Automatically Exploring Hypothesis Spaces

October 9, 2018

1 Hypothesis generation

A goal of the grammar learner, whether it is a child, linguist, or computer program, is to infer the grammar that gave rise to a collection of utterances. In reference to the previous section (Sec. **??**), we can think of this as a kind of *exploratory data analysis.* One way of formalizing this inference problem is to (1) place a prior distribution over grammars, for example, a distribution that puts more probability mass on shorter or simpler grammars; (2) equip each grammar with a *likelihood model* that specifies exactly how likely a grammar is to produce a given utterance; and then (3) use Bayes's rule to work backwards from the utterances to the grammar that was likely to produce them:

$$P\left(\text{grammar}|\text{utterances}\right) \propto \underbrace{P\left(\text{utterances}|\text{grammar}\right)}_{\text{Likelihood}} \times \underbrace{P\left(\text{grammar}\right)}_{\text{Prior}} \tag{1}$$

We can think of the above equation as a recipe for scoring how well a grammar explains a collection of utterances. Once we have a prior and a likelihood, we know how to evaluate how well a grammar explains the data, modulo the assumptions implicit in the prior and likelihood. Our goal will be both to automatically recover the single best grammar maximizing 1, as well as systematically exploring and generating the space of possible grammars (i.e. hypotheses).

Here we consider an AGL learner tasked with learning a grammar generating different word forms, for example those in marcus1999rule,gerken2010infants. We model these grammars as context-sensitive rewrite rules chomsky1968sound, commonly used in generative phonology, but at a high level this framing can apply to learning other aspects of grammar like semantics piantadosi2011learning, syntax perfors2011learnability, and morphology tim. Each rewrite rule is a function that both inputs and outputs a sequence of phonemes. These rewrites are written as "input → output / left_right" which means that "input" gets rewritten to "output" whenever "left" is to the left and "right" is to the right. Rules can refer to sets of phonemes by writing down vectors of phonological features; rules can also bind variables to phonemes or syllables.

Given a collection of rewrite rules that define possible sequences of phonemes, we can define *prior* probabilities over grammars, and the *likelihood* of the data given a grammar. A simple and intuitive prior over grammars is one which penalizes longer grammars and favors parsimonious grammars, for example defining

$$P\left(\text{grammar}\right) \propto \exp\left(-\#\text{ symbols in grammar}\right)$$

An example of a likelihood model, and the one we use here, is just to count the number of extra bits or symbols needed to encode an utterance given the grammar[1].

In theory, this probabilistic framing has now bought us a procedure for automatically generating grammars that do a good job explaining the data. In practice, the space of all grammars is infinite and combinatorial, and so cashing out this framing requires a efficient procedure for searching the space of grammars and honing in on those that maximize equation 1. A subfield of computer science, called **program synthesis**, has developed efficient techniques for solving seemingly intractable combinatorial search problems like those that arise in grammar learning. We use a program synthesis tool called Sketch solar2008program to search for grammars maximizing equation 1, which was previously applied to language learning problems in [6]. For more detail on these program synthesis techniques we refer the reader to [5]. This cashing out of the framing makes our approach an instance of **Bayesian Program Learning**; see [7].

The fundamental hypothesis underlying AGL research is that artificial grammar learning must engage some cognitive resource shared with first language acquisition. To gain insights into first language acquisition, we instead study AGL in controlled experiments. In particular, we are interested in how the human mind deals with the trade-off between grammars that are *a priori* probable (the prior prefers small grammars; "parsimony"), and grammars that assign high likelihood to the data (and therefore closely "fit the data"). For example, a learner could infer a grammar that just memorized the data (perfect fit but poor parsimony) or it could infer a grammar that can generate every possible word (parsimonious but a poor fit). Effective grammar learners must navigate this trade-off.

To analyze this trade-off, we present the model with artificial data drawn from grammars used in a series of AGL experiments gerken2010inf
The grammars used (and the notation we use for them) are described in Table 1.

Rather than producing a single grammar – a single hypothesis on the grammar a child might acquire in such an experiment – we use the model to search a massive space of possible grammars and compute all those grammars that are optimal solutions to the trade-off given some weighting of parsimony and fit to data. That is, we search for the set of grammars that are not worse than another grammar along the two competing axes; this set is called the **pareto front** mattson2005pareto. Intuitively, grammars on the pareto front are ones which an ideal Bayesian learner prefers, *independent* of how the learner decides to relatively weight the prior and likelihood.

Figure 1 diagrams the Pareto fronts for two AGL experiments as the number of example words provided to the learner is varied. What these Pareto fronts show is (1) the set of grammars entertained by the learner, and (2) how the learner weighs these grammars against each other as measured by the prior (parsimony) and the likelihood (fit to the data).

[1]This is equivalent to a minimum description length (MDL) interpretation.

