General algorithms for learning languages

Learnable representations for languages

Alexander Clark

Department of Computer Science
Royal Holloway, University of London

August 2010
ESSLLI, 2010

Topics

- General algorithms
- Regular – Context-free – context-sensitive
- Transductions
- Discussion

Outline

Common properties

Several of the algorithms have common structure:

- We have a set of primitive elements that we use to define symbols
  - As we increase the set of primitives, the language increases.
- We have a set of features or experiments
  - As we increase the experiments, the language decreases
  - For sufficiently large, the language is a subset of the target.

Let's extract some general principles.

Primitive elements

We have a finite set of primitive elements Q

- $q \in Q$: a primitive element
- $\mathbf{D}(q)$: a set of strings defined by q
- $[[q]]$: a symbol in the grammar.

Example: DFAs

- q is a string
- $\mathbf{D}(q) = \{w | qw \in L\}$
- $[[q]]$ a state in the DFA

Sets of strings

Given a finite set Q, we have a collection of sets of strings
$\{\mathbf{D}(q) | q \in Q\}$
We can define various deductive rules.

Basic example

Suppose $\mathbf{D}(p) \subseteq \mathbf{D}(q)$.
Then if we know that $u \in \mathbf{D}(p)$, we can deduce that $u \in \mathbf{D}(q)$.

Not so basic example

Binary (B) rule

Suppose $\mathbf{D}(q)\mathbf{D}(r) \subseteq \mathbf{D}(p)$.
Then if a string $u \in \mathbf{D}(q)$ and $v \in \mathbf{D}(r)$, we can deduce that $uv \in \mathbf{D}(p)$

Not so basic example

Binary (B) rule

Suppose $\mathbf{D}(q)\mathbf{D}(r) \subseteq \mathbf{D}(p)$.
Then if a string $u \in \mathbf{D}(q)$ and $v \in \mathbf{D}(r)$, we can deduce that $uv \in \mathbf{D}(p)$

CFG

Suppose $N \rightarrow PQ$
Then if $P \overset{*}{\Rightarrow} u$ and $Q \overset{*}{\Rightarrow} v$
$N \overset{*}{\Rightarrow} uv$

Crucial point: local validity of rules

Lexical rules

The inferences must start somewhere:

Lexical (L) rules

We can infer that $w \in \mathbf{D}(q)$
Valid if $w \in \mathbf{D}(q)$
$[[q]] \rightarrow w$

Special cases:

L0 $[[q]] \rightarrow \lambda$

L1 $[[q]] \rightarrow a,\ a \in \Sigma$

Defining a language

Initial (I) rules

Suppose $\mathbf{D}(q) \subseteq L$
Then if $w \in \mathbf{D}(q)$, we can deduce that $w \in L$.
$S \rightarrow [[q]]$

Inference scheme

Suppose we have some finite set Q; a representation function $\mathbf{D}(q)$, and a languge L.
We can consider all possible *valid* inference rules of the type:

- L0,L1
- B,S
- I

This will give us a finite set of rules, which we call G.
We say that $[[q]] \stackrel{*}{\Rightarrow}_G w$ if there is a proof, using these rules that $w \in \mathbf{D}(q)$.
Define $L(G) = \{w | S \stackrel{*}{\Rightarrow}_G w\}$

Pause for discussion

- $L(G)$ is a context-free language
- G is basically just a CFG.

Pause for discussion

- $L(G)$ is a context-free language
- G is basically just a CFG.
- If the rules are all valid, then $L(G) \subseteq L$.

Pause for discussion

- $L(G)$ is a context-free language
- G is basically just a CFG.
- If the rules are all valid, then $L(G) \subseteq L$.
- We have only used an arbitrary collection of rules: we can add any more rules that we want.

Pause for discussion

- $L(G)$ is a context-free language
- G is basically just a CFG.
- If the rules are all valid, then $L(G) \subseteq L$.
- We have only used an arbitrary collection of rules: we can add any more rules that we want.

Conjunctive (C) rules

Suppose $\mathbf{D}(q) \cap \mathbf{D}(r) \subseteq \mathbf{D}(p)$.
Then if a string $u \in \mathbf{D}(q)$ and u is also in $\mathbf{D}(r)$ we can deduce that $u \in \mathbf{D}(p)$.
$[[p]] \rightarrow [[q]] \wedge [[r]]$.

