Learning syntactic parameters without triggers by assigning credit and blame

Brandon Prickett, Kaden Holladay, Shay Hucklebridge, Max Nelson,
Rajesh Bhatt, Gaja Jarosz, Kyle Johnson, Aleksei Nazarov, and Joe Pater
University of Massachusetts Amherst and University of Toronto

1 Introduction

Parametric approaches to syntax have been widely adopted since the advent of the Principles & Parameters framework in the early 1980s. Parameters are designed to provide a solution to the linguistic facet of Plato's Problem—namely, the logical problem of how language, with potentially infinite well-formed expressions, can be acquired on the basis of the finite data the learner encounters (Chomsky, 1981). Parameters limit the hypothesis space for the learner, reducing syntactic acquisition to the setting of a finite number of innate parameters.

Providing an explicit theory of how parameters are set has proved to be a challenge. Some theories require the presence of triggers—unambiguous data which entail that one or more parameters need to be set a certain way (Gibson & Wexler, 1994). A challenge for models utilizing triggers is that even simple parametric systems generate languages with no unambiguous data points (Gibson & Wexler, 1994); these languages are unlearnable by trigger-based approaches without further stipulations about the learning process.

Other models that avoid the need for triggers require the learning space to be *smooth* (e.g. Yang, 2002). That is, these models require "a correlation between the similarity of grammars and the languages they generate" (Sakas *et al.*, 2017). Unfortunately for these approaches, realistic language data rarely represents a smooth learning problem (Dresher, 1999). As we demonstrate here, even a relatively small number of parameters can lead to systems that are not efficiently learned by such approaches.

In this work, we adapt two domain-general learning algorithms from computational phonology that are not dependent on smoothness or triggers because of their ability to analyze the contributions made by individual parameters. As a baseline, we compare these to Yang's (2002) *Naïve Parameter Learner*, which does not perform this kind of analysis and has been shown to require smoothness (Straus, 2008). We apply all three models of acquisition to syntactic learning, testing them on two simple parametric systems containing headedness and movement parameters. Our results show that Yang's (2002) algorithm is not sufficient for the task, while the other two models succeed—suggesting that future theories of syntactic learning should incorporate analysis of individual parameter settings into their learning.

2 Background

2.1 Hidden Structure and the Credit/Blame Problem

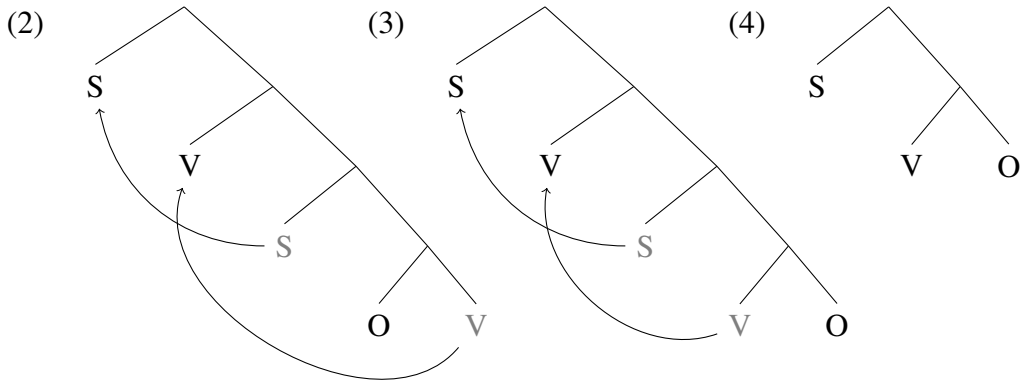
Even in simple parametric systems, a single surface string may be consistent with a number of different parameter settings. The learner must induce the correct settings for their language in the face of this ambiguity. That is, the learner must induce the *hidden structure* underlying each datum it encounters. In the case of syntactic parameters, this includes hierarchical structure, which is absent from the linearized surface string.

For example, consider a learner attempting to acquire the correct values for the parameters defined in (1b), which mimic some of the syntactic variation found in Germanic languages. While English, for example, features SVO (Subject-Verb-Object) word order, and main verbs remain relatively low, the standard analysis of German since Bach (1962) is that it is an SOV language. However, German's SOV ordering is obscured because of verb fronting in matrix clauses (often yielding surface SVO).

(1) a. HEADEDNESS
if 0 : the constituent {O, V} is linearized 'OV'
if 1 : The constituent {O, V} is linearized 'VO'

b. V2
if 0 : there is no movement
if 1 : the structurally highest non-V comes linearly first, and the highest V comes linearly second

Imagine that the learner hears a sentence with an SVO word order. The structure of this sentence could be any of the trees shown in (2-4).



