarrow$,
2. M konvergiert gegen eine korrekte Hypothese,
   d.h. $\exists h \in \mathbb{N} : \varphi_h = f$ und $\exists m \in \mathbb{N}\ \forall n \geq m : M(f[n]) = h$
3. *d zeigt die Endhypothese an*,
   d.h. $\forall n \in \mathbb{N} : d(f[n]) = 1 \Leftrightarrow M(f[n]) = h$.

Erkennungstyp: ***FIN***

Konsistentes Lernen

analog zu formalen Sprachen

Definition 4.5:
Ein Lernverfahren $M \in \mathcal{P}$ lernt eine Funktion $f \in \mathcal{R}$ ***konsistent*** im Limes genau dann wenn:

1. M berechnet für jedes Anfangsstück von f eine Hypothese,
   d.h. $\forall n \in \mathbb{N} : M(f[n]) \downarrow$,
2. M konvergiert gegen eine korrekte Hypothese,
   d.h. $\exists h \in \mathbb{N} : \varphi_h = f$ und $\exists m \in \mathbb{N}\ \forall n \geq m : M(f[n]) = h$
3. *Jede Hypothese von M ist konsistent*,
   d.h. $\forall n \in \mathbb{N}\ \forall x \leq n : \varphi_{M(f[n])}(x) = f(x)$.

Erkennungstyp: ***CONS***

Totale Hypothesen

Definition 4.6:
Ein Lernverfahren $M \in \mathcal{P}$ lernt eine Funktion $f \in \mathcal{R}$ mit ***totalen Hypothesen*** im Limes genau dann wenn:

1. M berechnet für jedes Anfangsstück von f eine Hypothese,
   d.h. $\forall n \in \mathbb{N} : M(f[n]) \downarrow$,
2. M konvergiert gegen eine korrekte Hypothese,
   d.h. $\exists h \in \mathbb{N} : \varphi_h = f$ und $\exists m \in \mathbb{N}\ \forall n \geq m : M(f[n]) = h$,
3. *jede Hypothese von M ist total*,
   d.h. $\forall n \in \mathbb{N} : \varphi_{M(f[n])} \in \mathcal{R}$.

Erkennungstyp: ***TOTAL***

Erste Einsichten

Theorem 4.1:
$FIN \subseteq TOTAL \subseteq CONS \subseteq LIM$

Beweis.

$\longrightarrow$ Übungsaufgabe

Beispielklassen

Beispiel 1:

Functions of finite support:
$U_{FFS} = \{f \in \mathcal{R} \mid f(x) > 0 \text{ nur für endlich viele } x \in \mathbb{N}\}$

Menge aller primitiv-rekursiven Funktionen
Selbstbeschreibende Funktionen:
$U_S = \{f \in \mathcal{R} \mid \varphi_{f(0)} = f\}$
$U_{S1} = \{f \in \mathcal{R} \mid \varphi_{f(0)} = f, f(x) > 0 \text{ für alle } \in \mathbb{N}\}$

Selbstbeschreibung $\longrightarrow$ ***Quines***

Beispiel für Java:

```
import java.text.*;class a{public static void main(String x[]){char b[]={34};
char c[]={123};String s[]=new String[3];s[0]="import java.text.*;class a{2}
public static void main(String x[]){2}char b[]={2}34};char c[]={2}123};
String s[]=new String[3];s[0]={1}{0}{1};s[1]=new String(b);s[2]=new String(c);
System.out.println(MessageFormat.format(s[0],s));}}";s[1]=new String(b);s[2]=
new String(c);System.out.println(MessageFormat.format(s[0],s));}}
```

Lernen durch totale Verfahren

- in Bedingung 1 Termination nur für Anfangsstücke der Zielfunktion gefordert
  - Macht das einen Unterschied?
- Können nun noch fordern, daß IIM total ist

Definition 4.7:
Sei *ET* ein Erkennungstyp. $R - ET$ bezeichnet die Menge aller lernbaren Funktionenklassen, für die es eine *totale* IIM gibt.
Erkennungstypen: ***R-LIM***, ***R-FIN***, ***R-CONS***, ***...***

Lernen durch totale Verfahren

Theorem 4.2:
R-LIM $=$ *LIM*.
R-TOTAL $=$ *TOTAL*.
R-FIN $=$ *FIN*.

