Naive Parameter Learning for Optimality Theory - The Hidden Structure Problem[*]

Gaja Jarosz

Yale University

1. Introduction

This paper introduces a new learning algorithm for Optimality Theory (OT; Prince and Smolensky 1993/2004) that can cope with structural ambiguity, a kind of hidden structure. The paper discusses results of simulations with this new algorithm as well as a random baseline on a large test set with structural ambiguity that has been used to evaluate several existing learning algorithms.

1.1 Background

There exist a number of provably correct learning algorithms for Optimality Theory and closely related theories. These include *Constraint Demotion* (CD; Tesar 1995, *et seq.*), a family of algorithms for classic OT. For Harmonic Grammar (Legendre, Miyata and Smolensky 1990; Smolensky and Legendre 2006) and related theories (e.g. maximum entropy), there is *Stochastic Gradient Ascent* (SGA; Soderstrom, Mathis and Smolensky 2006, Jäger 2007). There is also the *Gradual Learning Algorithm* for Stochastic OT (GLA; Boersma 1997), which works well in most cases but is known not to be correct in the general case (see e.g., Pater 2008). The success of these algorithms (and correctness proofs in the case of CD and SGA) relies on the assumption that learners are provided with full structural descriptions of the data, including prosodic structure as well as underlying representations, which are not available to the human learner.

All of the above learning algorithms are *error-driven*: the learner's grammar is updated whenever an error is produced. The update involves comparing the constraint violations of the learner's output and the observed output to identify how the rankings or weightings of constraints should be adjusted in order to increase the harmony or

[*] The development of this work has benefited by comments from Edward Flemming, Adam Albright, Paul Smolensky, Colin Wilson, Bruce Hayes, and audiences at NELS 40 and LSA 2010, where portions of this work were presented. The author would like to thank Bob Frank for particularly helpful discussion, Joe Pater for sharing the grammar and distribution files for the TS2000 test set, and Paul Boersma and Bruce Tesar for creating them. Any errors are the sole responsibility of the author.

likelihood of the observed output relative to the error. Hidden structure poses a challenge for error-driven learning because it obscures the constraint violations of the observed output. Structural ambiguity, such as footing and syllabification, obscures the violations of any constraints that reference that structure whether they assign that structure or simply depend on it. For example, if the overt form is a tri-syllabic word with medial stress, there are (at least) two analyses: one with an initial iambic foot and another with a final trochaic foot. Without the footing, the violations of constraints like TROCHAIC and IAMBIC are unknown. If underlying representations are unknown, then the violations of faithfulness constraints for the observed output are likewise unknown. Thus, without the full structural descriptions, the vector of constraint violations for the observed output required to calculate the update to the grammar is not apparent from the learning datum.

Within OT, learning without full structural descriptions has been a topic of ongoing work since at least Tesar (1997a) and Tesar and Smolensky (1998). In order to apply error-driven learning in the presence of structural ambiguity, Tesar and Smolensky (1998) proposed *Robust Interpretive Parsing* (RIP), which provides an educated guess, based on the current constraint ranking, about the structure of the observed output, enabling the grammar updates to be calculated in the usual way. Tesar and Smolensky (2000) presented simulation results for this procedure on a large metrical phonology test set with structural ambiguity (henceforth referred to as TS2000), which is discussed in Section 3. They found that this learning procedure learned 60.5% of the languages in the system correctly when starting from an unranked initial grammar. In subsequent work, Tesar (1997b, 2004) proposed a procedure, *Multi-recursive Constraint Demotion* (MRCD), that keeps track of a (limited) set of viable grammatical hypotheses simultaneously and is guaranteed to converge on a correct ranking in the presence of structural ambiguity. Based on simulations with similar metrical phonology systems, Tesar showed that the number of hypotheses that must be considered during learning is relatively small and that learning is quite efficient. Nonetheless, MRCD does involve additional complexity beyond that of standard CD, GLA, and SGA, which maintain just one grammatical hypothesis at a time. Boersma (2003) and Boersma and Pater (2008) applied Tesar and Smolensky's (1998) Robust Interpretive Parsing approach to the GLA and SGA algorithms. Boersma and Pater (2008) also replicated Tesar and Smolensky's simulations with RIP/CD and presented results of simulations with RIP/GLA and RIP/SGA on the TS2000 test set. Averaged over multiple runs, the algorithms in Boersma and Pater's simulations successfully learned between 47% and 89% of the languages in the system, with RIP/SGA for noisy Harmonic Grammar getting the best performance. In sum, while a variant of CD that keeps track of multiple hypotheses simultaneously is guaranteed to find a correct ranking, the performance of RIP/CD, RIP/GLA, and RIP/SGA, which maintain one grammatical hypothesis at a time, is more variable, with none of them achieving perfect performance in the presence of structural ambiguity.