The tree in (2) has a word order of SOV before movement, by virtue of the fact that HEADEDNESS is set to 0. However, since the V2 parameter for this language is set to 1, both the structurally highest non-V (the S) and the V move to higher positions in the tree, and this yields an SVO order. In (3), prior to movement, the ordering of terminals is SVO since HEADEDNESS=1, but since V2 is again set to 1, string-vacuous fronting of S and V occurs. In (4), no movement occurs (V2=0), but HEADEDNESS is set to 1, and so the resulting string is likewise SVO.

In short, the interaction between the HEADEDNESS and V2 parameters results in ambiguity for even simple SVO sentences. Whenever V2 has a value of 1, verbs will occur in the second position of the sentence. This will often obscure the value of HEADEDNESS, since a verb could originate after its object (if HEADEDNESS=0), but be moved to a position preceding the object (if V2=1).

If learning is assumed to follow a general strategy of updating parameters in response to incorrect predictions, the learner can face problems with this kind of ambiguity. For example, if the learner has both of these parameters set to zero (which would result in an SOV sentence) and encounters an SVO sentence, which parameter setting is to blame for the incorrect prediction is completely ambiguous. The learner could blame HEADEDNESS, and push that parameter towards a setting of 1, resulting in an SVO language that generates tree like (4). On the other hand, the learner could blame V2, and attempt to push V2 toward a value of 1, resulting in a V2 language with SOV order before movement (2). Or, the learner could blame both parameters, and move toward being a V2 language with SVO order (3) pre-movement. Nothing in the string SVO provides any evidence about which of these solutions is the correct one.

Likewise, if the learner correctly predicts an SVO word order, it is not clear which parameter should be credited with the success and which (if any) parameters are irrelevant. The issue of assigning responsibility to the correct parameter will be referred to as the *Credit/Blame Problem* (Clark, 1989; Dresher & Kaye, 1990; Dresher, 1999) for the remainder of this paper. There are several ways of handling this problem in language acquisition, and we discuss some of these different methods in §2.2 and §2.3.

2.2 Smoothness and Triggers

One way that models overcome the Credit/Blame Problem is by depending on *triggers* in their training data. Triggers are unambiguous forms that give a model the knowledge it needs to find the correct setting for a specific parameter. That is, in a trigger-based approach to learning, at least one unambiguous datum must exist in a language for each parameter in the grammar. In syntactic modeling, these triggers have taken the form of linearized strings (Gibson & Wexler, 1994) as well as parsed subtrees (Fodor, 1998).

One issue with a dependence on triggers is that in natural language, and even in most simplified learning scenarios, sentences that provide unambiguous information about a parameter setting will not exist in all languages (Gibson & Wexler 1994; although see Fodor 1998 for how using parsed subtrees as triggers can help with this). Yang (2002) proposes a solution to triggers' shortcomings: a probabilistic grammar that can make use of ambiguous and unambiguous data points. He shows that such a learner can acquire the correct syntactic parameter settings, even when there are no data present that completely disambiguate certain parameter values.

However, while this approach does not depend on triggers, it does depend on a smooth learning space (Straus, 2008; Nazarov & Jarosz, 2017). That is, in order for Yang's (2002) algorithm to reliably converge on the correct parameter settings, changing each individual parameter must move the model as a whole closer to the

full solution (Dresher, 1999). If the model finds itself in a situation in which its parameter settings are incorrect, but making any single change would lower its accuracy on the training data, the correct parameter settings can only be stumbled upon by chance (Sakas *et al.*, 2017). Not only is this chance performance qualitatively different than the kind of behavior that infants seem to display, but it is also lacks the efficiency that would be needed to account for human language acquisition (Jarosz & Nazarov, 2019).

This weakness in Yang's (2002) proposal stems from the Credit/Blame Problem discussed in §2.1. That is, while the algorithm is able to use ambiguous data to reward and penalize parameter settings, it fails to consider which parameters deserve credit for correct predictions and which are to blame for incorrect ones. More details about how parameter settings *are* updated in this approach can be found in §3.1, below.

2.3 Using Analysis in Learning

In order to appropriately address the Credit/Blame Problem, the parts of a grammar that are responsible for correct and incorrect predictions must be estimated during learning. We call this process *analysis* and present results in §4 from two models that make use of it in their learning.

One of these methods is *Expectation Driven Learning* (Jarosz, 2015; Nazarov & Jarosz, 2017; Jarosz & Nazarov, 2019), an algorithm that samples from a probabilistic grammar in order to change parameter settings in a way that will increase the probability of the training data, given the model's grammar. The other method that we explore here is Gradient Descent in a Maximum Entropy (Goldwater & Johnson, 2003) adaptation of parametric systems. In this adaptation, parameters are replaced by weighted, opposing constraints, and learning involves using the gradients of a loss function to determine which constraint weights are responsible for a model's incorrect and correct predictions.

