Language Acquisition: Parameter Setting

Matilde Marcolli
CS101: Mathematical and Computational Linguistics

Winter 2015

Example: a 3-parameter system of grammars

- E. Gibson, K. Wexler, *Triggers*, Linguistic Inquiry, 25 (1994) 407–454

X-bar production rule: two word-order parameters

- a parameterized Phrase Structure Grammar with production rules

$$XP \to \mathrm{Spec}X' (\Pi_1 = 0) \text{ or } X'\mathrm{Spec} (\Pi_1 = 1)$$

$$X' \to \mathrm{Comp}X' (\Pi_2 = 0) \text{ or } X'\mathrm{Comp} (\Pi_2 = 1)$$

$$X' \to X$$

- XP phrase of lexical type X (N noun, V verb, A adjective,...)
- Spec = specifier (e.g. “the old” in “the old book”)
- Comp = complement

• Spec and Comp are constituents that can further be broken down into structure comprising other Spec and Comp elements...

• so also have productions

$$\text{Spec} \to XP, \quad \text{Comp} \to XP$$

• Spec and Comp positions in a phrase may be blank: productions

$$\text{Spec} \to \emptyset, \quad \text{Comp} \to \emptyset$$

• Note that production rules are parameterized

• Spec-first languages $\Pi_1 = 0$; Spec-final languages $\Pi_1 = 1$

• similarly Comp-first and Comp-final languages, $\Pi_2 = 0$, $\Pi_2 = 1$

• Example: English is Spec-first Comp-final;
Bengali is Spec-first Comp-first

A Transformational Parameter

- parameters Π_1 and Π_2 above are generative (word order)
- the *V2-parameter* governs movement of words in a sentence
- Example: German sentences

- *Karl kauft das Buch*
- *Ich weiß, dass Karl das Buch kauft*

- the first sentence looks Comp-final, the second looks Comp-first
- *deep structure* (generated by grammar production rules) is Comp-first; but an additional parameter $\Pi_3 = 1$ (the $V2$-parameter) is set so that in *surface structure* (obtained by transformational rules) finite verbs must move to second position in declarative clauses
- special case of the Move-α transformations of Transformational Grammars

3-parameter model

• restrict to these three parameters Π_1, Π_2, Π_3

• space of 8 possible grammars

• alphabet $\mathfrak{A}$ just given by the syntactic categories (parts of speech): V,N,A,...

Language Learning in the Principles and Parameters setting

• language acquisition = correctly identifying the parameters of the target grammar

Gibson and Wexler's Triggering Learning Algorithm

• sequence of (positive) examples of sentences $s_1, s_2, \dots, s_n, \dots$

• after each new example received, learner either stays on same state or moves to new one (by affecting some parameter change)

• successful learning: identified target language and after some example s_N no longer move from a certain state

• two constraints:

1. only one parameter change at each step
2. if s_n not recognized by present state, effect parameter change only if this makes s_n recognizable

Steps of TLA algorithm:

1. *Initialization*: start at a random point in the space of parameters and a grammar with those values of parameters
2. *Input*: receive positive example sentence s drawn with a uniform distribution
3. *Error detection*: if current grammar generates s go to previous step and receive new input; if grammar does not parse go to next step
4. *Single-step hill climbing*: select a single parameter uniformly randomly, check if flipping parameter makes s compatible; if yes flip, if no get new input

Learnability

• still learnability problem occurs: Gibson and Wexler showed the 8-parameter space of previous example is unlearnable with TLA

• source of the problem: local maxima (false solutions) that process cannot escape

• ... but *conjectured*: learnability holds if there are triggers for each pair of hypothesis and target in the parameterized space of grammars

• trigger: a sentence *s* in target language that cannot be parsed with hypothesis grammar and that give (indirect) information about the target parameter structure

• ... but stochastic model shows still insufficient: even if such path from hypothesis to target always exists, learner may with high probability take a wrong path that leads to a (wrong) other local maximum

Parameter Space Learning as a Markov Chain

- N parameter: space $\mathcal{H}$ of grammars with 2^N points
- each boolean vector of length N: a hypothesis state
- space endowed with Hamming distance (distance = number of parameters that differ)
- possible transitions between states can only change one parameter
- weights p_{ij} on transition from state i to state j: probability of transition
- probabilities p_{ij} are determined by a probability distribution $\mathbb{P}$ on the language $\mathcal{L}$ of the target grammar
- target state has an oriented loop to itself and no other outgoing edges (absorbing state)