1.2 Overview

The present paper describes a new online learning algorithm, the *Naive Pairwise Ranking Learner* (NPRL), an adaptation of Yang's (2002) *Naive Parameter Learning* to OT, which achieves a 100% success rate on the TS2000 system. Like the earlier algorithms, NPRL is an online algorithm that processes a single data point at a time. Like RIP/CD, RIP/GLA, and RIP/SGA, NPRL maintains one grammatical hypothesis at a time. In contrast to all the earlier algorithms, NPRL sidesteps the issue of structural ambiguity altogether because its updates do not depend on it. Also, NPRL is not error-driven: it makes adjustments to the current grammar hypothesis not only when the current hypothesis results in an error but also when the learner's hypothesis results in an output that matches the data. Despite this apparent success, further investigations into the algorithm's performance, however, reveal unusual behavior reminiscent of random search. This motivates investigation into a truly naive learner, random search, which provides a baseline for performance on this test set. It turns out that the baseline beats the performance of NPRL (getting 100% accuracy in less time) as well as the performance of RIP/CD, RIP/GLA, and RIP/SGA on this test set.

The remainder of the paper is structured as follows. Section 2 presents Yang's (2002) Naive Parameter Learner and the Naive Pairwise Ranking Learner, its application to OT. Section 3 presents the simulations with NPRL and experiments with the random baseline as well as related discussion. Finally, Section 4 discusses the implications of these findings for the evaluation and development of learning algorithms in OT, and Section 5 presents conclusions.

2. Naive Pairwise Ranking Learning

The Naive Pairwise Ranking Learner is an adaptation of Yang's (2002) *Naive Parameter Learning* (NPL) to Optimality Theory (see also Pearl 2009 for an application of NPL to metrical phonology). NPL is designed for a parametric grammatical framework in which each parameter is associated with a probability distribution over values it can take on. Given some data, NPL randomly selects a grammar by selecting values of each of the parameters according to their specified probabilities. If the resulting grammar successfully generates the data, all parameter values of the selected grammar are rewarded, and if the grammar does not successfully generate the data, all parameter values are penalized. Yang discusses a particular update rule called the linear reward-penalty scheme (Bush and Mosteller 1951; Yang 2002), which rewards and penalizes values for binary parameters according to the formulas in (1), where *i* refers to the iteration. The magnitude of the update is modulated by a learning rate γ, and the updated probability for the opposing parameter value is simply $1\text{-}p_{i+1}$ for binary parameters. Thus, if a selected grammar is successful, all the participating parameter values are rewarded according to (1)a, and if it is unsuccessful, all participating parameter values are penalized according to (1)b.

(1) Linear Reward-Penalty Scheme (Bush and Mosteller 1951; Yang 2002)
   a. Reward parameter value p: $p_{i+1} = p_i + \gamma(1 - p_i)$
   b. Penalize parameter value p: $p_{i+1} = (1 - \gamma)p_i$

Naive Pairwise Ranking Learning, the adaptation of NPL to OT, is made possible by representing a ranking in terms of a set of binary parameters specifying the relative ranking between each pair of constraints. For example, the ranking *A* » *C* » *B* corresponds to the binary parameter settings of *A* » *B* set to 1, *A* » *C* set to 1, and *B* » *C* set to 0. As in NPL, each parameter setting is actually associated with a probability. Thus, for each pair of constraints (*a, b*), the grammar specifies a probability (whose complement is the probability associated with (*b*, *a*)). This probability specifies for each pair of constraints the relative preference for one relative ranking over the other and determines how frequently one relative ranking versus the other will be used when selecting a random ranking. Specifically, ranking selection proceeds iteratively by selecting a pair of unranked constraints, selecting their relative rank according to the probability specified by the grammar, setting any pairwise rankings implied by the newly set relative ranking, and repeating until all pairwise rankings are set to 1 or 0, corresponding to a total ranking. Thus, the probabilities associated with each pair of constraints together with this sampling procedure determine a probability distribution over the set of total rankings. This grammatical representation and sampling procedure are also discussed by Jarosz (2009) in the context of maximum likelihood learning of phonology.

Given this way of representing stochastic rankings and of selecting total rankings, it is straightforward to apply Yang's algorithm. NPRL processes one overt form at a time like the OT learning algorithms discussed above. For each overt form, NPRL generates a random ranking from its current stochastic grammar and uses it to generate an output for the current data point. If the overt portion of the learner's output matches the observed output, all pairwise ranking probabilities corresponding to the selected ranking are rewarded. If the overt portion of the learner's output does not match the learning datum, all pairwise ranking probabilities corresponding to the selected ranking are penalized. This procedure is summarized in (2).

(2) Naive Pairwise Ranking Learner
   a. Sample a data point *d*
   b. Generate a random ranking *r* using the current parameter settings
   c. Produce learner's output *o* for *d* using *r*
   d. If $o = d$, reward all parameter values in *r* according to (1)
   e. If $o \neq$ d, penalize all parameter values in *r* according to (1)

Unlike the earlier OT learning algorithms, NPRL does not attempt to reward or penalize *particular* constraints, which is what makes it "naive". If the ranking results in a mismatch with the observed data, *all* the pairwise ranking probabilities corresponding to that ranking are penalized. For example, if a ranking of *A* » *B* » *C* results in a mismatch, the parameter values for *A* » *B*, *B* » *C*, and *A* » *C* are decreased. No effort is made to determine which constraint rankings are to blame for the error. Likewise, if the ranking yields a match, *all* pairwise ranking probabilities corresponding to the selected ranking are rewarded. Thus, NPRL is not error-driven since learning occurs after matches as well as after errors. As discussed above, structural ambiguity poses a challenge for error-driven learners since the constraint violations of the observed output are obscured. In contrast, structural ambiguity poses no special challenge for NPRL because NPRL is (blissfully) unaware of it. Structure is irrelevant to the calculation of matches and mismatches, which considers only the overt portions of the outputs. In the realm of stress, for example, matches occur whenever the stress contours are identical, irrespective of what footing was responsible for those stress contours. In other words, NPRL sidesteps the issue of structural ambiguity entirely because the calculation of the update does not depend on structure.

