Convergence properties of a gradual learner in Harmonic Grammar*

Paul Boersma and Joe Pater, February 26, 2008

Abstract. This paper investigates a gradual on-line learning algorithm for Harmonic Grammar (HG). We adapt a perceptron convergence proof to a categorical version of HG, proving convergence to a grammar appropriate for the target language, given full knowledge of the structure of the learning data. We prove “almost sure convergence” when the grammar incorporates evaluation noise (as in Stochastic OT). Tests of implementations of the algorithm show that it converges quickly. Tests on representative learning problems with hidden structure show that the probability of convergence of the HG algorithm exceeds the Error-Driven Constraint Demotion Algorithm and the Gradual Learning Algorithm.

Keywords: learnability, Optimality Theory, Harmonic Grammar, Stochastic OT, Noisy HG, perceptron

1. Introduction

An especially attractive feature of Optimality Theory (OT; Prince and Smolensky 1993/2004) is that it has an associated family of provably convergent learning algorithms, referred to as a group as Constraint Demotion (CD). These algorithms were initially proposed by Tesar (1995) and are also presented in Tesar and Smolensky (1998, 2000); our discussion of CD will refer to the 2000 book presentation (TS).

The primary goal of OT learning algorithms is to find a constraint ranking that renders optimal a set of input-output mappings, that is, to find a ranking that makes each of these mappings optimal in its own candidate set. This set of input-output mappings can be referred to as the target language. CD is guaranteed to succeed in finding a ranking that meets this goal for a target language, provided certain conditions are met.

The main condition for CD convergence is that the target language must be generated by some total ranking of the constraint set (TS, pp. 47-50, 91). This condition goes beyond a more basic restriction on all learning algorithms: they can only learn a target language that their underlying grammar model can represent. CD operates with stratified constraint hierarchies, which allow multiple constraints to

* Acknowledgments forthcoming.
have the same rank, and which can thus represent target languages that cannot be generated by a total ordering of constraints. TS note in particular that stratified constraint hierarchies allow ties between multiple members of a candidate set, where each candidate has a distinct pattern of constraint violation. With a total ordering of the constraint set, only one these candidates can be made optimal. TS observe that CD will fail to converge when optima are not unique in each candidate set.

If the set of languages that the learning algorithm must deal with includes only ones with a single optimum per candidate set, that is, the set of languages that Prince and Smolensky’s (1993/2004) categorical version of OT with a total constraint order defines, then this condition on CD convergence is unproblematic. However, there are natural language phenomena that have been claimed to involve multiple optimal outputs for a single input: so-called free variation. In OT, this has been analyzed in terms of random variation between constraint rankings (Kiparsky 1993, Anttila 1997, Boersma 1997, Boersma and Hayes 2001, amongst others; see Anttila 2007, Coetzee and Pater to appear for recent overviews). In this view of variation, the constraint ranking is in a total order when it evaluates any given candidate set, and does produce a single optimal output for every input in each evaluation. However, random variation in the ordering of the constraints across instances of evaluation can produce variation in outcomes. As in the case of multiple optima in a single candidate set produced by a stratified hierarchy, CD will often fail to converge on multiple optima produced by variable rankings (Boersma 1997, Boersma and Hayes 2001).

While there are several versions of OT that can represent variation, Stochastic OT (Boersma 1997, Boersma and Hayes 2001) appears to be the only one that has an associated learning algorithm. The Gradual Learning Algorithm (GLA) usually succeeds in finding a grammar that reproduces probability distributions over outputs in the learning data, if those distributions can be represented by Stochastic OT. As well being able to learn languages with variation, the GLA differs from CD in its robustness in the face of noise in the learning data, and in exhibiting gradual learning curves similar to those observed in natural language acquisition (Boersma 1997, Boersma and Levelt 2000). Both of these attributes are desirable if the learning theory is to function as a realistic model of language acquisition. However, the GLA also diverges from CD in that it is not guaranteed to converge: there exist categorical sets of language data that can be represented by OT (and therefore by Stochastic OT) but cannot be learned by the GLA (Pater 2008).

In this paper we present an approach to grammar learning that combines the greatest strengths of the two OT learning algorithms. Like CD, the learning algorithm is guaranteed to converge on sets of learning data with unique optima. Like the GLA, the learning algorithm is compatible with a theory of grammar that represents
variation, and when combined with that grammar model, is robust in noise and produces gradual learning curves. Furthermore, when run with a stochastic model of grammar, the learning algorithm continues to find grammars that make correct categorical choices of optima.

This approach to learning differs from both the GLA and CD in that it assumes a grammar of numerically weighted constraints, as in OT’s predecessor Harmonic Grammar (HG; Legendre, Miyata and Smolensky 1990, Smolensky and Legendre 2006; see also Pater, Bhatt and Potts 2007 for comparisons of HG to OT). To deal with variable outcomes, we adopt Stochastic OT’s evaluation noise: the values of the weights are subject to random variation each time the grammar is used to evaluate a candidate set. We refer to the resulting grammar model as Noisy HG. We call the learning algorithm HG-GLA (for Harmonic Grammar’s Gradual Learning Algorithm). It was first applied to Maximum Entropy constraint grammars by Jäger (2003/to appear), and to a connectionist implementation of HG by Soderstrom, Mathis and Smolensky (2006). As these references indicate, HG-GLA adapts proposals from statistical and connectionist learning to the case of generative grammar learning (see further §2). As in OT, however, the constraints that make up HG grammars, and define the space of possible languages, can be adapted from research in the generative tradition.

We introduce the theories of grammar and learning in section 2, and then turn to our main results in sections 3 and 4: the proofs of HG-GLA convergence and tests of implementations. Section 3 proves that a categorical HG learner with HG-GLA will succeed in finding appropriate constraint weights for any finite set of language data generated by a categorical version of HG. This is an adaptation and extension of existing proofs of convergence for Rosenblatt’s (1958) perceptron learning algorithm. We also provide results from learning simulations on randomly generated languages that show that in practice HG-GLA converges very fast, and also that it converges even if constraint weights are limited to positive values, as in ‘positive HG’ (Prince 2002). Section 4 proves that just like the categorical HG learner of section 3, a Noisy HG learner with HG-GLA is also guaranteed to find a correct grammar for a set of unique optima, by extending the proof in section 3. Again, tests of implementations show that the process of finding a correct grammar turns out to be fast in practice, and that the learning result also holds with positive HG. Furthermore, individual learners display a natural-looking learning curve. As a comparison, we show that similar simulations with a Stochastic OT learner with OT-GLA display a moderate incidence of non-convergence.
Section 5 discusses how HG learners generalize from finite datasets to infinite target languages, and compares the nature of generalization in OT and HG. It also briefly discusses how some of the assumptions of our proof might be relaxed.

In sections 6 and 7, we present an initial investigation and discussion of the behaviour of HG-GLA on learning problems that present well-known challenges to CD, and to other learning algorithms. A further condition on the convergence of CD is that the learner must have full information about the structure of the learning data. As noted by TS, this condition is often not a realistic characterization of natural language acquisition. For instance, the learner may receive information about the position of stress, but not about any additional prosodic surface structure like footing. Such ‘hidden structure’ problems are extremely common; they occur every time the learner has to infer phonological structures from raw auditory data. TS propose an approach to the learning of hidden structure termed Robust Interpretive Parsing (RIP), and test this approach in conjunction with Error-Driven Constraint Demotion (EDCD). In section 6, we replicate their tests, and those of Boersma (2003) on stochastic OT with OT-GLA, and also test RIP with HG-GLA and categorical and Noisy HG, including ‘Positive’ versions. We find that the highest probability of convergence is obtained by HG-GLA with Noisy HG.