Monotonicity

As we increase Q:

- $\{w | \mathbf{D}(q) \stackrel{*}{\Rightarrow} w\}$ will increase
- $L(G)$ will increase

If Q is such that $L(G) = L$, then any superset of Q is good.

Lattice

Natural structure

$\{\mathbf{D}(q) | q \in Q\}$ is a collection of sets. So it is naturally to consider a lattice.

If it is a partition, then the lattice is trivial.

CFG lattice

- V is the set of non-terminals in a CNF CFG
- $V(w) = \{N \in V | N \stackrel{*}{\Rightarrow} w\}$.
- Consider $\{V(w) | w \in \Sigma^*\}$

Three types

Valid rules

Rules are valid if the inference is in fact correct for all strings.
How can we figure out which rules are valid and which are not?

a priori Rules that are valid by definition

certain Rules that are definitely valid given some finite amount of information

defeasible Rules that seem valid, but might turn out to be invalid given further information.

Maintain a set X of experiments that we use to test defeasible rules.

Congruential approach

Definition

$\Sigma \cup \{\lambda\} \subseteq Q \subset \Sigma^*$

$\mathbf{D}(q) = \{u | C_L(u) = C_L(q)\}$

- $\mathbf{D}(u)\mathbf{D}(v) \subseteq \mathbf{D}(uv)$ is *a priori*
- $u \in \mathbf{D}(u)$ is *a priori*
- $\mathbf{D}(u) \subseteq L$ is *certain* if we know that $u \in L$.
- tricky: $\mathbf{D}(u) \subseteq \mathbf{D}(v)$

Congruential approach

Definition

$\Sigma \cup \{\lambda\} \subseteq Q \subset \Sigma^*$

$\mathbf{D}(q) = \{u | C_L(u) = C_L(q)\}$

- $\mathbf{D}(u)\mathbf{D}(v) \subseteq \mathbf{D}(uv)$ is *a priori*
- $u \in \mathbf{D}(u)$ is *a priori*
- $\mathbf{D}(u) \subseteq L$ is *certain* if we know that $u \in L$.
- tricky: $\mathbf{D}(u) \subseteq \mathbf{D}(v)$
- If $\mathbf{D}(u) \subseteq \mathbf{D}(v)$ then $\mathbf{D}(u) = \mathbf{D}(v)$

Positive data only

Substitutable

E-rules are *certain*

If we see lur, lvr then $\mathbf{D}(u) = \mathbf{D}(v)$

Testing congruence

Set of experiments

X is a set of contexts
Test $C_L(u) \cap X = C_L(v) \cap X$

Key points:

- As X increases the test is more accurate
- For sufficiently large X all invalid defeasible E rules will be removed.
  Pick a context in symmetric difference of $C_L(u)$ and $C_L(v)$.

Primal versus dual

	Primal	Dual
Q	u	(l, r)
$\mathbf{D}(q)$	$\{v \mid v \equiv_L u\}$	$\{w \mid lwr \in L\}$
X	(l, r)	u
L	a priori	certain
B	a priori	defeasible
E	defeasible	–
I	certain	a priori

Overall algorithm

Algorithm maintains two sets:

- Primitive elements: Q
- Set of experiments or tests X

Make

Construct grammar from Q,X and information about L

IncreaseQ

Increase Q because the hypothesis undergenerates.

IncreaseX

IncreaseX to remove invalid rules because the hypothesis overgenerates.

Algorithm

$E \leftarrow \emptyset$;
$Q \leftarrow$ **InitQ** ;
$X \leftarrow$ **InitX** ;
$G \leftarrow$ **Make**(Q, X, O, E) ;
while w_i *is a positive example* **do**
    $E \leftarrow E \cup \{w_i\}$;
    for $w \in E$ **do**
        if *not* $S \stackrel{*}{\Rightarrow}_G w$ **then**
            $Q \leftarrow Q \cup$ **IncreaseQ**(E) ;
    $X \leftarrow X \cup$ **IncreaseX**(w_i) ;
    $G =$ **Make**(Q, X, O, E) ;