3. Simulations

3.1 The Language Data and Experimental Set-up

This section presents the results of a number of simulations, all of which are carried out on the TS2000 test set developed by Tesar and Smolensky (2000). This test set consists of 124 constructed languages that can be modeled by the set of 12 metrical structure constraints shown in (3). Each language is a set of 62 words, which are sequences of light or heavy syllables (e.g. [L H L]) ranging in length between two and seven syllables. Each language associates a particular foot structure and stress pattern with each word (e.g. [(L0 H2) L0]), which includes markings for primary stress (1), secondary stress (2), and unstressed (0). The learner, however, is exposed only to the overt stress patterns (e.g. [L0 H2 L0]) and must infer the ranking of constraints and the footing that underlie these patterns.

(3) Constraints (Tesar and Smolensky 2000)

FOOTBIN	Each foot must be either bimoraic or bisyllabic
PARSE	Each syllable must be footed
MAIN-RIGHT	Align head foot with right edge of the word
MAIN-LEFT	Align head foot with left edge of the word
IAMBIC	The final syllable of a foot must be the head
NONFINAL	The final syllable of a word must not be footed
WSP	Each heavy syllable must be stressed
WORD-FOOT-RIGHT	Align right edge of the word with a foot
WORD-FOOT-LEFT	Align left edge of the word with a foot
FOOT-NONFINAL	A head syllable must not be final in its foot
ALL-FEET-RIGHT	Align each foot with right edge of the word
ALL-FEET-LEFT	Align each foot with left edge of the word

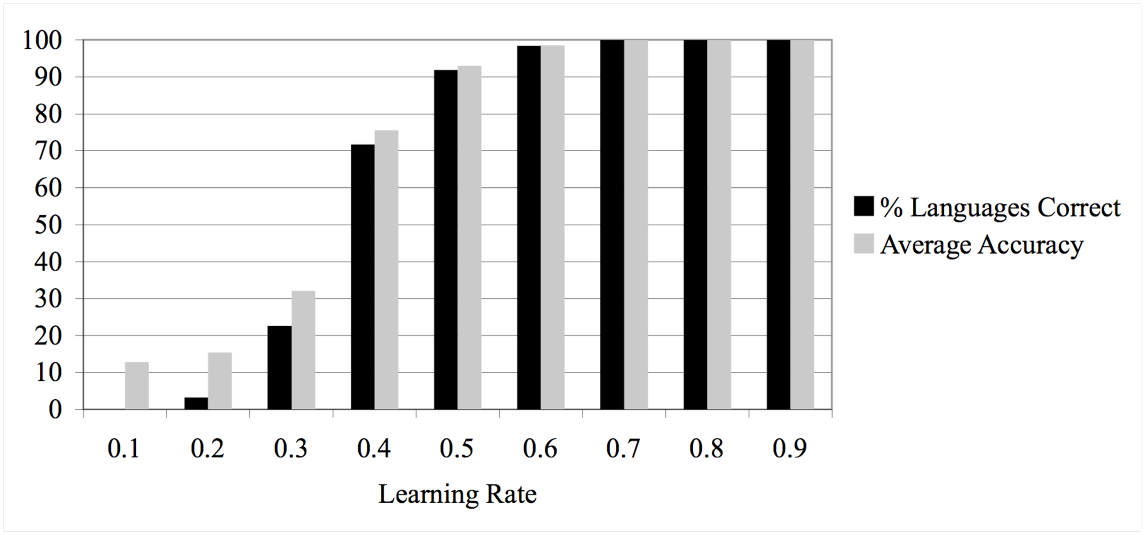
Following Boersma and Pater (2008), the learning algorithm was allowed a maximum of 1,000,000 iterations, where each iteration corresponds to the processing of one overt form, to learn each language. Learning of a language was deemed successful when the algorithm converged on a ranking that correctly generated the stress contours for all words in the language. A finer-grained measure of performance was also calculated, representing the average accuracy across all the words in a language. In order to gauge the accuracy over time and of the final-state grammars, the grammars of the learners were queried every 100 iterations and an accuracy for each word was determined by randomly sampling from the current grammar 100 times. All NPRL simulations begin from the same initial state, with all parameters set to 0.5, corresponding to a maximally unbiased grammar under which all rankings are equally likely.

