Learning syntactic parameters without triggers by assigning credit and blame

Prickett, Holladay, Hucklebridge, Nelson,
Bhatt, Jarosz, Johnson, Nazarov, & Pater

University of Massachusetts Amherst
University of Toronto

May 16, 2019

Introduction

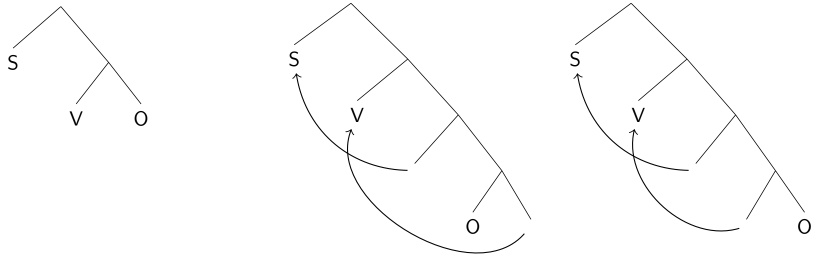
- Parametric approaches have been widely adopted in syntax since Chomsky (1981) as a solution to Plato's Problem
- Learning of realistically-sized syntactic parameter systems has received insufficient attention[1]
- We adapt two domain-general learning algorithms from phonological modeling and apply them to two simple syntactic parametric systems
- Parameter settings are learned without triggers (cf. Gibson & Wexler 1994, Fodor 1998), and with consistent time-to-convergence (cf. Yang 2002)

[1] Especially when compared with phonological modeling: see Dresher & Kaye (1990), Tesar & Smolensky (2000), Nazarov & Jarosz (2017, 2019)

Learning syntactic parameter settings: the challenge

A learner hears an SVO sentence
What should their target grammar be?

VO, no V2? OV and V2? VO and V2?



Hidden structure in parametric learning

- Movement can make the settings of headedness parameters ambiguous
- Conversely, headedness parameters can make the settings of movement parameters ambiguous
- How does the learner find the right settings of parameters?
- The learner must somehow induce the “hidden structure”[2]: the setting of parameters that correctly predicts the surface linear order for all data

[2] e.g. Tesar (1998), Jarosz (2006), Nazarov (2016)

Previous approaches utilize "triggers"

- Gibson & Wexler (1994):
  - Learners need data that unambiguously require parameters to be set a certain way
  - These data are referred to as *triggers*
- Fodor (1998):
  - Learners are on the lookout for particular subtrees, provided by UG, that are informative w.r.t. parameter setting
  - These subtrees are *triggers*
- In both approaches, parameters are set categorically once the relevant triggers are encountered
- A problem is that even in simple parameter systems, many languages have no unambiguous surface data (Gibson & Wexler 1994, Gould 2015)

Embracing ambiguity

- Ambiguous data are not meaningless
  - An SVO sentence is evidence for V2 if the learner believes their language is underlyingly OV
  - Existing beliefs, from a prior or from other data, can reduce ambiguity
- The extent to which an ambiguous datum is informative should depend on the strength of a learner's existing beliefs about individual parameter settings
- Crucially, maintaining gradient preferences over hypotheses allows the learner to utilize ambiguous data[3]
- Maintaining beliefs about individual parameter settings requires solving the Credit/Blame Problem

[3] Jarosz (2006), Jarosz (2015), Gould (2015)

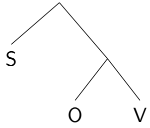
The Credit/Blame Problem

The Credit/Blame Problem is the challenge of determining which *individual* parameters are responsible for good/bad predictions

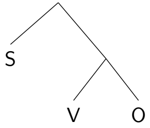
Consider two headedness parameters, *Comp* and *Spec*

These determine how objects are linearized w.r.t. verbs and how subjects are linearized w.r.t. {V, O}, respectively

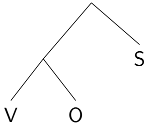
Current belief: SOV



Observed: SVO



Naïve update: VOS



A naïve learner, with no means for analysis, will penalize *all* current parameter settings

Current work

- We implement three models for parameter learning
- The first of these, the Naïve Parameter Learning (Yang, 2002), has already been applied to syntactic learning
- We also apply two other approaches – Expectation Driven Learning (Jarosz 2015) & a Maximum Entropy Classifier – to syntactic parameter learning in a novel way
- In all models, learners update gradient beliefs about their grammar
- Only in the latter two models do learners probabilistically *analyze* ambiguous forms – they directly address the Credit/Blame Problem

Learning task

The learning task is to find the correct settings for headedness and movement parameters given ordered pairs

The first element in each ordered pair represents constituency, and conforms to the following template

- { A { S { O V } } }

The second element is a linearized string

Example learning data:

Constituency	**String**
{ S { V } }	SV
{ A { S { O V } } }	AVSO
{ S { O V } }	SVO
...	...