Neither of these approaches are guaranteed to converge on the correct solution[1] when hidden structure is present in learning. However, as we show below, using analysis in the process of acquisition allows them to succeed on the simple systems we test, overcoming the dependence on smoothness that limits approaches like Yang (2002).

3 Learning Models

3.1 Naïve Parameter Learner

The *Naïve Parameter Learner* (NPL; Yang 2002) is an incremental, online algorithm for finding a set of parameter values. The NPL does this by assigning probabilities to every parameter setting and sampling from this probability distribution. If a sampled set of parameter values creates output that matches the model's training data, those values are rewarded—however, if an incorrect output is produced, the values are penalized. These penalties and rewards are implemented using the

[1] We define a "correct solution" as a run in which the model has an accuracy equal to or greater than 90% on the training data at the end of learning.

Linear Reward-Penalty Scheme (Bush & Mosteller, 1951), described by the equation in (I), where ψ_i is a parameter setting i, $R()$ is a reward function that updates parameter values, λ is the model's learning rate, and G_t is the grammar at time t.

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

In the NPL, every parameter receives the same reward at each point in learning. If the grammar correctly predicts the datum observed in the model's training data, $R(\psi) = 1$ for all parameters used in the generation of that string, but if the datum is incorrectly predicted, $R(\psi) = 0$ for each of these parameter values.

The probabilistic nature of the NPL allows it to overcome the need for triggers. For example, consider two data points (d_a and d_b) and a two-parameter system (with binary parameters p_1, p_2). Suppose that d_a unambiguously supports a setting of 1 for p_1 but is ambiguous between settings of 1 and 0 for p_2. Similarly, d_b is ambiguous between settings of 1 and 0 for p_1. However, the information d_b gives about p_2 is dependent on the settings of p_1: if p_1 is set to 1, d_b unambiguously supports p_2=1, but if p_1 is 0, d_b unambiguously supports p_2=0.

If the NPL was used to learn the correct settings for this system, at the beginning of training, d_b wouldn't change the probability for any parameter settings, since it would be completely uninformative. However, every time the model saw d_a, it would increase the probability of p_1=1, and as this probability increased, d_b would become a useful data point for raising the probability of p_2=1. While neither d_a nor d_b could act as categorical triggers for p_2, when using a probabilistic approach to learning, these ambiguous data could work together to allow the model to find the correct grammar.[2]

However, while the NPL avoids the need for triggers, as mentioned in §2.2, the model does require smoothness. This is related to the rewards and penalties discussed above: since the full set of parameter settings is given credit/blame when a correct/incorrect prediction is made, the algorithm requires improvements in accuracy as parameter values are updated. Otherwise, the NPL's approach can only succeed by randomly chancing upon the correct solution when it samples from its parameter probabilities (a method of success that becomes exponentially less efficient with each parameter added to the system). This random acquisition of a pattern's solution, rather than gradually accumulating knowledge about the correct parameter settings, is a behavior that we observe in our results in §4 and has been shown to cause the NPL to be slower than chance at converging on a solution, given more complex parametric systems (Jarosz & Nazarov, 2019).

3.2 Expectation Driven Learning

Expectation Driven Learning (EDL; Jarosz, 2015) is another way to find parameter settings using probabilistic rewards and penalties. However, unlike the NPL, each parameter is given its own value for $R(\psi_i)$ at each update, with the value of the reward/penalty being proportional to that parameter setting's probability, given the

[2] In this simple example, the NPL's success depends on the parameters' independence. Parameters are not usually independent in more complex systems, like the example discussed in 2.1.

training data. The algorithm utilizes Bayes' Theorem to calculate this value, using an estimate of the training data's probability given the grammar. This shown in (II), where d is a single training datum and ψ_i is a single parameter setting.

$$\text{(II)} \quad R(\psi_i) = \frac{p(d|\psi_i)p(\psi_i)}{p(d)}$$

To estimate $p(d|\psi_i)$, the model creates a temporary grammar with parameter i categorically set to either 0 or 1 and samples the remaining parameters, according to their probabilities in the current grammar. A fixed number of samples are taken (this is a hyperparameter set by the analyst) and the proportion of matches with d provides the estimate that the algorithm needs. The probability of ψ_i is equal to the current probability for that setting in the model's grammar. The value of $p(d)$ is found by adding the estimates for $p(d|\psi_i)$ and $p(d|\psi_j)$, where ψ_j and ψ_i are opposite settings for the same parameter.

This results in EDL assigning a reward/penalty to each parameter setting that takes into account how well that setting individually predicts the datum of interest (given the rest of the grammar's current probabilities). As discussed in §2.3, by incorporating this analytic step into its updates, EDL is able to avoid the Credit/Blame Problem that arises when other models learn patterns with hidden structure. If a particular parameter setting (i.e. ψ_i) is responsible for either a correct or incorrect prediction, that will be directly incorporated into its reward or penalty.