Grammar	Example input to model	Inferred grammar	Natural language analogues
ABA	wofewo lovilo fimufi	$\varnothing \rightarrow \sigma_i$ / #_$\sigma\sigma_i$	Reduplication (eg, Tagalog)
ABB	wofefe lovivi fimumu	$\varnothing \rightarrow \sigma_i$ / σ_i_#	Reduplication (eg, Tagalog)
ABx	wofeka lovika fimuka	stem+x	concatenative morphology
AAx	wowoka loloka fifika	stem+x $\varnothing \rightarrow \sigma_i$ / #_σ_i	reduplication concatenative morphology
AxA	wokawo lokalo fikafi	x+stem $\varnothing \rightarrow \sigma_i$ / #_$\sigma\sigma_i$	Infixing Reduplication
Pig Latin	pɪg→ɪgpe latɪn→atɪle æsk→æske	$\varnothing \rightarrow \mathrm{C}_i$ / #C_i[$]_0$_# $\varnothing \rightarrow$ e / _# C $\rightarrow \varnothing$ / #_	

Table 1: Using Bayesian program learning to model artificial grammar learning. Each model given 5 examples. Grammars ABA and ABB are from Marcus et al. 1999; grammars AAx and AXA from Gerken, 2006. Some special symbols used in the grammar rules are # to mean the start or end of the input; C/V to mean a consonant/vowel; σ to mean a syllable; and $\varnothing$ to mean the empty string. For example, the rule $\sigma \rightarrow \varnothing$ / #_C deletes a syllable at the start of a word if it is followed by a consonant. The rule [-son] $\rightarrow$ [+voice] rewrites segments that are -sonorant (e.g., k,d,v,ʃ,...) to their voiced counterparts (e.g., g,d,v,ʒ,...). The rule $\mathrm{C}_i \rightarrow \varnothing$ / _C_i deletes a consonant if it is followed by the same consonant, and $\varnothing \rightarrow \sigma_i$ / #_σ_i inserts a copy of a word initial syllable. The special subscript "0" means zero or more occurrences of the subscripted segment, so for example V $\rightarrow$ [+hi] / [+hi][$]_0$_ means that vowels become +hi whenever there exists a +hi phoneme anywhere to the left.

As an example of how the Pareto front offers a lens on the space of possible hypotheses, consider the upper left Pareto front in figure 1 (*aab, 1 example*). Here the learner has seen a single word, [vɛfefe]. Some grammars living on the Pareto frontier are:

- **A grammar that generates every possible word:** In the lower right-hand corner of Figure 1 is the grammar labeled "surface=underlying", which just says that every word (a "surface pronunciation") is represented ("underlyingly") literally how it is pronounced. The observed word [vɛfefe] is represented as /vɛfefe/, which has 6 symbols, giving a fit to the data of -6.
- **A grammar that duplicates syllables:** Highlighted in dark red in Figure 1 is a grammar with the single rule $\varnothing \rightarrow \sigma_i$ / _σ_i# which inserts a copy of the last syllable. This rule generates the word [vɛfefe] by starting with the stem (i.e., underlying form) /vɛfe/ (which has 4 symbols, fit to data of -4) and then applying the rule in the grammar, which copies the last syllable and makes [vɛfefe].
- **A grammar that duplicates syllables and appends morphemes:** Highlighted in pink in Figure 1 are grammars that duplicate a syllable but also append or prepend extra morphemes. These correspond to generalizations where the learner believes that different parts of [vɛfefe] correspond to prefixes or suffixes in the language. For example, the grammar in the upper left corner incorporates the suffix /fe/ as well as the prefix /vɛ/, and explains the observation using an underlying form that is completely empty (fit to data of 0) – this grammar has memorized the observed data, and so maximally compresses it, but at the cost of having many symbols inside of the grammar (22 symbols, vs 6 symbols in the grammar that just duplicates a syllable).

Grammar induction is an under constrained problem, and in general there are infinitely many grammars that could explain a collection of utterances. Despite this ambiguity, both children and linguists can make plausible inferences about which grammars best explain a data set. The computational tools shown here offer a generic way of generating and visualizing the range of alternative hypotheses, and explaining why a child, linguist, or computer program might prefer one over the other.

References

[1] N. Chomsky and M. Halle. *The sound pattern of English.* Studies in language. Harper & Row, 1968.

[2] Steven Thomas Piantadosi. *Learning and the language of thought.* PhD thesis, Massachusetts Institute of Technology, 2011.

[3] Amy Perfors, Joshua B Tenenbaum, and Terry Regier. The learnability of abstract syntactic principles. *Cognition*, 118(3):306–338, 2011.

[4] Timothy J. O'Donnell. *Productivity and Reuse in Language: A Theory of Linguistic Computation and Storage.* The MIT Press, 2015.