3.2 NPRL Simulation Results

Since NPRL is a non-deterministic algorithm, results reported are based on 10 separate runs for each language. Results are summarized in Figure (4), which compares the performance of the algorithm for different learning rates ranging between 0.1 and 0.9. With high learning rates between 0.7 and 0.9, the success rate is 100% – all the languages were successfully learned on all the runs. Furthermore, learning is (apparently) fast: while there are 12! (i.e. 479,001,600) different total rankings in this system, the algorithm converged on a correct ranking after an average of 16,004 iterations (with a learning rate of 0.8). Notably, the 100% success rate beats the performance of existing learning algorithms: RIP/CD, RIP/GLA, and RIP/SGA.

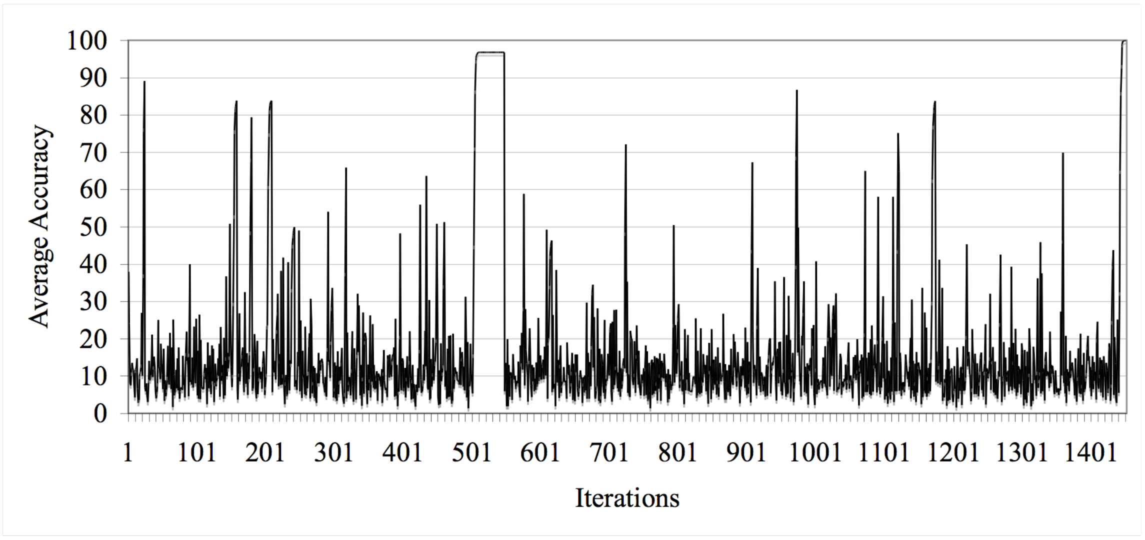
As shown in (4), however, the performance of the algorithm varies dramatically depending on the learning rate. For a learning rate of 0.1, average accuracy falls to just 12.8%, with none of the languages learned correctly in full. Since none of the languages are ever learned correctly on these runs, all runs continue for the maximum of 1,000,000 iterations. As the low accuracy suggests, the final grammars for these runs are not very close to correct. In fact, for the runs with low learning rates, no learning seems to occur at all. The grammars after 1,000,000 iterations closely resemble the initial state grammar, with nearly all parameters set to values between 0.4 and 0.6. Overall, there is no consistent net effect of the rewards and punishments for runs with low learning rates.

(4) Performance of NPRL at Different Learning Rates



What, then, explains the successful learning with higher learning rates? Learning with such high learning rates is very jumpy. Consider the following scenario. The learner begins with all parameters set to 0.5. Suppose the learner then samples a data point and a ranking that results in a match. With a learning rate of 0.7, the successful parameter values will be updated to 0.85. At this point the previous ranking is very likely to be selected, and if it is, and if it is successful for the next data point, the parameter values will be updated to 0.96, which is close to a categorical grammar. If this selected ranking is indeed a correct ranking that works for all the data, the learner will likely converge on the total ranking very quickly. However, if the ranking does not work for all the data, it will eventually result in a mismatch. After a mismatch, a setting of 0.96 will jump all the way to 0.29, essentially flipping to the opposite grammar in one iteration. With higher learning rates, these updates are even more extreme. This jumpiness can be observed by querying the grammar after each iteration to examine how the average accuracy changes over time. A plot showing average accuracy over time for a representative run with a learning rate of 0.7 is shown in (5). If the learning algorithm were gradually moving toward a correct grammar, this would be reflected in the changes in the average accuracy over time. On the contrary, the plot reveals that this algorithm is jumping around the space in an apparently random fashion, getting close to a correct grammar on many occasions and then jumping back to a grammar no better than where it started.