Beweisidee.

Gegeben: IIM $M \in \mathcal{P}$

$M'(f[n])$:
Suche das größte $k \leq n$, so daß $M(f[k])$ innerhalb von n Schritten eine Hypothese berechnet.
Falls solch ein k gefunden, gib $M(f[k])$ aus, ansonsten eine Defaulthypothese.

Analyse:

- $M' \in \mathcal{R}$
- Wenn M eine Funktion im Limes lernt, dann auch M'

Beweis für *TOTAL* und *FIN* analog. *qed*

Konsistentes Lernen durch totale Verfahren

Theorem 4.3:
R-CONS $\subset$ *CONS*

ohne Beweis

Konsistentes Lernen vs. Lernen im Limes

Theorem 4.4:
CONS $\subset$ *LIM*

Wir zeigen: $U_{S1} \cup U_{FFS} \in$ *LIM* $\setminus$ *CONS*

Wir definieren ein Aufzählung ψ rekursiver Funktionen wie folgt:

$$\psi_i(0) = i$$

$$\psi_i(x+1) = \begin{cases} 1 : M(\psi_i[x]) \neq M(\psi_i[x] \circ 1) \\ 2 : \text{sonst} \end{cases}$$

Analyse:

1. ψ_i ist berechenbar
   - trivial
2. U_{FFS} ist anfangsstückvollständig
   - deshalb: $M(\tau)$ ist definiert für jedes beliebige Anfangsstück τ
   - insbesondere ist $M(\tau)$ konsistent für jedes τ
3. ψ_i ist total
4. entweder arbeitet M auf dem Graph von ψ_i inkonsistent oder M konvergiert nicht

Konsistentes Lernen vs. Lernen im Limes

Wir haben nur ein Problem: M muß ja ψ_i mglw. gar nicht lernen, d.h. gilt $\psi_i \in U_{S1} \cup U_{FFS}$?

Theorem 4.5:
Fixpunktsatz
Für jede Gödelnumerierung φ gilt:
Für jede allgemein-rekursive Funktion $f \in \mathcal{R}$ existiert ein $n \in \mathbb{N}$ (der Fixpunkt) so daß $\varphi_n = \varphi_{f(n)}$ gilt.

Wir bringen nun alles zusammen

- Bedingung 3 der Gödelnumerierung:
  $\forall \psi \in \mathcal{P}^2 \; \exists c \in \mathcal{R} \; \forall x \in \mathbb{N} : \psi_i(x) = \varphi_{c(i)}(x)$.
- $c \in \mathcal{R} \rightarrow$ es gibt einen Fixpunkt i^*, d.h. $\varphi_{i^*} = \varphi_{c(i^*)}$
- $\psi_{i^*} = \varphi_{c(i^*)} = \varphi_{i^*}$
- nach Definition gilt $\varphi_{i^*}(0) = i^*$ und $\varphi_{i^*}(x) > 0$ für alle $x \in \mathbb{N}$
- daraus folgt: $\varphi_{i^*} \in U_{S1} \cup U_{FFS}$

Aber: M kann φ_{i^*} nicht konsistent lernen!

Konsistentes Lernen vs. Lernen im Limes

$U_{S1} \cup U_{FFS} \in LIM \rightarrow$ Übungsaufgabe

qed

Total-Konsistentes Lernen

Es gibt einen Unterschied im konsistenten Lernen für totale und partielle IIMs.

Aber: Auch *R-CONS*-Maschinen müssen jedoch nicht überall konsistent sein, sondern nur auf Anfangsstücken zu lernender Funktionen.

Definition 4.8:
Ein Lernverfahren M heißt ***total-konsistent*** gdw. $M \in \mathcal{R}$ und $\forall f \in \mathcal{R}\ \forall n \in \mathbb{N}$: $\varphi_{M(f[n])}[n] = f[n]$.
T-CONS$= \{U \subseteq \mathcal{R} \mid \exists M \in \mathcal{R} : M$ ist total-konsistent und $U \subseteq$ *LIM*$(M)\}$.