Section 7 briefly discusses why the subset problem for the acquisition of OT and HG grammars (Smolensky 1996) appears to be less severe for HG-GLA than for EDCD or OT-GLA. Although EDCD is guaranteed to find a ranking that renders all of the input-output pairs in the target language optimal, that ranking may be insufficiently restrictive, insofar as it also renders optimal input-output pairs that the analyst, or human learner, judges not to belong to the target language. Following Boersma and Levelt (2003), we show that the shared ‘symmetry’ property of OT-GLA and HG-GLA can yield more restrictive results than EDCD. Following Jesney and Tessier (2007), we show that HG weighted constraint interaction used by HG-GLA can lead to more restrictive outcomes than the ranked constraints used in EDCD and OT-GLA.

Section 8 concludes that HG has a learning algorithm that in all respects investigated either equals or outperforms the OT learning algorithms that we have compared it to, and that there is considerable promise for further development of HG learning theory.

2 Harmonic Grammar and a Gradual Learning Algorithm

We assume that our readers are familiar with OT as presented in Prince and Smolensky 1993/2004 as well as with CD (see also Kager 1999 and McCarthy 2002, 2007a, 2008 for introductions to categorical OT and CD). Because HG is likely less
familiar, we first provide an introduction to this theory of generative grammar, and to its noisy version, along with the associated learning algorithm.

In HG, the well-formedness of a linguistic representation is defined in terms of a *harmony* function (Legendre *et al.* 1990, Smolensky and Legendre 2006). The harmony of a representation is the sum of its weighted constraint scores. Prince and Smolensky (1993/2004: 236) point out that the optimal member of a candidate set can be defined in HG terms. In (2), we provide a simple example of an HG candidate evaluation. Following Legendre, Sorace and Smolensky (2006) we denote violations by negative numbers (an empty cell means no violation); constraint weights are shown beneath the constraint names, and harmony values are shown at the end of candidate rows. The candidate \((i_1, o_11)\) incurs one violation of \(C_1\), which has a weight of 1.0, so its harmony is \(1.0(-1) = -1\). Because the second candidate \((i_1, o_12)\) violates a constraint with lower weight, it has a higher harmony than the first candidate, namely \(-0.9\). The last candidate \((i_1, o_{13})\) has two violations of \(C_2\) and one of \(C_1\), which yields a harmony score of \(1.0(-1) + 0.9(-2) = -2.8\), which is also lower than \(-0.9\). Given Prince and Smolensky’s definition of optimality as maximal harmony, the second candidate is chosen as optimal, as indicated by a pointing finger.

\[
\begin{array}{|c|c|c|c|}
\hline
i_1 & C_1 & C_2 & \text{Harmony} \\
\hline
& 1.0 & 0.9 & -1.0 \\
\otimes & -1 & -1 & -0.9 \\
\otimes & -1 & -2 & -2.8 \\
\hline
\end{array}
\]

The learning algorithm we adopt is very similar to the Gradual Learning Algorithm for OT (OT-GLA; Boersma 1997 *et seq.*). The learner is given a single input-output pair at a time, and adjusts the constraint weights only when the current state of its grammar chooses a different output as optimal (for the same input). This kind of *error-driven learning* is also known from perceptrons (Rosenblatt 1958) and their connectionist relatives, from EDCD (Tesar 1995 *et seq.*), and from the parsing-based parameter setting algorithm of Sakas and Fodor (2001).

The following is a simple example of an ‘error’ made by an HG learner. Suppose that at some point during the acquisition period a learner has the weights shown in (2), and that the learner is then presented with the learning datum \((i_1, o_{11})\). The learner will consider \(o_{11}\) as the ‘correct’ output for the input \(i_1\), as indicated with the check mark in tableau (3). Meanwhile, the learners’ own optimal output (the one with highest
harmony given her current grammar) is $o_{12}$, and this is indicated with the pointing finger in (3); the learner will now consider this form ‘incorrect’.

(3) An ‘error’ in HG

<table>
<thead>
<tr>
<th>$i_1$</th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{\circ} o_{11}$</td>
<td>-1</td>
<td>-1.0</td>
</tr>
<tr>
<td>$\circ o_{12}$</td>
<td>-1</td>
<td>-0.9</td>
</tr>
<tr>
<td>$o_{13}$</td>
<td>-1</td>
<td>-2.8</td>
</tr>
</tbody>
</table>

Upon making such an error, the learner updates the constraint weights so as to render the error less likely in encountering subsequent data with similar properties. The update procedure we use is one that is widely used in connectionist and statistical learning, and has been applied to grammatical constraint weighting under several names. Under the name of *stochastic gradient ascent*, Jaeger (2003/to appear) applied it to HG’s stochastic cousin called *Maximum Entropy grammar*; under the name of *gradient descent in error*, Soderstrom, Mathis and Smolensky (2006: eqs. 14, 18, 21, 35d) applied it to a connectionist implementation of categorical HG; and under the name of *perceptron update rule* Pater (2008) applied it to the present version of categorical HG. Here we call it by its function, namely *HG-GLA*: Noisy Harmonic Grammar’s gradual learning algorithm.

The first step of the learning algorithm is to calculate the difference between the violation scores of the incorrect and correct forms. This yields an error vector, as shown in (4). The error vector in (4) results from subtracting the scores of the incorrectly optimal ($i_1, o_{12}$) from the scores of the correct ($i_1, o_{11}$); for instance, for $C_2$ we compute 0 minus $-1$, which yields $+1$. The error vector is the HG equivalent of the mark-data pair or winner-loser pair used in OT learning (Tesar 1995 et seq.), and was first explicitly discussed for Maximum Entropy grammars by Goldwater and Johnson (2003) and for HG by Becker and Pater (2007).

(4) The error vector: scores of correct pair minus scores of incorrect pair

<table>
<thead>
<tr>
<th>Correct ~ incorrect</th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>($i_1, o_{11}$) ~ ($i_1, o_{12}$)</td>
<td>-1</td>
<td>+1</td>
</tr>
</tbody>
</table>

Next, the constraint weights are updated by adding to them the values of this error vector, each multiplied by a small constant $\epsilon$, termed *plasticity* in the GLA literature (or *learning rate* in the machine learning literature). The weight of a constraint that is violated more in the correct form than in the incorrect form will thus be lowered, and
the weight of a constraint that is violated more in the incorrect form will be raised. If we set the plasticity at \( \varepsilon = 0.4 \), and add the scaled difference values to the constraint weights, we obtain tableau (5).

(5) Updated grammar

<table>
<thead>
<tr>
<th>( i_1 )</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sqrt{\varepsilon} \cdot 0.6 )</td>
<td>-1</td>
<td>-0.6</td>
</tr>
<tr>
<td>( \sigma_{12} )</td>
<td>-1</td>
<td>-1.3</td>
</tr>
<tr>
<td>( \sigma_{13} )</td>
<td>-1</td>
<td>-2</td>
</tr>
</tbody>
</table>

In tableau (5), \( 0.4 \cdot (-1) = -0.4 \) has been added to the weight of \( C_1 \), and \( 0.4 \cdot (+1) = 0.4 \) has been added to the weight of \( C_2 \). As a result, the grammar has changed so much that the behavior of the learner when presented with the next instance of \( i_1 \) will have changed: as we see in (5), \( \sigma_{11} \) has become the optimal form in the learner’s grammar, and since this form is also the correct form, learning has stopped as far as input \( i_1 \) is concerned. With smaller plasticities than 0.4, the acquisition period is likely to take longer.

The most accurate name for this type of learning algorithm (starting with the perceptron) is stochastic gradient descent in error: ‘stochastic’ because it is an on-line learning algorithm that relies on a randomized sequence of learning data, and ‘gradient descent in error’ because the ‘error’, which is the harmony difference between learner’s incorrect optimum and the correct form, is reduced; ‘in error’ also implies the error-drivenness of the algorithm, in contrast with algorithms that update their parameters on every input.