The way in which EDL uses a model's current grammar to improve its parameter settings is an example of *Expectation Maximization* (Dempster *et al.*, 1977). This method has been successfully applied in phonology to a number of hidden structure problems (e.g. Jarosz, 2006), since it allows updates to be made during learning without a full understanding of a form's underlying structure.

3.3 Maximum Entropy Model

Maximum Entropy Grammar (MaxEnt; Goldwater & Johnson, 2003) is another way to represent the syntactic patterns that we explore here. MaxEnt models define a probability distribution over possible candidates (in our case, sentence structures) based on OT-syle constraint violations (Smolensky & Prince, 1993) and those constraints' weights. In our implementation, violations for each constraint can be any natural number and weights can be any positive, real-valued number (note that while these restrictions are relatively standard in phonology, they are not true of all MaxEnt models).

The constraints we use correspond to the parameters used in the EDL and NPL models. For each parameter, there is a pair of constraints such that each of the pair's two rankings correspond to one of the parameter's values. In a MaxEnt framework, the probability of a structure (σ) according to the model is e raised to the weighted sum of that structure's constraint violations (H_σ, often called Harmony), normalized across all data with the same input (we call the set of data that share the same input I). The equations for the Harmony and probability of a structure are shown in (III) and (IV), respectively.

(III) $H_\sigma = \sum_{c \in C} (c(\sigma) w_c)$ (IV) $p(\sigma) = \frac{e^{H_\sigma}}{\sum_{\sigma' \in I} (e^{H_{\sigma'}})}$

Learning a language in this framework consists of finding the optimal weights to describe the model's training data (Berger *et al.*, 1996). To find these weights, we use stochastic gradient descent. This algorithm uses the difference between a model's expectations and and the observed forms in the training data to make gradual updates to each constraint's weight. This difference is formalized as a loss function (specifically, we used *log likelihood*) and the weight updates are equal to the gradients of this loss function scaled by a learning rate. The negative gradient for any constraint, given an observed form σ, is the difference between the number of observed and expected violations of that constraint, as shown in (V).

(V) $w_{i,t+1} = w_{i,t} + \lambda \left(Obs_i(\sigma) - Exp_i(\sigma) \right)$

$Exp_i(\sigma)$ is the expected number of violations of i, given σ. This is formalized as the weighted sum of the violations of i for all possible forms that share an input[3] with σ (including σ itself), where each datum is weighted by the probability that the current grammar assigns it. This is shown in (VI).

(VI) $Exp_i(\sigma) = \sum_{\sigma' \in I} c_i(\sigma') p(\sigma')$

$Obs_i(\sigma)$ is the number of violations of the ith constraint for the observed form σ. Since the training data only contains surface strings (with no explicit syntactic structures), $Obs_i(\sigma)$ needs to be estimated. To find this estimation, we use a weighted sum of violations for all structures consistent with an observed surface string (we call this set of structures S). The weight that each structure's violations receive is equal to the model's current estimate of that structure's probability.

(VII) $\widehat{Obs_i}(\sigma) = \sum_{\sigma \in S} c_i(\sigma) \frac{p(\sigma)}{\sum_{\sigma' \in S} p(\sigma')}$

This allows updates to be based on the model's current beliefs about the structure of the observed data. Like EDL, this approach to estimating $Obs_i(\sigma)$ is an implementation of *Expectation Maximization* (Dempster *et al.*, 1977) and allows the MaxEnt model to perform the kind of analysis outlined in §2.3. For other examples of Expectation Maximization in a MaxEnt framework, see Pater *et al.* (2012) and the references therein, as well as Johnson *et al.* (2015), Nelson (2019), and Prickett & Pater (2019).

[3]For more on how the inputs and outputs are structured in our examples, see §4.2–3.

4 Learning Simulations

4.1 Learning Task

To test the three models discussed in §3, we set up two simplified typologies: one with three parameters and one with four. Parameters were of headedness and movement varieties, as in (1b). Both of these scenarios abstracted away from individual words and presented the model with input-output pairs consisting of syntactic role labels: S, O, V, and A(dverb). The input data denoted constituency relationships (with no linear order), while the output data gave linear surface order. The settings of the parameters, which determine how input data are mapped to output data, were always hidden from the model during training.

Below we show that for both three- and four-parameter systems, EDL and MaxEnt outperform the NPL because of their ability to assign credit and blame to different parameters.

4.2 Three Parameter Results

Our three-parameter system is based on the system used by Gibson & Wexler (1994), with two headedness parameters (COMP and SPEC; for "complement" and "specifier") and one movement parameter (V2). The definitions for these are given in (5), and the full typology of languages is shown in Table 1.