Markov Chain and Learnability

- $\mathcal{A} : \mathcal{D} \to \mathcal{H}$ (memoryless) learning algorithm
- $\mathcal{G}^{(t)} \in \mathcal{H}$ target grammar
- $\mathbb{P}$ probability on $\mathcal{D}$ (from probability on $\mathcal{L}_{\mathcal{G}^{(t)}}$: positive examples)
- closed set C of states: subset of states with no outgoing arc directed at other states (outside C)
- learnability: $\mathcal{A}$ identifies $\mathcal{G}^{(t)}$ in the limit with probability 1
- Fact: $\mathcal{G}^{(t)}$ learnable through $\mathcal{A}$ algorithm and probability $\mathbb{P}$ iff in associated Markov Chain every closed set C contains $\mathcal{G}^{(t)}$

Construction of the Markov Chain

- one state of Markov chain for each parameter vector (2^N nodes)
- when receiving input s (with probability $\mathbb{P}(s)$) state $\mathcal{L}_s$
- arrow from state $\mathcal{L}_s$ to set $\mathcal{L}_{s'}$ iff both
  1. next sentence s' is not parsed by $\mathcal{L}_s$ but is parsed by $\mathcal{L}_{s'}$
  2. $\mathcal{L}_s$ and $\mathcal{L}_{s'}$ are a single parameter-flip from each other

- first property occurs with probability (sentences both in $\mathcal{L}^{(t)}$ and $\mathcal{L}_{s'}$ but not in $\mathcal{L}_s$)

$$\sum_{x \in (\mathcal{L}_{s'} \smallsetminus \mathcal{L}_s) \cap \mathcal{L}^{(t)}} \mathbb{P}(x)$$

- second property with probability $1/N$ (parameter to flip chosen uniformly at random)

Probabilities

- $\mathbb{P}(s \to s') = \sum_{x \in (\mathcal{L}_{s'} \setminus \mathcal{L}_s) \cap \mathcal{L}^{(t)}} \mathbb{P}(x)$

$$\mathbb{P}(s \to s) = 1 - \sum_{s' \neq s} \mathbb{P}(s \to s') = 1 - \sum_{\substack{s' \neq s \\ x \in (\mathcal{L}_{s'} \setminus \mathcal{L}_s) \cap \mathcal{L}^{(t)}}} \mathbb{P}(x)$$

Construction procedure summary:

1. assign $\mathbb{P}$ on $\mathcal{L}^{(t)}$
2. assign a state to each language $\mathcal{L}$ with 2^N states
3. compute Hamming distances
4. if $d_H(\mathcal{L}_s, \mathcal{L}_{s'}) > 1$ set $\mathbb{P}(s \to s') = 0$
5. normalize by target language: $\mathcal{L}' = \mathcal{L} \cap \mathcal{L}^{(t)}$
6. if Hamming distance 1: take $\mathbb{P}(s \to s') = N^{-1}\mathbb{P}(\mathcal{L}'_{s'} \setminus \mathcal{L}')$
7. take $\mathbb{P}(s \to s) = 1 - \sum_{s' \neq s} \mathbb{P}(s \to s')$

States in Markov Chains

- equivalent states in a MC: s is reachable from s' (following an oriented path) and vice versa
- recurrent state in a MC: chain returns to s in a finite number of steps with probability 1
- transient state in a MC: not recurrent
- $\mathbb{P}_{ss'}(n)$ = probability of going from state s to state s' in n steps
- state s' transient $\Rightarrow \lim_{n\to\infty} \mathbb{P}_{ss'}(n) = 0$ for all s
- canonical decomposition of a Markov Chain

$$T \cup C_1 \cup \cdots \cup C_m$$

disjoint union of T = set of transient states, C_j = closed sets of equivalence classes of recurrent states

Why learnability result works?
(learnability iff all closed sets in Markov Chain contain target)

• if some closed set C does not contain target: if learner starts inside C will never reach target (unlearnable)

• suppose all closed set contain target: show using MC decomposition that all non-target states must be transient

• then $\lim_{n\to\infty} \mathbb{P}_{ss'}(n) = 0$ for s' transient shows with probability 1 must converge in the limit to target

• transience of non-target states: know target absorbing, so no other state can be in same equivalence relation (cannot reach any other state); target is recurrent (one arrow going back to itself in one step); target state is a closed class C_i in MC decomposition, but has to be in all closed sets so in all C_i's: only one C, rest is T