Learning task: three parameter system

Our three parameter system is based on that of Gibson & Wexler (1994)

There are two headedness parameters:

- *Comp*: O is to the (left, right) of V
- *Spec*: S is to the (left, right) of {V, O}

There is one movement parameter:

- *V2*: V is linearly second in the output, and the highest (non-V) word is linearly first

Learning task: four parameter system

Given the importance of learning models scaling well, we designed a system with four parameters and embedded clauses

This parametric system better reflects typology

Spec is done away with, *Comp* is retained

V2 is split into three parameters

- *V.move*: V (is, is not) fronted in all clauses
- *V.move.matrix*: V (is, is not) fronted in matrix clauses
- *Topic*: some non-V word (is, is not) fronted to first position

V.move	**V.move.matrix**	**Topic**	**Languages**
on	*n/a*	on	Yiddish, Icelandic
on	*n/a*	off	Irish
off	on	on	German
...	...	...	...

Models

Models:

- Naïve Parameter Learning (NPL)
- Expectation Driven Learning (EDL)
- Maximum Entropy Model (MaxEnt)

NPL & EDL make use of a Stochastic Parameter Grammar

MaxEnt uses a Maximum Entropy Grammar fit with Stochastic Gradient Descent

Stochastic Parameter Grammars

Every parameter has a probability distribution over possible values

Generate by randomly sampling a value for all parameters

Example: *Comp* is 0.8 left and 0.2 right

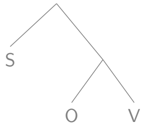
- Will generate OV 80% of the time, VO 20%

The values of parameters are 0 and 1, arbitrarily assigned

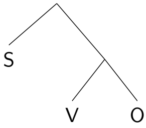
Naïve Parameter Learning (Yang 2002)

Observe data, sample from grammar to generate prediction

Observed: SOV
(Comp-L, Spec-L)



Sampled: SVO
(Comp-R, Spec-L)



Update with the Linear Reward-Penalty Scheme (Bush and Mosteller 1951):

$$p(\psi_i \mid G_{t+1}) = \lambda R(\psi_i) + (1 - \lambda) p(\psi_i \mid G_t)$$

$R(\psi_i)$ is 1 if observed matches sampled, otherwise 0 – blanket reward or penalty for all sampled parameters

Expectation Driven Learning

Expectation Driven Learning (Jarosz 2015) is similar to NPL, but for all parameters estimates $p(\psi_i|d)$ to assign credit/blame

- Loop through all parameters, fix their values at 1 and sample remaining parameters n times
- Use n samples to estimate probability of observed datum given parameter value, $p(d|\psi_i)$
- Invert the conditional with Bayes' Theorem:

$$p(\psi_i|d) = \frac{p(d|\psi_i)p(\psi_i)}{p(d)}$$

- Replace binary reward value with continuous $p(\psi_i|d)$:

$$p(\psi_i \mid G_{t+1}) = \lambda p(\psi_i|d) + (1-\lambda)p(\psi_i \mid G_t)$$

Maximum Entropy Grammar

All parameters correspond to two weighted *constraints*

- i.e. *Comp* corresponds to two constraints; *Comp-L* and *Comp-R*

Structures have a value of 0 or -1 for each constraint: 0 if consistent, -1 if inconsistent

Weights define a probability distribution over structured outputs for a given input constituency[4]

$$p(\omega) = \frac{e^{\psi_\omega \cdot \boldsymbol{w}}}{\sum_{\omega' \in \Omega} e^{\psi_{\omega'} \cdot \boldsymbol{w}}}$$

Probability of a surface string can be calculated by summing over structured forms consistent with that string

[4] See Goldwater and Johnson (2003) for earliest use of MaxEnt phonology

Stochastic Gradient Descent for MaxEnt

SGD is an online, error-driven learning algorithm

- Update only when predicted does not match observed
- Optimize weights to maximize likelihood of the training data

Given an error - compute the *gradient* of the loss with respect to all weights, update opposite the gradient

$$w_{\psi_i,t+1} = w_{\psi_i,t} + \lambda(\overbrace{\psi_i(\omega)}^{\textit{Observed}} - \overbrace{\sum_{\omega' \in \Omega} \psi_i(\omega) p(\omega)}^{\textit{Expected}})$$

Derivation of training data not directly observable, handled with a generalization of Expectation Maximization

- Use current grammar to define probability distribution over possible parses of the observed string, treat distribution as observed [5]

[5] Tesar and Smolensky (1998), Pater et al (2012), Jarosz (2015)

Recap - Characteristics of the three models

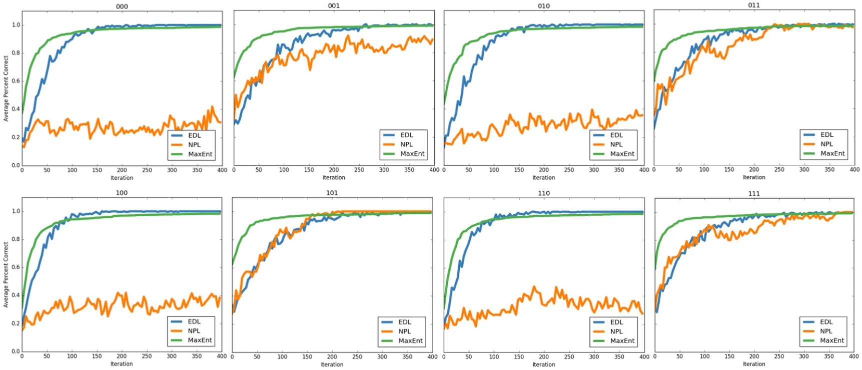
Probabilistic knowledge:

- ✓ NPL (Stochastic Parameter Grammar)
- ✓ EDL (Stochastic Parameter Grammar)
- ✓ MaxEnt (Maximum Entropy Grammar)

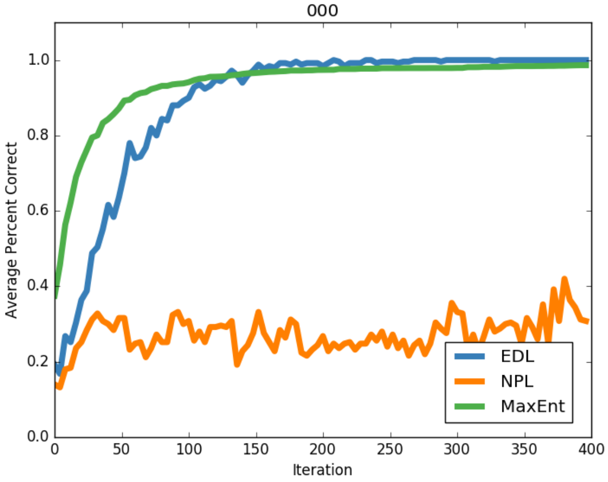
Attribution of credit and blame:

- ✗ NPL (Linear Reward-Penalty Scheme)
- ✓ EDL (reward and penalty based on estimating $p(\psi_i|d)$)
- ✓ MaxEnt (Stochastic Gradient Descent with probabilistic parsing of observed forms)

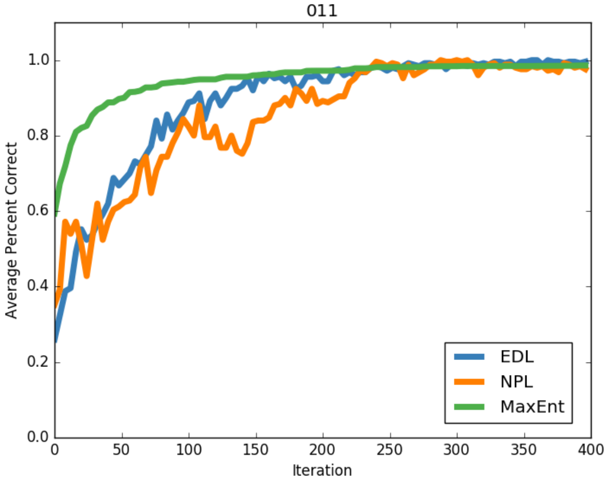
3-parameter learning curves (averaged over 10 runs)



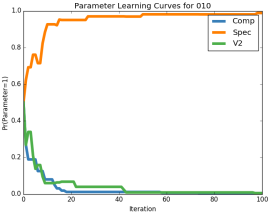
3-parameter learning curves – NPL non-convergence

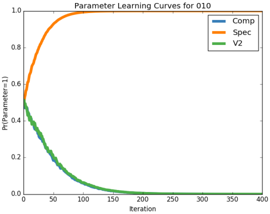


3-Parameter learning curves – NPL convergence

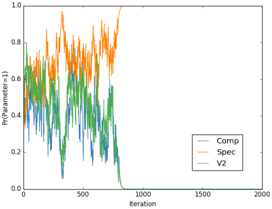


Accumulation of knowledge vs. chancing upon solution



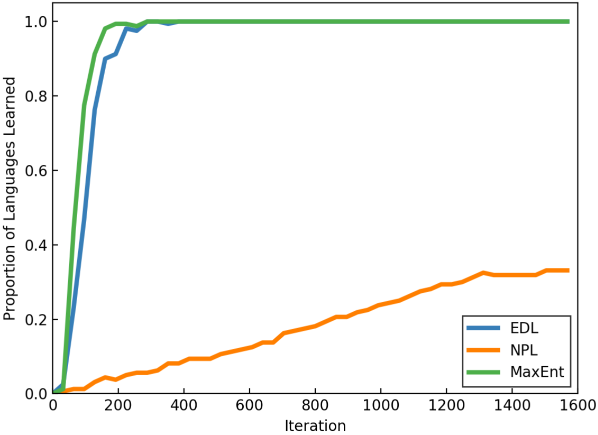


Probability of parameters by iteration for MaxEnt (Left) and EDL (right)



Probability of parameters by iteration for NPL

4-parameter system



Conclusions

Syntactic parameters can be learned from ambiguous data by accumulating information across observations

- There is no need for triggers, on either definition (as unambiguous surface strings, or as unambiguous subtrees)
- This type of learning requires attributing credit and blame to parameters based on how likely they are to produce a form consistent with observations

NPL learns parameters in a small test problem, but does so by chancing upon the solution rather than gradually accumulating information

Expectation Driven Learning with stochastic parameters and Gradient Descent with a Maximum Entropy grammar learn all languages in 3 and 4 parameter systems – making use of ambiguous data and gradient beliefs