(5)
- COMP: determines order of O and V before movement
  if 0 : OV
  if 1 : VO
- SPEC: determines order of S and {O, V} before movement
  if 0 : S{O,V}
  if 1 : {O,V}S
- V2:
  if 0: no movement
  if 1: V is linearly second in the output, and the highest (non-V) word is linearly first

We trained the NPL, EDL, and MaxEnt models on each of these languages for 400 iterations, where an iteration was an update based on a single training datum (that is, each model was trained using an online version of its algorithm[4]). Since there were four distinct data points in each language, 400 iterations equated to 100 passes through the full set of training data (referred to as epochs below). EDL and the NPL both used a learning rate of .05 and began with all parameter setting probabilities set to .5. MaxEnt used a learning rate[5] of .5, with initial weight values of 0. Figure 1 shows learning curves for each model, averaged over 10 runs, for all of the languages in the three-parameter typology.

These results demonstrate that the EDL and MaxEnt models converged for all of the languages within 400 iterations. However, the NPL did not converge for all of the languages in this typology, given the same amount of training. The NPL also

[4]Note that in the case of the NPL, only online learning can be performed.

[5]While the learning rate used in MaxEnt does not directly correspond to the learning rate in the other two approaches, the value we chose produced updates on the model's accuracy that were similar in magnitude to EDL's.

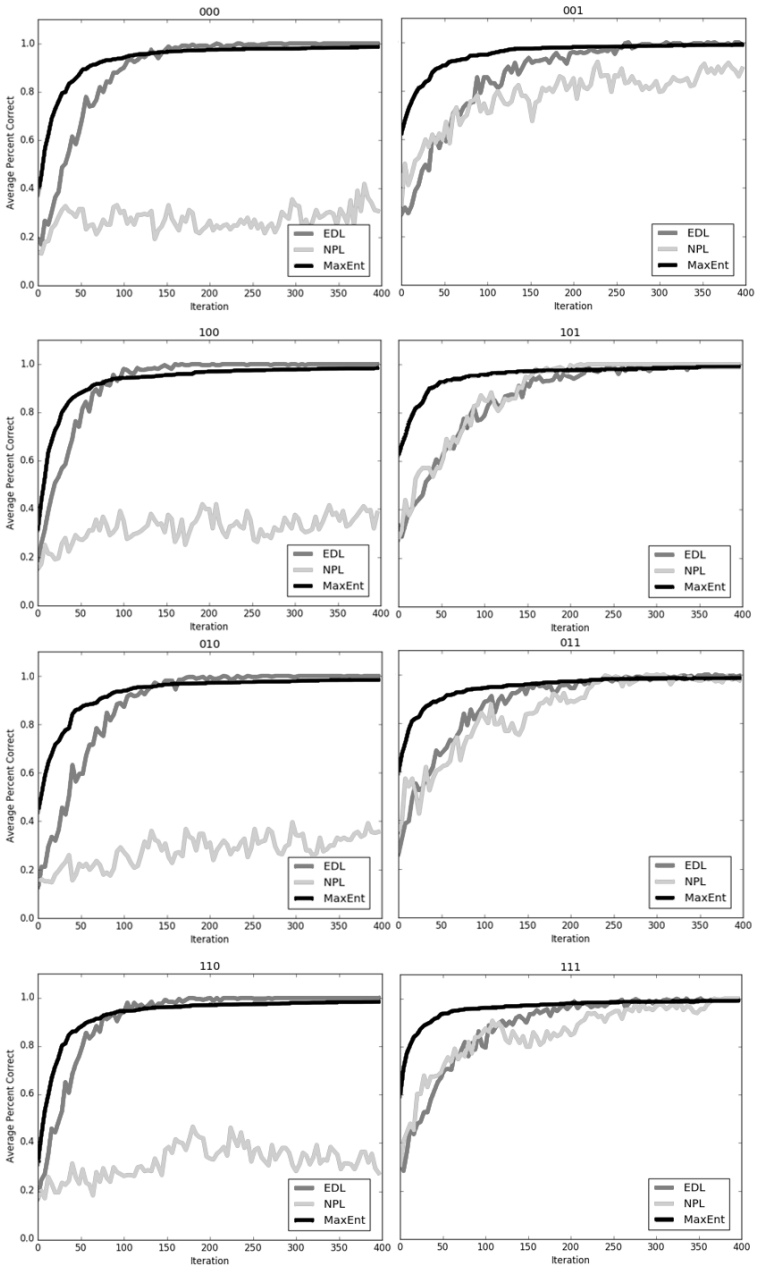


Figure 1: Learning curves for each model in each language for the three-parameter typology.