(5) Average Accuracy over Time for a Sample Run with $\gamma = 0.7$

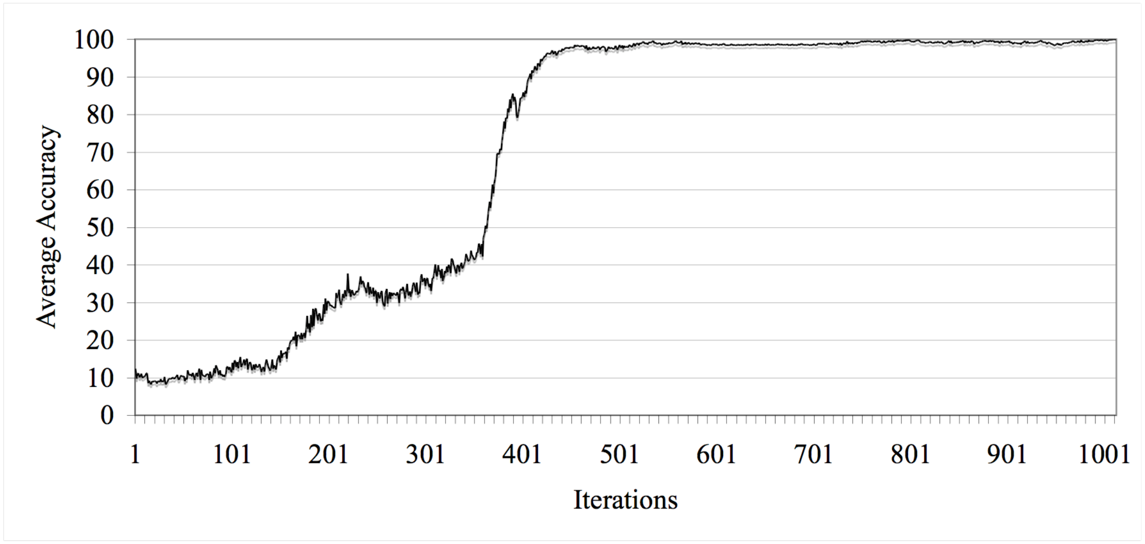


This jumpy behavior together with the inconsistency of learning at lower learning rates suggests the possibility that the success of NPRL with high learning rates is simply an artifact of the extent to which NPRL mimics random search. To explore this possibility, Section 3.4 discusses simulations with just such a learner, a random baseline. Before turning to the random baseline, however, the next section examines the relative contributions of the rewards and punishments to NPRL's performance.

3.3 NPRL Without Punishment

As the discussion above suggests, NPRL's naive punishments prevent any gradual progress: when the learner gets to a grammar that is close to correct but cannot generate all the data, there will eventually be an error. Even though the grammar is close to correct, the learner does not capitalize on the progress it has made since the error results in a punishment that pushes the grammar away into unrelated territory. Indeed, with respect to gradual learning, a variant of NPRL with reward updates only (no updates after errors) behaves more sensibly. Figure (6) shows average accuracy calculated after each iteration for a sample run of NPRL with reward updates only. It is clear from this behavior that reward updates result in a gradual accumulation of grammatical preferences, resulting in gradual learning.

(6) Accuracy over Time for a Successful Run of Reward Only NPRL (γ = 0.1)



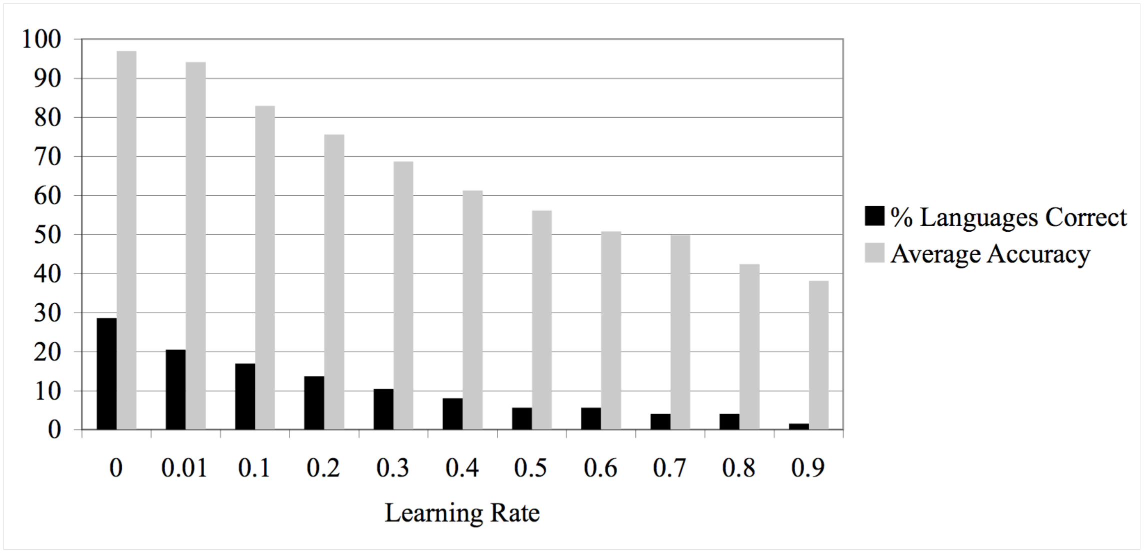
Unfortunately, Reward Only NPRL exhibits such sensible learning curves only a fraction of the time because most of the time it does not converge on a correct grammar. Its performance is summarized in (7). The best performance is at a learning rate of 0.001, with only 28.6% of the languages learned correctly[1]. Notably, performance of Reward Only NPRL improves somewhat with *decreasing* learning rates, which is opposite of the pattern for NPRL. Its performance for high learning rates between .7 and .9 is very poor, with less than 10% of the languages being learned successfully. Thus, it does not seem that the successful performance of NPRL can be attributed to the reward updates. On the other hand, Punishment Only NPRL does not make any sense because it cannot converge on any grammar (unless it begins the search at the correct grammar). The only update mechanism it has is to jump (partially) away from the current grammar, which never results in a correct grammar for these languages generated from total rankings. Simulations with learning rates between 0.1 and 0.9 confirm that Punishment Only NPRL learns none of the languages. Thus, it is the particular combination of rewards and punishments that allows NPRL to succeed at high learning rates. This further supports the possibility that NPRL succeeds by mimicking random search.