As section 3 shows, this learning algorithm is very effective: given any finite set of unique optima that can be described by an HG grammar, it is guaranteed to converge on a correct weighting. By “unique optima”, we mean that in the data the learner is exposed to, every input has a single optimal output, as assumed in the CD convergence proof. Thus, like the proof of convergence for EDCD, the proof in section 3 is only valid for the special case of learning a language with categorical choices of outputs.

As mentioned in the introduction, it has been claimed that variation between optima for a single input corresponds to the natural language phenomenon of “free” variation. In phonology, this is the variation between two phonological surface forms of a single underlying form, across utterances that are identical in all relevant respects. To deal with such cases, we modify HG by incorporating evaluation noise, as in
Stochastic OT (Boersma 1997 et seq.), so that we obtain Noisy HG.\footnote{Another stochastic variant of HG is Maximum Entropy grammar (Johnson 2002, Goldwater and Johnson 2003, Jäger 2003/to appear, Fischer 2005, Wilson 2006). In this model, every output candidate has a relative probability that rises exponentially with its harmony. One difference from Noisy HG is that Maximum Entropy grammars allow harmonically bounded candidates to surface (Paul Smolensky, p.c., Jesney 2007). These always have zero probability in a version of Noisy HG that limits weights to positive values (see §3.5).} In Noisy HG, every time the grammar is used to evaluate a candidate set, the weighting values are perturbed by noise: each constraint’s value is temporarily altered by adding a random positive or negative number sampled from a normal distribution. This will lead to different weightings across instances of evaluation, which will produce variation. The proportion of times a candidate in any given candidate set is chosen is determined by the weights of the constraints, and the size of the noise distribution. The effect of noisy evaluation is illustrated in (7), which shows two evaluations for the input $i_1$ in the same noisy grammar. The basic weights (corresponding to the ranking values of Stochastic OT) of the constraints are 1.1 and 1.0 for $C_1$ and $C_2$ respectively. The weights after the addition of noise (corresponding to the disharmonies of Stochastic OT) are shown in parentheses. Depending on these stochastic weights, either output $o_{12}$ or output $o_{11}$ will be chosen, as these two tableaux show. Because $o_{12}$ violates the constraint with a lower basic weight, it will be optimal in a greater proportion of evaluations.

(7) Variation in choices of optima in noisy HG: one grammar, two outputs

<table>
<thead>
<tr>
<th>$i_1$</th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.1 (1.1)</td>
<td>1.0 (0.9)</td>
</tr>
<tr>
<td>$o_{11}$</td>
<td>$-1$</td>
<td>$-1.1$</td>
</tr>
<tr>
<td>$o_{12}$</td>
<td>$-1$</td>
<td>$-0.9$</td>
</tr>
</tbody>
</table>

As we show in section 4, adding such noise does not affect the ability of the learner to find correct weightings for categorical choices of optima, and it does lead to realistic learning curves.

Before we turn to our proof, we provide one slightly more complex illustrative example to more fully demonstrate HG evaluation and learning. The OT literature usually speaks of constraint violations rather than of constraint satisfactions, but the OT formalism does not really disallow the use of positive constraint satisfaction, so we explicitly allow it here (following Legendre, Sorace and Smolensky 2006). In tableau (8) output candidate $o_{31}$ violates constraint $C_1$ twice, but it also positively satisfies $C_3$ once. There are various ways in which constraints might be formulated so that they assign negative scores on violation and/or positive scores on satisfaction; the details are irrelevant to our convergence proofs and tests. We do note, though, that
positive satisfactions seem to be more compatible with HG than with OT (see e.g. Boersma and Pater 2007, Boersma and Escudero to appear).

(8) A more complex illustrative HG evaluation

<table>
<thead>
<tr>
<th>$i_3$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
<th>$C_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_{31}$</td>
<td>-2</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>-47.0</td>
</tr>
<tr>
<td>$o_{32}$</td>
<td>-1</td>
<td>-2</td>
<td></td>
<td></td>
<td>-54.0</td>
</tr>
<tr>
<td>$o_{33}$</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td></td>
<td>-34.0</td>
</tr>
<tr>
<td>$o_{34}$</td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
<td>-53.0</td>
</tr>
</tbody>
</table>

Since output candidate $o_{31}$ violates $C_1$ twice, and the constraint has a weight of 39.0, an amount of $-78.0$ is contributed to the harmony of the candidate. The candidate does not violate or positively satisfy constraint $C_2$, hence there is no contribution to the harmony from this constraint. Since candidate $o_{31}$ satisfies both $C_3$ and $C_4$ positively, these satisfactions together contribute an amount of $14.0 \cdot (1) + 13.0 \cdot (1) = +27.0$ to the harmony of $o_{31}$. Candidate $o_{31}$ violates $C_5$ once, but because the constraint has a negative weight, this ‘violation’ contributes a positive amount of $-4.0 \cdot (-1.0) = +4.0$ to the total harmony of $o_{31}$, which is therefore $-78.0 + 27.0 + 4.0 = -47.0$. The harmonies of the remaining three candidates are computed in analogous ways. Candidate $o_{33}$ ends up with the highest harmony, so it is the winner in the tableau and the actual output of the HG grammar for input $i_3$.

This tableau provides an example of the fact that in HG, unlike OT, violations of a constraint with lower weight can ‘gang up’ to overcome a violation of a constraint with higher weight: for candidate $o_{32}$, the double violation of $C_4$ weighs heavier than the single violation of $C_3$ plus the double ‘violation’ of $C_5$ (see Legendre et al. 2006, Pater et al. 2007a, and Tesar 2007 for discussion of the typological consequences of such gang effects).

To illustrate how the learning algorithm works for more complicated instances of constraint violation and satisfaction, let us assume that (8) represents the optimum that a learner chooses upon being presented with the correct mapping ($i_3$, $o_{31}$), that is the first candidate in the tableau. With this candidate taken as the correct form, and the optimum in (8) as the incorrect one, the resulting error vector is as in (9).

(9) Error vector from the constraint scores in (8)

<table>
<thead>
<tr>
<th>Correct ~ incorrect</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
<th>$C_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(i_3$-$o_{31})$ ~ $(i_3$-$o_{33})$</td>
<td>-2</td>
<td>+1</td>
<td>+2</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>
These are the values that will be multiplied by the plasticity $\varepsilon$ and then added to the constraint weights. This vector illustrates the important point that the degree of change of constraints in this learning algorithm is sensitive to the degree of difference between the correct form and the error. Because the correct form satisfies $C_3$ twice, the weight of $C_3$ will be raised by $2\varepsilon$, whereas the weight of $C_2$ will only be raised by $\varepsilon$.

In the version of HG proposed in Legendre et al. (1990) and Smolensky and Legendre (2006), constraints can be both satisfied and violated, and can have both negative and positive weights. It is possible to limit constraints to assigning only negative numbers (violations) or positive numbers (satisfactions), and weightings can be limited to non-negative values (Keller 2000, 2006) or to positive ones (Prince 2002). In our proof, we assume the original, most general, formulation of the theory. In §3.5 we discuss the motivation for banning negative weights, and test learners that impose a positivity restriction on weights.

3. Convergence of HG-GLA in the noise-free case

In this section, we first provide a formal definition of the learning algorithm described in §2 (HG-GLA), restricted to a learner without evaluation noise. We then provide a convergence proof for this noise-free case, and finally show that a computer-implemented version of the algorithm converges very fast in practice.

3.1 Formalization of the language data and the learner

Our proof operates on the basis of a finite set of language data that we call the dataset. The learning data that are supplied to the learner are sampled from that finite set. Our proof guarantees convergence for any learning simulation that is conducted in this fashion, as are all published OT learning simulations that we are aware of. In section 5, we discuss how an HG learner generalizes from a finite set of learning data to the infinite set of data characterized by a language, and briefly discuss how the finiteness assumption on the dataset might be weakened, so that the proof could generalize to the case of directly sampling from the infinite set itself.