	000	001	010	011
{S{V}}	SV	SV	VS	SV
{S{OV}}	SOV	SVO	OVS	SVO
{A{S{V}}}	ASV	AVS	AVS	AVS
{A{S{OV}}}	ASOV	AVSO	AOVS	AVOS
	100	101	110	111
{S{V}}	SV	SV	VS	SV
{S{OV}}	SVO	SVO	VOS	SVO
{A{S{V}}}	ASV	AVS	AVS	AVS
{A{S{OV}}}	ASVO	AVSO	AVOS	AVOS

Table 1: Full three-parameter typology. The first column shows the inputs that each model received, the rows containing numbers give the labels for each language, and all other cells show the appropriate outputs, given each input and language. Each digit in the language labels represents a parameter setting, with the first, second, and third digits representing COMP, SPEC, and V2, respectively.

differed from the other models in how it arrived at a successful grammar. While MaxEnt and EDL both gradually accumulated knowledge over the course of learning, the NPL typically underwent an abrupt transition from incorrect to correct parameter settings. The abruptness of the NPL model is obscured in Figure 1 since results are averaged across multiple runs. However, the phenomenon is clearer in Figure 2, where a single representative run for the NPL is shown.

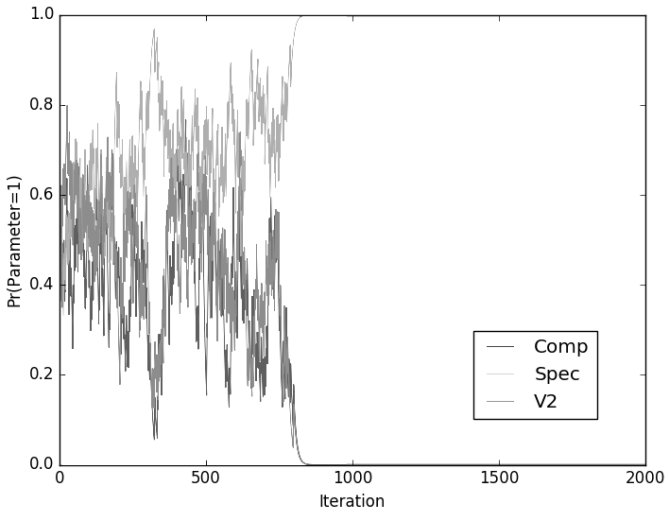


Figure 2: Learning curves from a single run of the NPL. Each line represents a different parameter setting's probabilities. Solution is abruptly discovered around the 800th iteration.

In Figure 2, the NPL model oscillates around chance performance for the majority of learning (with each parameter setting's probabilities hovering near .5) and then stumbles upon the correct answer abruptly around the 800th iteration. This is

in contrast to both EDL and the NPL which gradually change their parameter setting probabilities (or constraint weights) to better match the data, and thus slowly move toward the correct grammars in training. This gradual accumulation of knowledge is shown for individual runs of EDL and MaxEnt in Figures 3 and 4, respectively.

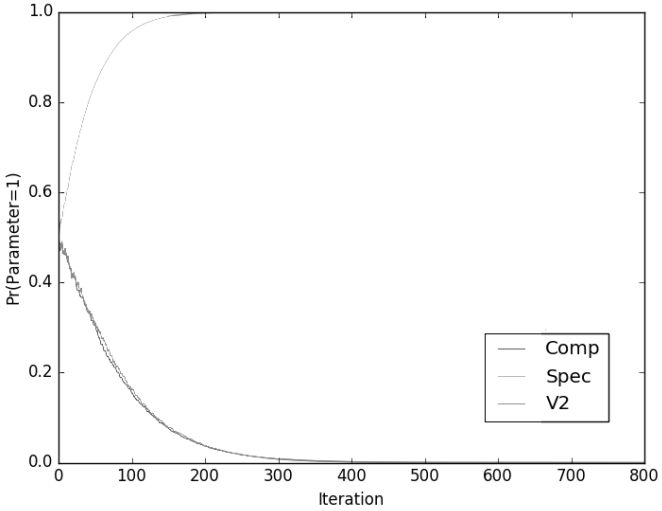


Figure 3: Learning curves from a single run of EDL. Each line represents a different parameter setting's probabilities. Solution is gradually discovered.

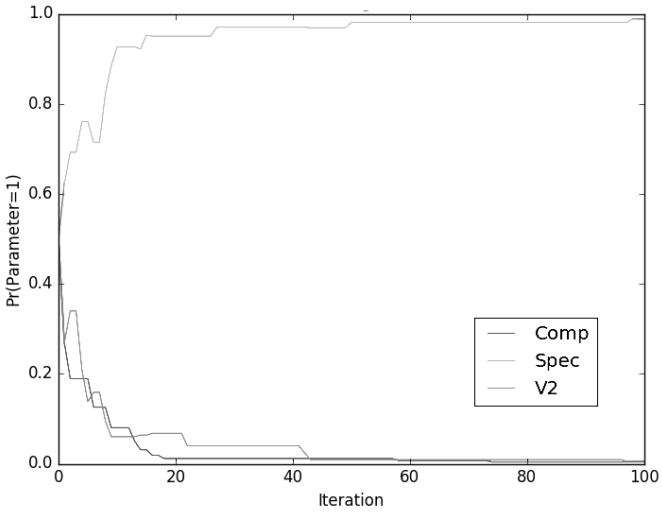


Figure 4: Learning curves from a single run of MaxEnt. Each line represents a the difference between the two constraints relevant to each parameter.