Before turning to the random baseline, one observation regarding Reward Only NPRL warrants further discussion. While the proportion of languages learned successfully is very small, average accuracy reaches almost 97%, indicating that even when the learner does not converge on a correct grammar, its final grammar is very close to correct. Thus, NPRL reward updates appear to be doing something right. There is a

[1] At this low learning rate the time to converge on a correct grammar is already almost 250,000 iterations on average so performance on lower learning rates is not reported.

good reason for this. Reward updates are related to the updates of a well-known learning algorithm, Expectation Maximization (EM; Dempster, Laird, and Rubin 1977), for which there are mathematical proofs of convergence. EM updates are conditioned on successful generation of each datum, whereas NPRL rewards occur only when the data is successfully generated by the current grammar. The problem with the reward only version of NPRL is that nothing forces the learner to be responsible for all the data. In related work, Jarosz (2009) proposes a family of learning algorithms for the grammatical representation used here, including two online variants, using updates based on EM. Although in-depth discussion of these algorithms is beyond the scope of this paper, there is reason to be optimistic about their prospects. Thus, although the performance of reward-only NPRL is quite poor, the basic learning strategy of rewarding successful generation of the data should not be dismissed.

(7) Performance of Reward Only NPRL Across Learning Rates



3.4 A Random Baseline

Baselines are standard practice in computational linguistics in general, but explicit evaluation with baselines has played little role in computational modeling work within OT. Baselines give a sense of the difficulty of a learning problem and provide a reference point for interpretation of quantitative results. For example, an accuracy of 85% on part-of-speech tagging may sound pretty good until one learns that simply guessing the most likely tag for each word yields an accuracy of around 90% (Charniak et al 1993).

To investigate the successful performance of NPRL with high learning rates and to provide a baseline for learning algorithms on this test set, this section presents the results of simulations with a baseline relying on random search. The learning strategy is very simple: select a total ranking at random (assuming all rankings are equally likely). For each data point, determine if the ranking generates it. If so, do nothing; if not, choose a new total ranking randomly. Basically, learning entails picking rankings at random until errors are no longer produced. Such a procedure is generally considered implausibly slow for hypothesis spaces of any complexity so it may come as a surprise that the random baseline learns 100% of the languages in the TS2000 system. More specifically, given a maximum of 1,000,000 iterations (where an iteration corresponds to the processing of one data point, as before), the random baseline learns all the languages on all runs (with 10 runs for each language). Furthermore, even though there are nearly 500 million total rankings of constraints in this system, the random baseline converges on a correct ranking in an average of 10,025 iterations. Please note that the counting of iterations is very generous: it not only includes iterations on which a new random ranking was selected but also iterations resulting in a match, with no change in ranking. In sum, the random baseline gets perfect performance, beating the performance of RIP/GLA, RIP/SGA, and RIP/CD, and finding a correct ranking in less time than NPRL with high learning rates!

How can this be? Although there are nearly 500 million total rankings, the performance of the random baseline indicates that the number of distinct languages is much, much smaller[2]. In other words, each distinct stress pattern for the 62 overt forms that comprise a language is consistent with thousands of total rankings on average. Learning is successful when the learner selects any one of these compatible rankings. In sum, the baseline indicates that it is not hard to achieve 100% accuracy on this test set given one million iterations.

3.5 Discussion of Simulations

With regard to NPRL, the success of the random baseline suggests that the performance of NPRL with high learning rates can be attributed to its jumpy, random behavior. Because of the very large magnitude of the updates, the learner is able to jump around the search space very quickly. The jumps made by NPRL are not totally random, however, since they depend stochastically on the state of the grammar on the previous iteration. Apparently, totally random jumps are more productive than the informed jumps of NPRL since the random baseline finds a correct ranking in fewer iterations on average. In sum, NPRL's effectiveness depends on the degree to which it mimics random search, but it is never quite as effective as true random search presumably because it does not jump around as quickly or as randomly.

Whatever the cause of NPRL's successful learning, both NPRL and the random baseline beat the performance of leading learning algorithms like RIP/GLA and RIP/SGA on this test set. The next section discusses the broader implications of this result.

4. General Discussion

To summarize, this paper introduces a new learning algorithm, NPRL, that outperforms RIP/GLA, RIP/SGA, and RIP/CD given the same learning conditions. Analysis of NPRL

[2] Tesar (2004) found that a system like this one minus the two Word-Foot constraints generated 2,140 distinct structural descriptions for the set of 62 overt forms, much less than the 10! number of total rankings. Some of these 2,140 languages were weakly equivalent (assigned the same stress patterns), however, and the number of distinct stress patterns was smaller still, 1,527. The system examined here is a bit larger so the number of distinct languages must likewise be somewhat larger (though no more than a few thousand).

suggests that its successful performance can be attributed to an approximated random search. True random search outperforms all of the above algorithms, achieving perfect performance in fewer iterations on average than NPRL. This raises a number of questions regarding the evaluation and development of computational models of phonological learning. The following discussion addresses some of these questions.