The dataset, then, consists of a finite number $M$ of ‘correct’ input-output pairs. Thus, the dataset will have $M$ different input (e.g. underlying) forms $i_m$ ($m = 1...M$), each of which is coupled with a unique ‘correct’ output form $o_m$.

We assume that the language user, especially the learner, maintains a finite number $K$ of constraints $C_k$ ($k = 1..K$) with weights $w_k$; these weights are used for input-output processing, as in (2) and (8), and can change as a result of learning, as in (5) and (9). We have five assumptions about the learner’s processing, and three about learning.
The first assumption about processing is that for each input \( i_m \) of the dataset the learner can compute (or is given) a finite number \( N_m \) of output candidates \( o_{mn} \) \((n = 1..N_m)\), as in the tableaux of §2 (see §5.2 on the standard OT infinite candidate set). The second assumption is that the learner can compute (or is given) the extent to which each input-output pair \( (i_m, o_{mn}) \) satisfies each constraint \( C_k \) \((k = 1..K)\). We call the extent of constraint satisfaction \( s_{mnk} \); for instance, the value of \( s_{324} \) in tableau (8) is \(-2\). The third assumption is that the learner can compute a harmony value for each input-output pair \((i_m, o_{mn})\), as a weighted sum of satisfaction scores:

\[
(10) \quad H_{mn} = \sum_{k=1}^{K} w_k s_{mnk}
\]

In (8), for instance, the value of \( H_{53} \) is \(-54.0\). The fourth assumption is that the learner, when asked to evaluate an input form \( i_m \), can identify one or more ‘optimal’ forms \( o_{mrs} \), i.e. forms that have a maximum harmony among the output candidates:

\[
(11) \quad \forall m = 1..M, n = 1..N_m : \sum_{k=1}^{K} w_k s_{mrk} \geq \sum_{k=1}^{K} w_k s_{mnk}
\]

The fifth assumption about processing is that if there are multiple optimal forms, the learner is able to randomly select one of them as the ‘winning candidate’ of her evaluation.

The objective of the learning procedure is to find constraint strengths \( w_k \) such that the optimal outputs for all inputs \( i_m \) in the learner’s grammar are identical to the correct outputs \( o_m \) of the dataset. The first assumption about learning is that for every input-output pair \((i_p, o_p)\) that arrives in the language data, the learner computes her own winning output candidate for the given input \( i_p \). Thus, this winning candidate is \( o_{pr} \), for some \( r \) in the range \( 1..N_p \); it has constraint satisfactions \( s_{prk} \) and harmony \( H_{pr} \). The second assumption is that the output form \( o_p \) given in the language data is among the learner’s set of output candidates for \( i_p \), i.e., the learner can identify the given output form as \( o_{pq} \), where \( 1 \leq q \leq N_p \), and thereby establish the satisfactions \( s_{pqk} \) and harmony \( H_{pq} \) of this form in her current grammar. The third assumption about learning is that the learner compares this form \( o_{pq} \) with her own winning candidate \( o_{pr} \). If it turns out that the two forms are different (and therefore indicate that the learning objective has not been reached yet), the learner labels the situation as an error: the form \( o_{pq} \) is now considered the ‘correct output’ and \( o_{pr} \) the ‘incorrect optimum’. The learner then takes action by changing the weights of the constraints from \( w_k \) to \( w'_{k} \):
\[ w_k' = w_k + \epsilon \cdot \left( s_{pk} - s_{rk} \right) \]

where \( \epsilon \) is the plasticity, a positive real number. This update rule is the same as we used in (5), where we took \( \epsilon = 0.4 \); the formula is identical to that used by Jäger (2003/to appear) for Maximum Entropy grammars and a notational variant of the formulas used by Soderstrom, Mathis and Smolensky (2006: eqs. 14, 18, 21, 35d) for their connectionist implementation of HG. The update rule implies that the learner raises the weights of all the constraints satisfied better with the correct output \( o_{pq} \) than with the incorrect optimum \( o_{pr} \), and lowers the weights satisfied with the incorrect optimum than with the correct output.

3.2 The unit target grammar

In this section we provide the formal background related to the grammar model assumed for the proof in the following section. We also introduce some of the basic concepts of the perceptron convergence proof, including its most fundamental one: the margin of separation. In HG terms, the margin of separation defines the smallest harmony distance (within the dataset) between any optimal form and its closest competitor.

As mentioned in the introduction, no learning algorithm can learn a language that its underlying grammar model cannot represent. Thus, the learning algorithm defined in §3.1 will fail to learn any language that cannot be described by a Harmonic Grammar. What we really want to prove, therefore, is that HG-GLA is successful for any dataset that can be generated by a Harmonic Grammar.

By safe assumption, therefore, there exists at least one weighting of the constraints that makes the set of correct forms in the dataset optimal. Given the nature of the weights (namely, real numbers) and the nature of the interaction of the finite set of \( K \) constraints, there probably exist many such weightings, but here we select just one, and call it the sample target grammar. The goal of the learner, then, is to arrive either at this sample target grammar or at any other grammar that describes the dataset correctly.

We do not know much about the constraints’ weights in the sample target grammar, but we do know that not all weights are zero, because if they were, all candidates would be optimal, and the sample target grammar would exhibit variation in its outputs (unless all inputs have only a single output candidate, but in that case the learner already has a correct grammar from the start). Since not all weights are zero, we can normalize the sample target grammar by dividing every constraint weight (strength) by the square root of the sum of the squares of the constraint weights. Thus, if the sample target grammar has six constraints with weights 17, 5, -3, 14, -5, and 9, the sum of their squares is $17^2 + 5^2 + (-3)^2 + 14^2 + (-5)^2 + 9^2 = 625$, whose square root is
25. When we divide all constraint weights by 25, we get the weights (0.68, 0.20, -0.12, 0.56, -0.20, 0.36). This grammar generates exactly the same language as it did before the division. We call such a form of the grammar a unit grammar: the sum of the squares of its weights is 1. For the unit sample target grammar, we write these weights as \( u_k \) \((k=1..K)\).

In the sample grammar, the optimal outputs are unique (they are the correct language forms). Therefore, the harmony (in the sample grammar) of any optimal form is greater than that of any other candidates of the same input. Suppose that for each input \( i_m \) the optimal output is \( o_{m,n} \) \((n_m \text{ is the index of this output in the learner’s tableau for } i_m)\). Then

\[
\forall m = 1..M, n = 1..N_m, n \neq n_m : \sum_{k=1}^{K} u_k s_{m,n,k} > \sum_{k=1}^{K} u_k s_{m,n} + \delta
\]

We now assume that for every input a second-best candidate can also be found (it does not have to be unique); it is the winner of the partial tableau that results if the best form is left out. The harmony differences between the second-best forms and the best forms are \( \delta_m \) \((m=1..M)\); for input \( i_m \), the harmony (in the unit target grammar) of all non-optimal candidates is at least \( \delta_m \) less than the harmony of the optimal candidate. Since the number \( M \) of inputs is finite, there must exist a value \( \delta \) that is the minimum of all \( M \) values \( \delta_m \). We therefore know that the harmony of any non-optimal candidate for any input is separated by at least \( \delta \) from the harmony of the optimal candidate for that same input (in the unit target grammar):

\[
\forall m = 1..M, n = 1..N_m, n \neq n_m : \sum_{k=1}^{K} u_k s_{m,n,k} \geq \sum_{k=1}^{K} u_k s_{m,n} + \delta
\]

We call \( \delta \) the margin of separation of harmony; this value is independent of the language data fed to the learner, and it is greater than 0 for any Harmonic Grammar that generates a nonvariable language with a finite number of possible inputs.