4.3 Four Parameter Results

To further test the models, we implemented a more complex, 4-parameter system that better represented syntactic typology in natural language. To do this, we removed SPEC, and split V2 into three different parameters. The full set of parameters used in these simulations is shown in (6).

(6) • COMP: O is to the (left, right) of V

- V.MOVE: V (is, is not) fronted in all clauses
- V.MOVE.MATRIX: V (is, is not) fronted in matrix clauses only
- TOPIC: some non-V word (is, is not) fronted to first position

The latter three parameters provide a better fit to the variation found in verb-fronting languages. Having both V.MOVE and V.MOVE.MATRIX imitates the contrast between langauges like Icelandic (with verb fronting in all clauses) and German (with fronting in matrix clauses only). A setting of V.MOVE=0 rendered the setting of V.MOVE.MATRIX irrelevant, such that, for example, under V.MOVE=0 and V.MOVE.MATRIX=1, no verb fronting would occur in any clause. The TOPIC parameter allowed for *any* non-verb to be fronted (not just the structurally highest non-verb), as in Germanic languages. If TOPIC was set to 0, no non-V was fronted, and a verb-initial word order could result under V2=1, mimicking Celtic (see McCloskey 1996: §1).

For the sake of space, we do not give the full set of data used in these simulations. However, every combination of parameter values was represented as a language (for a total of 16 languages), and the inputs in (7) were present in each of these languages' training data. (Note that these include auxiliaries; auxiliaries were considered verbs for the purposes of verb fronting). The data included multiple versions of each of these inputs: every datum appeared in matrix and non-matrix clauses, and data that contained subjects, objects, and adverbs featured each of these categories in topicalized and non-topicalized forms (with a total of 32 data points per language).

(7)
- {S{V}}
- {S{OV}}
- {Aux{S{V}}}
- {Aux{S{OV}}}
- {A{S{V}}}
- {A{S{OV}}}
- {Aux{A{S{V}}}}
- {Aux{A{S{OV}}}}

Qualitatively, the MaxEnt and EDL models' performance on this data set was similar to their performance on the 3-parameter system. They converged on the correct solution in all languages, and did so in a relatively small number of iterations. However, the NPL model suffered due to the added complexity and lack of smoothness that the 4-parameter system created. This is illustrated below in Figure 5, which shows how the NPL fails to converge on the majority of languages within 50 epochs (i.e. full passes through the training data).

5 Discussion

5.1 Summary

Our results have shown that both triggers (Gibson & Wexler, 1994; Fodor, 1998) and smoothness (Yang, 2002; Straus, 2008) are not necessary for learning simple syntactic parameter systems, as long as the model being used makes use of analysis while learning. Specifically, two such models (MaxEnt and EDL) could learn all

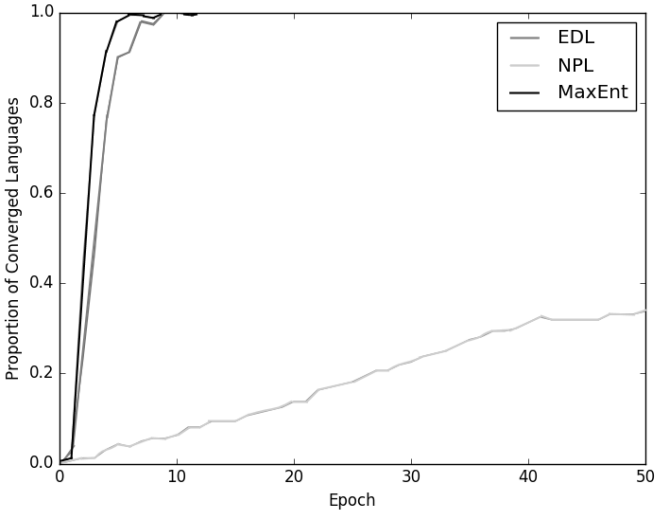


Figure 5: Proportion of languages that each model has converged on, by epoch of training. Convergence is defined as having $\geq$90% accuracy on the training data. Since there are 32 data points in training, 50 epochs represents 1600 updates to the models' parameters.

of the languages in two different parameter-based typologies in an efficient manner, while a model that depended on smoothness could not acquire these simple languages in a reasonable amount of training time. This suggests that the NPL and trigger-based approaches should be reevaluated, since models that do not depend on these properties can successfully learn simple syntactic patterns.