4.1 Regarding Performance on TS2000

Some authors have suggested that an algorithm's failure to learn can be a good thing. Boersma (2003) suggests that a language's unlearnability can provide a formal explanation for typological gaps. The present discussion assumes, in contrast, that via factorial typology, a constraint set defines the set of possible (and learnable) human languages. If a constraint set predicts bizarre languages, this is a problem with the constraint set.

For learning problems within classic OT, where the learning problem is taken to involve finding a total ranking of constraints, rejecting the random baseline comes down to a matter of speed. For a given constraint set CON, the search space of possible rankings is large (|CON|!), but it is finite. Therefore, it is possible to construct baseline learners, such as the random baseline or complete enumeration, that are guaranteed (eventually) to find a correct ranking, assuming one exists. Thus, within classic OT the baseline is 100%, and the only way to beat it is to get 100% more quickly[3]. The test set developed by Tesar and Smolensky (2000) and explored here defines a learning problem within classic OT. Each of the target languages in the system can be generated by a total ranking of the constraints. The simulations with the random baseline presented here establish what "more quickly" means for this test set. Specifically, to beat the random baseline on this test set, an algorithm must achieve 100% performance in fewer than 10,025 iterations on average, where an iteration consists of evaluating a single datum against a single grammatical hypothesis.

Multi-recursive Constraint Demotion (Tesar 1997b, 2004), which is guaranteed to converge on a correct ranking, and NPRL, which achieves 100% on the test set, are the only contenders for beating the random baseline in this sense. As shown above, however, NPRL takes more time on average than the random baseline to arrive at a correct ranking. Therefore NPRL does not beat the random baseline. MRCD on the other hand, is able to take advantage of the internal structure of OT by ruling out hypotheses that are inconsistent with previous ranking information, dramatically limiting the combinations of structural analyses that must be considered. In particular, Tesar (1997b) shows that for each of the languages in the TS2000 system MRCD finds a correct ranking within 160 applications (median of 50) of Constraint Demotion. Although, the number of applications of CD is not comparable to the iterations counted for the random baseline

[3] Eisner (2000) shows that in the general case, no learning algorithm for OT can be guaranteed to find a correct ranking faster than enumeration without exposure to full structural descriptions. However, constraints within OT do not have arbitrary definitions, and thus actual performance will depend crucially on the constraint set.

since it does not count the processing of data resulting in matches, the results reported by Tesar (2004) suggest that counting the number of matches is not going to dramatically increase the numbers for MRCD (at least not by a factor of 200). Thus, although a comparable count for MRCD should be confirmed, it seems likely that MRCD beats the random baseline for this test system.

What about the performance of the earlier algorithms that maintain one grammatical hypothesis at a time, RIP/GLA and RIP/SGA? Given one million iterations, their performance ranges between 55% and 89%, depending on the choice of evaluation and learning algorithm (Boersma and Pater 2008). Thus, with regard to accuracy, these algorithms do not beat the baseline. Boersma and Pater (2008) do not report how long on average it takes for the learners to settle on a correct grammar. Thus, it is not clear how the speed of learning compares to the random baseline. Further work comparing the performance of these algorithms with the performance of baselines such as the one considered here is needed.

4.2 Beyond TS2000

This paper focuses on the problem of learning total rankings in the presence of structural ambiguity. However, a principle advantage of algorithms such as the GLA and SGA, when paired with probabilistic extensions of OT such as Stochastic OT and noisy Harmonic Grammar, is their ability to learn languages with free variation and deal with noisy data (see e.g., Boersma and Hayes 2001). If the learning problem is extended to such cases, things get more complicated. Random search and enumeration are not viable baselines for learning in this setting because the space of target languages is uncountably infinite. Some cleverer baseline is needed. The constraint demotion family of algorithms, including MRCD, cannot learn in a noisy setting. Likewise, given noisy data, the random baseline would never settle on a total ranking since it would keep choosing rankings randomly forever. Finally, NPRL cannot learn in the presence of noisy data for similar reasons. As shown above, it can only learn total rankings successfully with very high learning rates. Given noisy data, such high learning rates would result in jumps to unrelated grammars each time an error was produced, making convergence impossible.

In sum, there is an algorithm, MRCD, that (probably) beats the speed of the random baseline on the TS2000 test set. In terms of performance alone, the random baseline outperforms RIP/GLA and RIP/SGA on the same test set. However, of all these algorithms, only RIP/GLA and RIP/SGA are capable of dealing with noisy data and learning variable languages. Thus, none of these algorithms is able to match the performance of the random baseline on this test set and also learn languages with free variation.

5. Conclusion

Baselines are needed in order to interpret quantitative results. The random baseline shows that simulations for the Tesar and Smolensky (2000) test set that allow learners 1,000,000 iterations to settle on a grammar are not very meaningful. The average time for the random baseline to converge depends on the number of distinct languages that are generable by the system. If the random baseline is to be rejected on the basis of simulations that allow learners one million iterations, harder test sets will be needed.