### 3.3 A perceptron proof for HG-GLA

The perceptron is a model of linear classification developed by Rosenblatt (1958). Its learning rule is convergent: if two sets of data are linearly separable, the perceptron update rule is guaranteed to find a set of weights that will correctly separate them (Block 1962, Novikoff 1962). Our convergence proof for HG-GLA closely follows the perceptron convergence proof by Novikoff (1962) and its application to a case of part-of-speech tagging by Collins (2002), and extends it to the case of HG learning
from any initial set of constraint weights, using the grammar and learning model we have just described.²

We prove convergence by showing that given any finite set of language data that can be generated by an HG grammar, the learner is guaranteed to make only a finite number of errors; after that, the learner will make no further errors and has reached a correct grammar. Therefore, the main goal of the proof is to show that for any given set of language data, we can define an upper bound on the number of errors that the learner will make.

The initial state of the learner (i.e. the state after zero errors) is characterized by a set of initial constraint weights \( w_k(0) \). These weights could be all zero, or all 10.0, or weights corresponding to any of the initial rankings proposed in the OT literature (e.g. all markedness constraints high, all faithfulness constraints low, as proposed by Demuth 1995, Levelt 1995, Ohala 1996, Smolensky 1996; see further §7). Since our proof works for any initial state, we do not impose any restrictions on the initial weights.

As the language data come in, the learner receives a sequence of ‘correct’ input-output pairs. However, since only the pairs that cause errors will make changes to the weights, we here consider only the pairs that cause an error. It therefore makes sense to define \( w_k(t) \) as the constraint weight of \( C_k \) after \( t \) errors; the set of \( w_k(t) \) for all \( k \) and all \( t \) therefore defines the entire history of the learner’s grammar. Let the \( t \)th error-causing input-output pair be \((i_{p(t)}, o_{p(t)})\), the incorrect winning candidate \( o_{p(t)r(t)} \), and the correct output form \( o_{p(t)v(t)} = o_{p(t)} \). The update rule for this data pair is then

\[
(15) \quad w_k(t) = w_k(t-1) + \epsilon \cdot \left( s_{p(t)v(t)k} - s_{p(t)r(t)k} \right)
\]

We can now start off on some mathematical ‘tricks’ that together allow us to establish an upper bound on the number of errors. First, we take the inner product of all terms in (15) with the weightings of the unit sample target grammar:

\[
(16) \quad \sum_{k=1}^{K} u_k w_k(t) = \sum_{k=1}^{K} u_k w_k(t-1) + \epsilon \cdot \left( \sum_{k=1}^{K} u_k s_{p(t)v(t)k} - \sum_{k=1}^{K} u_k s_{p(t)r(t)k} \right)
\]

With the help of (14), we can rewrite the last term, so that (16) becomes

² In proving convergence for the update rule (12) in a constraint-weighting model, we are preceded by Fischer (2005), who provides a proof of its convergence for Maximum Entropy grammars. Unfortunately, this proof does not immediately generalize to Noisy HG learners, whereas the perceptron proof does (as we show in §4). Within HG, Soderstrom, Mathis and Smolensky (2006) did use the update rule (12), but provided no convergence proof.
\begin{equation}
\sum_{k=1}^{K} u_k w_k(t) \geq \sum_{k=1}^{K} u_k w_k(t-1) + \varepsilon \cdot \delta 
\end{equation}

By induction, starting with \( t = 0 \), we derive

\begin{equation}
\sum_{k=1}^{K} u_k w_k(t) \geq \sum_{k=1}^{K} u_k w_k(0) + \varepsilon t \delta 
\end{equation}

The first term on the right is the inner product of the weight vector \( u_k \) of the unit sample target grammar and the initial weight vector \( w_k(0) \) of the learner’s grammar. We abbreviate it as \( W_0 \cdot U \), so that (18) can be written as

\begin{equation}
\sum_{k=1}^{K} u_k w_k(t) \geq \varepsilon t \delta + W_0 \cdot U 
\end{equation}

According to the Cauchy-Schwarz inequality, the absolute value of the inner product of two vectors is always less than or equal to the product of the norms of the two vectors:

\begin{equation}
\sqrt{\sum_{k=1}^{K} u_k^2} \sqrt{\sum_{k=1}^{K} w_k^2(t)} \geq \left| \sum_{k=1}^{K} u_k w_k(t) \right|
\end{equation}

Since the first factor on the left in (20) is 1 (since \( u_k \) defines a unit grammar, its norm is 1), we can combine (19) and (20) into

\begin{equation}
\sqrt{\sum_{k=1}^{K} w_k^2(t)} \geq \varepsilon t \delta + W_0 \cdot U 
\end{equation}

Loosely speaking, the overall constraint weights move away from zero with time, i.e. with the number of mistakes \( t \). The term on the left in (21) is the norm of \( w_k \). Equation (21) expresses the fact that the norm is bounded from below by a line that rises with \( t \). Equivalently, we can say that the square of the norm is bounded from below by a concave parabola:

\begin{equation}
\sum_{k=1}^{K} w_k^2(t) \geq \left( \varepsilon t \delta + W_0 \cdot U \right)^2 
\end{equation}

Beside a lower bound we can also determine an upper bound on this squared norm. We start by writing it in terms of what it was before the latest learning step:
\[
\sum_{k=1}^{K} w_k^2(t) = \sum_{k=1}^{K} \left( w_k(t-1) + \varepsilon \cdot \left( s_{p(t)q(t)k} - s_{p(t)r(t)k} \right) \right)^2 = \\
= \sum_{k=1}^{K} w_k^2(t-1) + 2\varepsilon \sum_{k=1}^{K} w_k(t-1) \left( s_{p(t)q(t)k} - s_{p(t)r(t)k} \right) + \varepsilon^2 \sum_{k=1}^{K} \left( s_{p(t)q(t)k} - s_{p(t)r(t)k} \right)^2
\]

According to (10), the factor after \(2\varepsilon\) involves a harmony difference:

\[
\sum_{k=1}^{K} w_k(t-1) \left( s_{p(t)q(t)k} - s_{p(t)r(t)k} \right) = H_{p(t)q(t)}(t-1) - H_{p(t)r(t)}(t-1)
\]

This is the difference between the harmony of the correct form in the learning data (a non-optimal, or perhaps tied, candidate in the learner’s current grammar) and the ‘incorrect’ optimum (an optimal candidate in the learner’s current grammar before the learning step), and must therefore be less than or equal to zero. We can now bound the squared norm in (23) from above by throwing away the \(2\varepsilon\) term from (23):

\[
\sum_{k=1}^{K} w_k^2(t) \leq \sum_{k=1}^{K} w_k^2(t-1) + \varepsilon^2 \sum_{k=1}^{K} \left( s_{p(t)q(t)k} - s_{p(t)r(t)k} \right)^2
\]

We can proceed by noting that the sum in the last term is again bounded from above: it can never be more than the sum of squares of maximum constraint satisfaction differences between the optimal candidates and their nonoptimal counterparts. If the maximum number of constraint violations in the unit grammar for any (potentially incorrectly optimal) input-output pair is \(V_{\text{max}}\), and the maximum number of positive constraint satisfactions for any input-output pair is \(S_{\text{max}}\), the sum in the last term can never be more than \(K \times \) the square of the sum of \(V_{\text{max}}\) and \(S_{\text{max}}\). For instance, if no cell in any of the \(M\) tableaux contains more than 5 violations or 3 positive satisfactions, the last term in (24) can never be more than \(8^2 \varepsilon^2 K\). Generally, (25) becomes

\[
\sum_{k=1}^{K} w_k^2(t) \leq \sum_{k=1}^{K} w_k^2(t-1) + \varepsilon^2 K D_{\text{max}}^2
\]

where \(D_{\text{max}}\) is defined as \(V_{\text{max}} + S_{\text{max}}\). This value is independent of the language data fed to the learner, and is guaranteed to exist, since the number of output candidates for each input is finite (we discuss this condition in §5.2).