Total-Konsistentes Lernen

Der vorige Beweis ergibt auch direkt die Einsicht, daß $U_S \notin$ *T-CONS* gilt.

Theorem 4.6:
T-CONS $\subset$ *R-CONS*.
T-CONS$\#$*FIN*.

Ohne Beweis:

Theorem 4.7:
T-CONS$\#$*TOTAL*.
R-CONS$\#$*TOTAL*.

Grenzen des Lernens im Limes

Theorem 4.8:
LIM $\subset \wp(\mathcal{R})$

$U_S \cup U_{FFS} \notin$ *LIM*

Beweis wie eben, nur etwas komplizierter...

Wir definieren ein Aufzählung ψ rekursiver Funktionen wie folgt:

Schritt 0: $\psi_i(0) = i$
Schritt $n > 0$: Sei ψ_i bis zu Stelle x definiert.
Suche das kleinste $j \in \mathbb{N}$ so daß

(a) $M(\psi_i[x]) \neq M(\psi_i[x] \circ 1 \circ 0^j)$ oder

(b) $M(\psi_i[x]) \neq M(\psi_i[x] \circ 2 \circ 0^j)$

gilt.
Im Fall (a) setze ψ_i fort mit $1 \circ 0^j$, im Fall (b) mit $2 \circ 0^j$.

Was ist eigentlich mit Identification by Enumeration?

Definition 4.9:
Eine Funktionenklasse $U \subseteq \mathcal{R}$ heißt ***effektiv aufzählbar*** ($U \in$***NUM***) gdw. $\exists g \in \mathcal{R}$: $U \subseteq \{\varphi_{g(n)} \mid n \in \mathbb{N}\} \subseteq \mathcal{R}$.

Theorem 4.9:
NUM $\subseteq$ *TOTAL*.
NUM $\subseteq$ *T-CONS*.

Wegen U_S und U_{FFS} wissen wir nun:

Folgerung 4.10:
NUM $\subset$ *TOTAL*.
NUM $\subset$ *T-CONS*.
NUM$\#$*FIN*.

Exakte Aufzählbarkeit

Achtung: es gibt noch die exakte Variante von Aufzählbarkeit:

Definition 4.10:
Eine Funktionenklasse $U \subseteq \mathcal{R}$ heißt ***exakt effektiv aufzählbar*** ($U \in$***NUM!***) gdw. $\exists g \in \mathcal{R} : U = \{\varphi_{g(n)} \mid n \in \mathbb{N}\}$.

Theorem 4.11:
NUM! $\subset$ *NUM*.

Beweis.

Sei M irgendeine nicht-aufzählbare Menge von Zahlen. Betrachte die Teilmenge U aller konstanten Funktionen, die M entspricht.

Die Menge aller konstanten Funktionen ist klarerweise aufzählbar, also auch U. U ist aber nicht exakt aufzählbar...

qed

Was passiert bei Nicht-Standardreihenfolge?

Mit ET^{arb} bezeichnen wir die Erkennungstypen, bei denen die IIM die Zielfunktion auf *beliebigen* Informationsfolgen erkennen muß.

Theorem 4.12:
$LIM^{arb} = LIM$.
$TOTAL^{arb} = TOTAL$.
$FIN^{arb} = FIN$.

Beweis.

$\longrightarrow$ Übungsaufgabe

Theorem 4.13:
$CONS^{arb} \subset CONS$.
$R\text{-}CONS^{arb} \subset R\text{-}CONS$.
$T\text{-}CONS^{arb} \subset T\text{-}CONS$.

Literatur:

[4] K.P. Jantke & H.-R. Beick: Combining Postulates of Naturalness in Inductive Inference. *Elektronische Informationsverarbeitung und Kybernetik* 17, pp. 465–484, 1981.

[5] R. Wiehagen & Th. Zeugmann: Learning and Consistency. *In: Jantke & Lange (eds.) Algorithmic Learning for Knowledge-Based Systems*, Lecture Notes in Artificial Intelligence 961, pp. 1–24, Springer-Verlag 1995.