Performance with regard to the classic OT learning problem is not the only consideration. It is an open question whether an algorithm that matches or at least approaches the accuracy of the random baseline and can also learn languages with free variation for systems of constraints representative of natural language exists. There are many additional criteria for evaluating computational models of phonological learning not considered here, such as the psychological plausibility of the learning procedure itself as well as the capacity to model the gradual process of phonological acquisition. Ultimately, evaluation of computational models of phonological acquisition must consider a conjunction of these and other criteria.

NPRL does not seem to be a promising alternative to existing algorithms. It does not beat the random baseline on the classic OT learning problem, and it does not extend to the learning of variable grammars. However, it does exploit a different type of learning strategy from the other algorithms, relying on matches and mismatches with the overt data, that allows it to deal with structural ambiguity. The basic strategy of rewarding components of the grammar that succeed in generating the overt data is not fully utilized by NPRL and should be explored in future work.

References

Boersma, Paul. 1997. How we learn variation, optionality, and probability. *Proceedings of the Institute of Phonetic Sciences* 21:43–58. University of Amsterdam.

Boersma, Paul. 2003. Review of Tesar and Smolensky (2000): *Learnability in Optimality Theory*. *Phonology* 20:436–446.

Boersma, Paul and Bruce Hayes. 2001. Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry* 32:45-86.

Boersma, Paul and Joe Pater. 2008. Convergence Properties of a Gradual Learning Algorithm for Harmonic Grammar. Ms., University of Amsterdam and University of Massachusetts, Amherst.

Bush, R. and Mosteller, F. 1951. A mathematical model for simple learning. *Psychological Review* 58:313-323.

Charniak, E., C. Hendrickson, and M. Perkowitz. 1993. Equations for part-of-speech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence.* AAAI Press/MIT Press, Menlo Park. 784-789.

Dempster, A., Laird, M. and Rubin, D. 1977. Maximum Likelihood estimation from incomplete data via the EM Algorithm. *Journal of Royal Statistics Society B* 39:1–38.

Eisner, Jason. 2000. Easy and hard constraint ranking in Optimality Theory: algorithms and complexity. In *Finite-state phonology: Proceedings of the 5th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON),* ed. by Jason Eisner, Lauri Karttunen, and Alain Theriault, 22-33. Luxembourg.

Jäger, Gerhard. 2007. Maximum Entropy Models and Stochastic Optimality Theory. In *Architectures, Rules, and Preferences. Variations on Themes by Joan W. Bresnan*, ed. Annie Zaenen, Jane Simpson, Tracy Holloway King, Jane Grimshaw, Joan Maling, and Chris Manning, 467-479. CSLI Publications, Stanford, California.

Jarosz, Gaja. 2009. Learning Phonology with Stochastic Partial Orders. Paper presented at the *3rd Annual Northeast Computational Phonology Meeting (NECPhon)*, MIT, Cambridge, MA, October 2009.

Legendre, Géraldine, Yoshiro Miyata, and Paul Smolensky. 1990. Can connectionism contribute to syntax? Harmonic Grammar, with an application. In *Proceedings of the 26th Regional Meeting of the Chicago Linguistic Society*, ed. M. Ziolkowski, M. Noske, and K. Deaton, 237-252. Chicago: Chicago Linguistic Society.

Pater, Joe. 2008. Gradual learning and convergence. *Linguistic Inquiry* 39:334-345.

Pearl, Lisa. 2009. Learning English Metrical Phonology: When Probability Distributions Are Not Enough. In *Proceedings of the 3rd Conference on Generative Approaches to Language Acquisition North America (GALANA 2008)*, ed. Jean Crawford, Koichi Otaki, and Masahiko Takahashi, 200-211. Somerville, MA: Cascadilla.

Soderstrom, Melanie, Donald Mathis, and Paul Smolensky. 2006. Abstract genomic encoding of Universal Grammar in Optimality Theory. In Smolensky and Legendre (2006), 403–471.

Tesar, Bruce. 1995. *Computational Optimality Theory*. Doctoral dissertation, University of Colorado, Boulder.

Tesar, Bruce. 1997a. An iterative strategy for learning metrical stress in Optimality Theory. In *Proceedings of the 21st Annual Boston University Conference on Language Development*, ed. Elizabeth Hughes, Mary Hughes, and Annabel Greenhill, 615–626. Somerville, MA: Cascadilla.

Tesar, Bruce. 1997b. Multi-Recursive Constraint Demotion. Ms., Rutgers University.

Tesar, Bruce. 2004. Using inconsistency detection to overcome structural ambiguity in language learning. *Linguistic Inquiry* 35:219-253.

Tesar, Bruce, and Paul Smolensky. 1998. Learnability in Optimality Theory. *Linguistic Inquiry* 29:229-268.

Tesar, Bruce, and Paul Smolensky. 2000. *Learnability in Optimality Theory*. Cambridge, MA: MIT Press.

Yang, Charles. 2002. *Knowledge and Learning in Natural Language*. Oxford: Oxford University Press.

Department of Linguistics
Yale University
370 Temple St., room 204
New Haven, CT 06520

gaja.jarosz@yale.edu