Outline

Multiple context free grammars

- CFGs have non-terminals that derive strings: $N \stackrel{*}{\Rightarrow} u$

Multiple context free grammars

- CFGs have non-terminals that derive strings: $N \stackrel{*}{\Rightarrow} u$
- MCFGs have non-terminals that derive tuples of strings: $N \stackrel{*}{\Rightarrow} (u, v)$
  Each non-terminal has a dimension $dim(A)$

MCFG derivation

$A \rightarrow f(B, C)$ where $f(\langle x_1, x_2, x_3 \rangle, \langle y \rangle) = \langle x_1 ayx_2, x_3 \rangle$
$dim(A) = 2, dim(B) = 3, dim(C) = 1$

Limited rules

Not all rules are allowed for combining

- Linear regular non-permuting

Limited rules

Not all rules are allowed for combining

- Linear regular non-permuting
  - Each variable used only once
  - The variables in each tuple must occur in that order

M-contexts

String and context

$w \in \Sigma^*$

$(l, r) \in \Sigma^*$

Context

A hole – $\square$

$l\square r \odot u = lur$

m-context

A string in $(\Sigma \cup \{\square\})^*$ with m holes

$u_0\square u_1 \ldots \square u_m \odot (v_1, \ldots, v_m)$

$u_0 v_1 u_1 \ldots v_m u_m$

Distribution

Suppose $\mathbf{w}$ is an m-tuple: $\langle w_1, \ldots, w_m \rangle$
$L/\mathbf{w} = \{\mathbf{v} | \mathbf{v} \odot \mathbf{w} \in L\}$

Congruence

Define $\mathbf{u} \equiv_L \mathbf{v}$ iff
$L/\mathbf{u} = L/\mathbf{v}$

Example

$L = \{cwcw | w \in \{a, b\}^*\}$
$\langle c, c \rangle \equiv_L \langle ca, ca \rangle \equiv_L \langle cb, cb \rangle$

Result

Yoshinaka and Clark (2010)

Congruential MCFGs

A congruential MCFG is one where all tuples generated by a non-terminal are congruent.

Result

Congruential MCFGs are polynomially learnable from MQs and EQs.

Outline

Transductions

Two alphabets

Σ, and Δ

$T \subseteq \Sigma^* \times \Delta^*$

- Machine translation
- When $\Sigma = \Delta$, inflectional morphology.

Functions

Suppose for each u in $Dom(T)$, there is only one v, such that $(u, v) \in T$.

- If we see $(u, v) \in T$, then we know that $(u, v') \notin T$.
- This means we can learn from positive data alone.

OSTIA algorithm

Classic inference algorithm.
All subsequential transducers

New algorithm

Primitive elements

$(u, v) \in \Sigma^* \times \Delta^*$

$\mathbf{D}(u, v) = \{(x, y) | (ux, vy) \in T\}$

New algorithm

Primitive elements

$(u, v) \in \Sigma^* \times \Delta^*$

$\mathbf{D}(u, v) = \{(x, y) | (ux, vy) \in T\}$

Rule types

I	$S \rightarrow [[(\lambda, \lambda)]]$ *a priori*
R	$[[(u, v)]] \rightarrow (u', v')[[(uu', vv')]]$ *a priori*
L0	$[[(u, v)]] \rightarrow (\lambda, \lambda)$ iff $(u, v) \in T$ *certain*
E	$[[(u, v)]] \rightarrow [[(u', v')]]$ defeasible

States

Given a pair (u, v) we define one state for every possible prefix.

Example $(aab, cdcc)$

(λ, λ)	(λ, c)	(λ, cd)	(λ, cdc)	$(\lambda, cdcc)$
(a, λ)	(a, c)	(a, cd)	(a, cdc)	$(a, cdcc)$
(aa, λ)	(aa, c)	(aa, cd)	(aa, cdc)	$(aa, cdcc)$
(aab, λ)	(aab, c)	(aab, cd)	(aab, cdc)	$(aab, cdcc)$

A priori transitions

Given states (u, v) and (ux, vy), add transition labelled $\rightarrow^{x,y}$.