By induction on \(t\), starting again at \(t = 0\), we get
\[
(27) \quad \sum_{k=1}^{K} w_k^2(t) \leq \varepsilon^2 K D_{\text{max}}^2 t + \sum_{k=1}^{K} w_k^2(0)
\]

The last term on the right in (27) is the squared norm of the learner’s initial weight vector. We abbreviate it as \( |W_0|^2 \), so that (27) becomes

\[
(28) \quad \sum_{k=1}^{K} w_k^2(t) \leq \varepsilon^2 K D_{\text{max}}^2 t + |W_0|^2
\]

This formula establishes that the squared norm of the learner’s weight vector lies below a line that rises with the number of errors \( t \). But in (22) we have seen that the squared norm also lies above a function that increases quadratically with \( t \). Since a rising parabola cannot stay below a rising line forever with increasing \( t \), the two bounding conditions together necessarily indicate that \( t \) itself must be limited. More formally, we can combine (22) and (28) to yield

\[
(29) \quad (\varepsilon \delta t + W_0 \cdot U)^2 \leq \varepsilon^2 K D_{\text{max}}^2 t + |W_0|^2
\]

which is a quadratic equation in \( t \) that can be solved:

\[
(30) \quad \varepsilon^2 \delta^2 t^2 - (\varepsilon^2 K D_{\text{max}}^2 - 2\varepsilon \delta W_0 \cdot U) t + (W_0 \cdot U)^2 \leq |W_0|^2
\]

\[
(31) \quad \left( t - \left( \frac{K D_{\text{max}}^2}{2 \delta^2} - \frac{W_0 \cdot U}{\varepsilon \delta} \right) \right)^2 \leq \left( \frac{K D_{\text{max}}^2}{2 \delta^2} - \frac{W_0 \cdot U}{\varepsilon \delta} \right)^2 + \frac{|W_0|^2 - (W_0 \cdot U)^2}{\varepsilon^2 \delta^2}
\]

The first thing to establish is that the right-hand side of this equation cannot be negative: if we write \( t = 0 \) in (20), and take into account that the norm of \( u_k \) is one, we see that \( |W_0 \cdot U| \) must be less than or equal to \( |W_0| \), so that the second term on the right in (30) must be positive or zero. We are therefore allowed to rewrite (31) as

\[
(32) \quad \left| t - \left( \frac{K D_{\text{max}}^2}{2 \delta^2} - \frac{W_0 \cdot U}{\varepsilon \delta} \right) \right| \leq \sqrt{\left( \frac{K D_{\text{max}}^2}{2 \delta^2} - \frac{W_0 \cdot U}{\varepsilon \delta} \right)^2 + \frac{|W_0|^2 (W_0 \cdot U)^2}{\varepsilon^2 \delta^2}}
\]

For large values of \( t \), namely \( t \geq K D_{\text{max}}^2 / (2 \delta^2) - W_0 \cdot U / (\varepsilon \delta) \), (32) is equivalent to (33):

\[
(33) \quad t \leq \frac{K D_{\text{max}}^2}{2 \delta^2} - \frac{W_0 \cdot U}{\varepsilon \delta} + \sqrt{\left( \frac{K D_{\text{max}}^2}{2 \delta^2} - \frac{W_0 \cdot U}{\varepsilon \delta} \right)^2 + \frac{|W_0|^2 (W_0 \cdot U)^2}{\varepsilon^2 \delta^2}}
\]

For small values of \( t \), i.e. \( t < K D_{\text{max}}^2 / (2 \delta^2) - W_0 \cdot U / (\varepsilon \delta) \), (32) is not equivalent to (33), but (33) holds a fortiori. Therefore, (33) expresses a true upper bound on \( t \).
The interpretation of (33) is that \( t \) can never be greater than a value \( t_{\text{max}} \) that is defined as the right-hand side of (33). This \( t_{\text{max}} \) is determined completely by properties of the hypothetical unit target grammar \( (D_{\text{max}}, K, U, \delta) \), properties of the initial state \((w_k(0))\), and fixed a priori properties of the learning algorithm \( (\epsilon) \); crucially, \( t_{\text{max}} \) does not depend on any properties of the language data as they are stochastically presented to the learner. This means that given an initial state and a target grammar, we know that the learning algorithm can never make more than a fixed number of \( t_{\text{max}} \) mistakes. The interpretation is that learning will stop after at most \( t_{\text{max}} \) mistakes. In other words, the learner will end up in a never-changing grammar after at most \( t_{\text{max}} \) mistakes.

What will this final stable grammar look like? Well, it is guaranteed to be a correct grammar for the target data set. For if it were not, the grammar would have to be one that is able to produce an incorrect output for at least one input. That input is certain to arrive at the learner at some point in time (under a common assumption addressed in the next paragraph); when it does, it will cause a mistake and must force learning, which is impossible because learning has stopped. Hence, the final grammar is a correct grammar of the target data set.

The next question is whether the final grammar will ever be reached. After all, \( t \) is not the time but the number of mistakes. Between any two consecutive mistakes there may be any number of input-output pairs that cause no mistake, i.e. input-output pairs for which the learner’s current grammar produces the same output as that given by the language data. In fact, the likelihood that an input-output pair generates a mistake probably decreases as the learner’s grammar approaches one that is appropriate for the target dataset; as a result, the average expected time between two consecutive mistakes tends to rise as acquisition proceeds. A necessary condition for convergence, therefore, is the same as the one mentioned by Tesar and Smolensky (1998: 246) for the CD convergence proof, namely that informative input-output pairs “are not maliciously withheld from the learner”. For the finite data set under discussion, the continual availability of all forms to the learner is guaranteed if input-output pairs are presented in an infinite sufficiently random sequence. Similarly, any incorrect form that in the learner’s grammar ties for harmony with a correct form, should not be maliciously withheld from being chosen as a winning candidate; the random selection between ties mentioned in §3.1 guarantees this.

Since the number of possible inputs \( M \), the number of output candidates per input \( N_m \), and the number of constraints \( K \) are finite, \( D_{\text{max}} \) in (33), which is the maximum number of constraint violations-minus-satisfactions in at least one correct grammar, will be finite as well. Therefore, \( t_{\text{max}} \) will have a finite value if a margin of separation \( \delta \) exists in at least one grammar of the target language; the hypothetical sample target grammar of §3.2 is just such a grammar, again as a result of the finiteness of all the
dimensions involved. This rounds up our proof for the finite case; the cases of an infinite number of possible inputs and an infinite number of output candidates per input are discussed in §5.2.

3.4 Tests of an implementation: noiseless case

It is one thing to show that a learning algorithm converges within a finite time, but quite another thing to show that it converges within some time that would be a realistic characterization of human learner’s youth. Depending on the characteristics of a learning problem, the upper bound on the number of errors expressed by \( t \) in (33) can be quite large. This issue could be addressed formally by applying the tools of complexity theory to the case of HG learning (see Riggle to appear for a complexity analysis of OT). We leave the formal study of the complexity of HG grammars for further research. Here we address this issue by showing that a computer-implemented version of the learning algorithm converges in a reasonable amount of time.