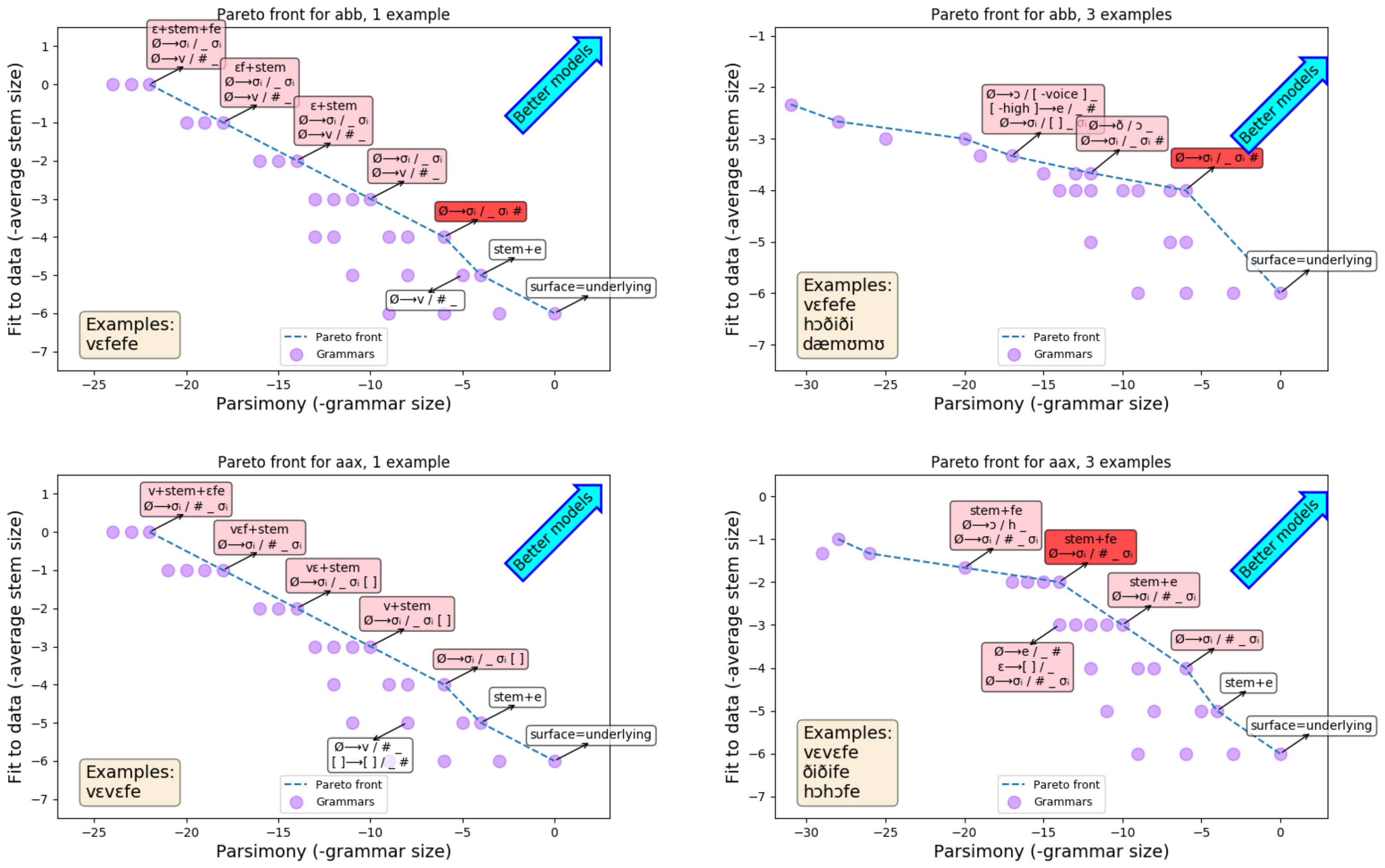

Figure 1: Pareto fronts for ABB (Marcus 1999; top two) & AAX (Gerken 2006; bottom two) learning problems for either one example word (left column) or three example words (right column). Rightward on x-axis corresponds to more parsimonious grammars and upward on y-axis corresponds to grammars that best fit the data, so the best grammars live in the upper right corners of the graphs. Red shade: ground truth grammar. Pink shade: shares structure with ground truth grammar. White shade: incorrect generalizations. As the number of examples increases, the Pareto fronts develop a sharp kink around the ground truth grammar, which indicates a stronger preference for the correct grammar. With one example the kinks can still exist but are less pronounced.

[5] Armando Solar Lezama. *Program Synthesis By Sketching.* PhD thesis, EECS Department, University of California, Berkeley, Dec 2008.

[6] Kevin Ellis, Armando Solar-Lezama, and Josh Tenenbaum. Unsupervised learning by program synthesis. In *NIPS*.

[7] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[8] LouAnn Gerken. Infants use rational decision criteria for choosing among models of their input. *Cognition*, 115(2):362–366, 2010.

[9] Gary F Marcus, Sugumaran Vijayan, S Bandi Rao, and Peter M Vishton. Rule learning by seven-month-old infants. *Science*, 283(5398):77–80, 1999.

[10] Michael C Frank and Joshua B Tenenbaum. Three ideal observer models for rule learning in simple languages. *Cognition*, 120(3):360–371, 2011.

[11] Christopher A Mattson and Achille Messac. Pareto frontier based concept selection under uncertainty, with visualization. *Optimization and Engineering*, 6(1):85–115, 2005.

[12] David Odden. *Introducing Phonology.* Cambridge University Press, 2005. Cambridge Books Online.