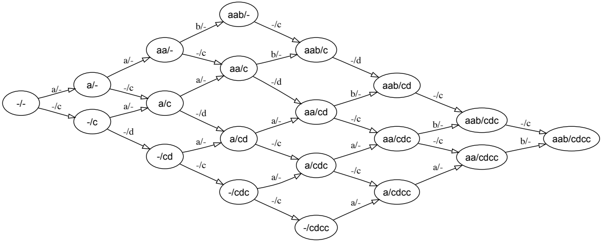
Example: start state (aa, cd)

- $(aa, cd) \rightarrow^{(\lambda,c)} (aa, cdc)$
- $(aa, cd) \rightarrow^{(\lambda,cc)} (aa, cdcc)$
- $(aa, cd) \rightarrow^{(b,\lambda)} (aab, cd)$
- $(aa, cd) \rightarrow^{(b,c)} (aab, cdc)$
- $(aa, cd) \rightarrow^{(b,cc)} (aab, cdcc)$

Prefix tree transducer

$(aab, cdcc)$

Only some of the transitions.



Terminal transitions

Identifying the final states

If $(u, v) \in T$, then we add a rule $(u, v) \rightarrow (\lambda, \lambda)$.
This is a certain rule: if we observe (u, v) then we know for sure that this rule is valid.

Defeasible rules

Equality rules if $\mathbf{D}(u, v) = \mathbf{D}(u', v')$.

Incorrect

If there is a pair of elements in the data of the form $(ux, vy), (u'x, w')$ such that $w' \neq v'y$, then we know that the rule is incorrect.

- If the rule were correct then we should see $(u'x, v'y)$.
- If we see a different output for the input $u'x$ then we know that $\mathbf{D}(u, v) \neq \mathbf{D}(u', v')$.

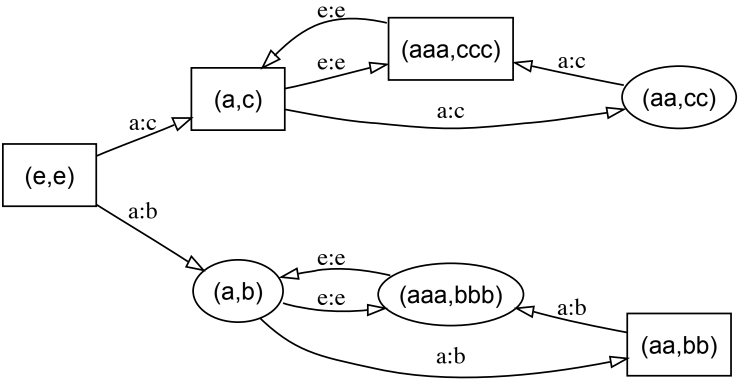
Example

Target

$$T = \{(a^{2n}, b^{2n}) | n \geq 0\} \cup \{(a^{2n+1}, c^{2n+1}) | n \geq 0\}$$

- Classic example of non-deterministic transducers
- Learnable class includes subsequential transducers, but maybe not all FSTs.

Example



Outline

Overview

Two problems

- Information theoretic problems
- Computational complexity

Progress

We now have a partial understanding of a class of algorithms for grammatical inference.

- Computationally efficient – solves the second problem
- Probabilistic data solves the first problem

Relevance to Language acquisition

Question

We have developed a **theoretical** approach to grammatical inference. What relevance does this have to the **empirical** question of language acquisition and linguistics in general?

Answer 1

Answer

None at all, because you are relying on MQs, which are not available.

“This is not a learnability result just a complexity result”

Answer 2

Answer

None at all, because you are just learning the sets of strings; what the child knows is a mapping between syntax and semantics.

- We can only learn the syntax/semantics interface from a richer set of data
- If we have more data, then that is a slightly different problem.

Answer 3

Wrong representation

We know that the right representation involves transformation (or Merge, or unification of feature structures . . .), and your algorithm doesn't output this.

Wrong trees

Your algorithm needs to output the right constituent structure trees; then we can evaluate against those that linguists have determined to be correct.

Conclusion

Jackendoff (2008)

1. Descriptive constraint: the class of languages must be sufficiently rich to represent natural languages
2. Learnability constraint: there must be a way for the child to learn these representations from the data available
3. Evolutionary constraint: it must not posit a rich, evolutionarily implausible language faculty

- We have a family of models that potentially satisfy all three criteria