5.2 Future Work

One primary avenue for future research is expanding the parameter space that models are tested on (for example, by using the parameter set in Sakas & Fodor, 2011). Using more realistic training data is another long-term goal of this project. All of the learning simulations we ran in this paper gave models access to all grammatical sentence structures, which is not a likely scenario for language-learning infants in the real-world. Artificial language learning experiments are another way to test models of language acquisition that has had success in the domain of phonology (e.g. Wilson, 2003; Moreton *et al.*, 2017, among others). Comparing these models to human performance on syntactic artificial language learning could be another way to differentiate between competing theories.

Additionally, the fact that EDL and MaxEnt were originally developed for phonology and generalized succesfully to the domain of syntax suggests that these solutions could generalize to other areas of linguistics as well. Applying these learning techniques to morphological and semantic acquisition could be useful steps in seeing how far their usefulness can be extended.

References

Bach, E. 1962. The order of elements in a transformational grammar of german. *Language* 38.263–269.

Berger, A. L., V. J. D. Pietra, & S. A. D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics* 22.39–71.

Bush, R. R., & F. Mosteller. 1951. A mathematical model for simple learning. *Psychological review* 58.313.

Chomsky, N. 1981. *Lectures on Government and Binding*. Foris.

Clark, R. 1989. On the relationship between the input data and parameter setting. In *Proceedings of NELS*, volume 19, 48–62.

Dempster, A. P., N. M. Laird, & D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1–22.

Dresher, B. E. 1999. Charting the learning path: Cues to parameter setting. *Linguistic Inquiry* 30.27–67.

Dresher, B. E., & J. D. Kaye. 1990. A computational learning model for metrical phonology. *Cognition* 34.137–195.

Fodor, J. D. 1998. Parsing to learn. *Journal of Psycholinguistic research* 27.339–374.

Gibson, E., & K. Wexler. 1994. Triggers. *Linguistic inquiry* 25.407–454.

Goldwater, S., & M. Johnson. 2003. Learning ot constraint rankings using a maximum entropy model. In *Proceedings of the Stockholm workshop on variation within Optimality Theory*, volume 111120.

Jarosz, G. 2006. *Rich lexicons and restrictive grammars: Maximum likelihood learning in Optimality Theory*. Johns Hopkins University dissertation. http://roa.rutgers.edu/files/884-1206/884-1206-7-0.PDF.

Jarosz, G. 2015. Expectation driven learning of phonology. *Ms., University of Massachusetts Amherst* . https://people.umass.edu/jarosz/edl_submitted.pdf.

Jarosz, G., & A. Nazarov. 2019. Evaluating domain-general learning of parametric stress typology. *Proceedings of the Society for Computation in Linguistics* 2.383–384.

Johnson, M., J. Pater, R. Staubs, & E. Dupoux. 2015. Sign constraints on feature weights improve a joint model of word segmentation and phonology. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 303–313.

McCloskey, J. 1996. On the scope of verb movement in irish. *Nature Language & Linguistic Theory* 14.47–104.

Moreton, E., J. Pater, & K. Pertsova. 2017. Phonological concept learning. *Cognitive science* 41.4–69.

Nazarov, A., & G. Jarosz. 2017. Learning parametric stress without domain-specific mechanisms. In *Proceedings of the Annual Meetings on Phonology*, volume 4.

Nelson, M. 2019. Segmentation and ur acquisition with ur constraints. *Proceedings of the Society for Computation in Linguistics* 2.60–68.

Pater, J., K. Jesney, R. Staubs, & B. Smith. 2012. Learning probabilities over underlying representations. In *Proceedings of the twelfth meeting of the Special Interest Group on Computational Morphology and Phonology*, 62–71. Association for Computational Linguistics.

Prickett, B., & J. Pater. 2019. Learning Hidden Structure with Maximum Entropy Grammar. *27^{th} Manchester Phonology Meeting* . https://github.com/blprickett/Hidden-Structure-MaxEnt.

Sakas, W. G., & J. D. Fodor. 2011. Generating CoLAG Languages Using the 'Supergrammar'. Technical report, City University of New York. http://www.colag.cs.hunter.cuny.edu/pub/COLAG_2011_supergrammar.pdf.

Sakas, W. G., C. Yang, & R. Berwick. 2017. Parameter setting is feasible. *Linguistic Analysis* 41.

Smolensky, P., & A. Prince. 1993. Optimality theory: Constraint interaction in generative grammar. *Optimality Theory in phonology* p. 3.

Straus, K. J. 2008. *Validation of probabilistic model of language acquisition in children*. Northeastern University dissertation. https://repository.library.northeastern.edu/files/neu:1578/fulltext.pdf.

Wilson, C. 2003. Experimental investigation of phonological naturalness. In *Proceedings of the 22nd west coast conference on formal linguistics*, volume 22, 533–546. Citeseer.

Yang, C. D. 2002. *Knowledge and learning in natural language*. Oxford University Press on Demand.