To test whether a non-noisy HG learner is convergent not just in theory but also in practice, we generate a random target grammar and a random dataset, as follows. We create the grammar as a set of \( K = 20 \) constraints with weights randomly drawn from a uniform distribution between 2.0 and 18.0. We then define \( M = 20 \) inputs \( i_m \), each with \( N_m = 20 \) output candidates, with each candidate violating each constraint 0, 1, 2, 3, 4 or 5 times (i.e. \( D_{\text{max}} = 5 \), as determined by an evenly distributed random choice from among these six possibilities.\(^3\) That is, we have 20 tableaux with 20 candidates, each with random integer violation scores from 0 to -5 on each of 20 constraints. We then use the constraint weights to select the optimal output for each tableau. If an output happens not be unique (which can happen if the two best candidates for that input have the same violation pattern), we discard the whole dataset and start again with new inputs, weights, and violation patterns. If necessary, this is repeated until all outputs are unique, so that we know that at least one target grammar exists (it is the grammar we just created). In this way, we create a dataset consisting of 20 ‘correct’ input-output pairs.

Once the target grammar exists, we create a categorical HG learner with the same inputs, output candidates, and violation patterns as the target grammar, but with all constraints weighted at \( w_k(0) = 10.0 \) (\( k = 1..K \)). We feed this learner ‘correct’ input-output pairs of the target language, randomly and evenly selected from among the 20 possible ‘correct’ input-output pairs in the dataset. The learner’s grammar evaluates each input with the HG evaluation mechanism, and if the optimal output thus determined is different from the ‘correct’ output given by the target language, the

---

\(^3\) Thus, one violation of a constraint with weight 17.0 will outweigh five violations of a constraint with weight 3.0.
learner takes an HG learning step with a plasticity of $\varepsilon = 1.0$.\footnote{In a computer implementation of the non-noisy learner, the plasticity has to have an integer value, so as to avoid floating-point rounding errors. This is necessary to allow the learner to go through stages with crucially tied candidates, i.e. stages where multiple candidates within a tableau have exactly equal harmonies. Such situations lead to random variation, and hence to more learning (§3.1).} After every 10 pieces of learning data, we check whether the learner has arrived in a grammar in which all outputs are uniquely optimal. If the learner has reached such a grammar at e.g. the 18th check, we conclude that the learner has converged between the 171st and 180th learning datum.

We performed this procedure not just for a single virtual learner, but for 100,000 virtual learners. They all converge, and do not take much time. Figure 1 shows a histogram of the convergence times.

The slowest learner required 3100 pieces of data, but most learners required only between 30 and 400 pieces data to converge.

HG learners should be able to learn any finite dataset generated by OT grammars, since given any set of constraints, any dataset generated by OT can also be generated by HG (cf. fn. 6 below). We therefore tested another 100,000 categorical HG learners on datasets generated by target OT grammars. We created such a target grammar as a random total ranking of constraints, and computed the 20 optimal outputs in the usual OT way. Again, all 100,000 HG learners converged, although it took them on average slightly more time than they needed for the target HG grammars.

We conclude that in practice categorical HG-GLA learners converge quite fast.

### 3.5 Tests of a learner with a positivity restriction

If HG is to function as an OT-like theory of language typology, then it appears necessary to ban negative weights (Keller 2000, 2006, Prince 2002, Pater et al.)

![Fig. 1. Histogram of convergence times of 100,000 categorical HG learners.](image-url)
To see why, consider the tableau in (34), which takes the violation profiles from (2) and applies a new constraint weighting.

(34) Harmonic bounding subverted

<table>
<thead>
<tr>
<th>( i_1 )</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( o_{11} )</td>
<td>-1</td>
<td>-1.0</td>
</tr>
<tr>
<td>( o_{12} )</td>
<td>-1</td>
<td>+2.0</td>
</tr>
<tr>
<td>( o_{13} )</td>
<td>-1</td>
<td>+3.0</td>
</tr>
</tbody>
</table>

In OT, output candidate \( o_{13} \) could never be optimal. As its violations are a proper superset of those of one of the other candidates (in fact, of both of them), this candidate is harmonically bounded (Samek-Lodovici and Prince 1999, 2005). Prince (2002) points out that OT harmonic bounding is preserved in a version of HG that limits weights to positive values (which we call *Positive HG*). Generalizing to the case of a version of Positive HG that permits constraints to assess positive satisfaction and negative violation scores, we can say that a candidate Can-A harmonically bounds another candidate Can-B iff Can-A and Can-B are members of the same candidate set, and the error vector produced by subtracting Can-B’s constraint scores from those of Can-A contains at least one positive value, and no negative ones (Becker and Pater 2007).

Another way of understanding the consequence of having both negative and positive weights is by noting that if a single constraint can be weighted negatively and positively, it can both reward and punish the presence of a single structure. Such constraints would behave very differently from those of OT.

Pater, Potts and Bhatt (2007b) provide a method for determining whether a set of input-output pairs can be rendered jointly optimal in positive HG, which makes use of linear programming’s simplex algorithm. This functions as the HG equivalent of the recursive version of CD, whose inconsistency detection properties are particularly useful for calculating predicted typologies (see esp. Hayes, Tesar and Zuraw 2003 on typology calculations using recursive CD).

We have not yet attempted to develop a version of HG-GLA that limits weights to positive values. However, we can test the present version of HG-GLA with a grammar model that does impose a positivity restriction. When supplied with a constraint weight with a value that is less than one, this grammar model replaces it with 1. Thus, even though the learning algorithm might give a constraint a negative weight, it will always be interpreted by the grammar as having a minimum value of 1.

We tested 100,000 learners in the fashion described in §3.4 on datasets originally
generated by random HG grammars, whose weights were sampled from a uniform distribution between 2.0 and 18.0. All 100,000 learners found a correct weighting for every one of these datasets.

4. Convergence of HG-GLA for noisy learners

As discussed in the introduction, variation between optima can be accounted for by adding noise to constraint weights. In the grammar of such a stochastic language, as well as in the grammar that the learner maintains during acquisition, the constraint strengths that appear in the evaluations of inputs, such as those written in the top rows of (2) and (3), are not \( w_k \) but \( w_k + N_k \), where \( N_k \) is a Gaussian random variable that is temporarily added to each constraint strength at evaluation time, as in (7). This kind of additive noise was introduced to OT by Boersma (1997) and was first combined with HG in Boersma and Weenink’s (2007) Praat implementation. Jesney (2007), Pater et al. (2007a), Boersma and Escudero (to appear), and Coetzee and Pater (to appear), apply Noisy HG to the modelling of variation. As well as accounting for variation in a target language, noise in the learner’s grammar plays a role in modelling gradual acquisition. In combination with a learning algorithm that adjusts the grammar gradually (e.g. OT-GLA or HG-GLA), noisy evaluation produces realistic sigmoid learning curves (see Boersma 1998, Boersma and Levelt 2000 on OT-GLA; also Jäger 2003/to appear for gradual learning in Maximum Entropy grammars).

Section 3 showed that HG-GLA finds correct constraint weights for categorical languages if the learner’s processing involves no evaluation noise. The present section shows that HG-GLA also finds correct weights for categorical languages if the learner’s processing does include evaluation noise. We show this first by extending the proof of section 3 to this case. We then provide learning simulations that show that this result holds in practice, and compare the HG results with those obtained for OT-GLA with Stochastic OT.

4.1 Convergence proof of the learning procedure for HG: noisy learner

Instead of by (10), the harmony of an output candidate \( o_{mn} \) given an input \( i_m \) is now given by

\[
H_{mn} = \sum_{k=1}^{K} (w_k + N_k)s_{mnk}
\]

The lower bound on the sum of the squares of the constraint weights in (22) is still valid. The computation of the upper bound, however, changes, because (24) is no longer true. Instead, it is
evaluation incorporates learning. Categorical

4.2 Tests of convergence for HG-GLA and OT-GLA: noisy learners

Now suppose that the evaluation noise has a maximum absolute value during acquisition of \( N_{\text{max}} \); for instance, if acquisition lasts 100,000 input-output pairs, \( N_{\text{max}} \) tends to be in the vicinity of \( 6N \), where \( N \) is the standard deviation of the evaluation noise. Equation (26) thus becomes

\[
\sum_{k=1}^{K} w_k^2(t) \leq \sum_{k=1}^{K} w_k^2(t-1) + \varepsilon^2 \sum_{k=1}^{K} \left( s_{(i)(j)(k)} - s_{(i)(r)(k)} \right)^2 - 2\varepsilon \sum_{k=1}^{K} N_k(t-1) \left( s_{(i)(j)(k)} - s_{(i)(r)(k)} \right)
\]

In the end, convergence is guaranteed by

\[
t \leq \frac{\varepsilon KD_{\text{max}}^2 + 2KN_{\text{max}}D_{\text{max}}}{2\varepsilon^2} - \frac{W_0 \cdot U}{\varepsilon \delta} + \sqrt{\left( \frac{\varepsilon KD_{\text{max}}^2 + 2KN_{\text{max}}D_{\text{max}}}{2\varepsilon^2} - \frac{W_0 \cdot U}{\varepsilon \delta} \right)^2 + \frac{|W_0|^2 (W_0 \cdot U)^2}{\varepsilon^2 \delta^2}} + \frac{W_0 \cdot U}{\varepsilon \delta}
\]

The value for \( t_{\text{max}} \) that one can derive from (38) is, other than in (33), no longer independent of the language data stochastically presented to the learner. This is because \( N_{\text{max}} \) is a property of the learning process rather than a property of the unit target grammar, the initial state, or the learning algorithm. However, the probability that \( N_{\text{max}} \) exceeds e.g. \( 20N \) during the lifetime of a human learner is vanishingly small (it is in the order of \( 10^{-50} \) if the learner hears 100,000,000 utterances); and if it should occur, we are still allowed to take e.g. \( N_{\text{max}}=100N \) without breaking (38). For all practical purposes, then, \( t_{\text{max}} \) will exist. Formally speaking, the algorithm converges almost surely, i.e. it converges with probability one.

4.2 Tests of convergence for HG-GLA and OT-GLA: noisy learners

We first test HG-GLA learners on their ability to converge on correct grammars for categorical (non-varying) languages, when the learners use evaluation noise in learning. The procedure is the same as that in §3.2, except that the learner incorporates a constraint weight noise with a standard deviation of \( N = 2.0 \) during the evaluation of every incoming input-output pair. That is, the learner takes the observed input, computes a weighting of her constraints by temporarily adding evaluation
noise, and computes the optimal output. If this optimal output is different from the ‘correct’ output that the learner has observed in the given input-output pair, an ‘error’ has occurred and the learner changes some constraint weights. We set the plasticity to a value of 0.1, so that it will take the learner multiple errors to overcome the noise. Again we check the learner’s grammar after every 10 learning data (input-output pairs). We judge that the learner has converged on a correct grammar if we find that the learner’s grammar produces correct outputs for 1,000 inputs randomly drawn from the set of 20 possible inputs, if the evaluation noise is reduced to a very small value of $10^{-6}$. Figure 2 shows a histogram of the convergence times.

![Histogram of convergence times of 100,000 Noisy HG learners.](image)

All 100,000 Noisy HG learners converge, in times comparable to those of the categorical HG learners of §3.2.

To get an idea of what a typical learning path looks like, we followed one learner in detail. Initially, this learner had all constraints weighted at 10.0 (like all others). For some of the 20 possible inputs, the learner already computes a correct output almost 100 percent of the time, if her evaluation noise is 2.0; for some other inputs, she scores close to zero percent correct. For $i_{19}$, the learner initially scores 12.6 percent correct (this we measure by feeding the learner’s grammar 1,000,000 tokens of $i_{19}$, and dividing the number of correct outputs by 1,000,000). Thus, 12.6 percent is the learner’s initial proficiency for $i_{19}$. We then feed the learner 3,000 learning data randomly drawn from the set of 20 ‘correct’ input-output pairs. After each learning datum, we compute her proficiency in the way just described, i.e. with 1,000,000 tokens of $i_{19}$ and an evaluation noise with a standard deviation of 2.0. Figure 3 shows how the proficiency develops in time. This figure, then, shows this learner’s learning curve for $i_{19}$.
The learning is jumpy: there are large jumps up when the learner makes an error on $i_{19}$, and typically small steps up or (more often) down when the learner makes an error on any other input. Some U-shaped learning can also be discerned. In the end, however, the learner moves towards 100 percent correctness on $i_{19}$ and all the other inputs.

We now evaluate Stochastic OT learners with OT-GLA in the same way. OT-GLA differs from HG-GLA in only two ways. First, the grammar model differs in that each time the numerical constraint values are used to evaluate a candidate set, they are converted to a corresponding ranking. Second, the learning procedure differs in the formulation of the update rule. When OT-GLA changes the ranking value of a constraint it simply moves up or down by an amount $\varepsilon$ (if the correct form does worse on the constraint than the incorrect optimum, the constraint moves down; if it does better, the constraint moves up). That is, the HG update rule in (12) is replaced by the OT update rule in (39).

$$w_k' = w_k + \varepsilon \cdot \text{sgn}(s_{pqk} - s_{prk})$$

As we noted beneath (8), in the HG update rule the amount of change is proportional to the size of the difference between the score incurred on the constraint by the correct form and by the incorrect optimum. This difference in the update rules is parallel to the difference in how the grammar models work. Under OT’s strict domination, the number of violations of a lower ranked constraint is irrelevant if a candidate is preferred by a higher ranked one, whereas HG permits gang effects, as illustrated in (8).

We tested Stochastic OT learners with OT-GLA on 100,000 datasets generated by a random total OT ranking: 20 inputs, 20 candidates per tableau, 20 constraints, all constraints initially ranked at the same height, evaluation noise during learning 2.0,
plasticity 0.1. Only 96.1% of the learners converged.5 This result shows that the convergence failures of the stochastic OT/OT-GLA combination are not limited to carefully engineered problems like that of Pater (2008).

One might wonder if the higher convergence rates of Noisy HG with HG-GLA than Stochastic OT with OT-GLA can be attributed solely to a difference in either the grammar models or the learning algorithms. To test this, one can switch the learning algorithms between grammar models. It turns out that Stochastic OT has more misconvergence with HG-GLA than with OT-GLA, and that with OT-GLA, Noisy HG does show misconvergences. Apparently, HG-GLA is the ‘best’ on-line learning algorithm for Noisy HG, and OT-GLA is the ‘best’ learning algorithm for Stochastic OT (until a fully convergent alternative is found).

The fact that HG-GLA continues to converge when noise is added to the grammar model allows it to combine the strengths of OT-GLA (representation of variable outcomes, learning in noise, gradual learning) with the strength of EDCD (guaranteed convergence on categorical outcomes). Further research will be required to better understand the exact nature of the differences between the variable systems produced by stochastic OT, Noisy HG, and by the Maximum Entropy approach to stochastic HG (see fns. 1, 2). In addition, further work is required to extend our convergence proof to the case of learning systems with variation (cf. Fischer 2005 on Maximum Entropy learning). However, in the tests of probability matching that we have run, there have been no noticeable differences between HG-GLA, OT-GLA, and an implementation of Jäger’s (2003/to appear) Maximum Entropy learner (see Coetzee and Pater 2008 for some examples).

5. Generalizing from the finite dataset

Our proof assumes a finite dataset from which the input-output pairs supplied to the learner are randomly drawn. This describes exactly the situation in many, if not all, OT/HG learning simulations, and the proof is therefore of considerable value as an underpinning of future work in this area. From the perspective of learning theory, however, this assumption raises an important issue. Convergence on a finite dataset

---

5 The highest success rate we have found for OT-GLA occurs for non-noisy learners who allow ties between constraints, i.e. constraints that have the same ranking are collapsed into a single constraint whose violation count for each candidate is the sum of the violations of the original constraints (as in the stratified hierarchies of CD in TS). Such learners (noise 0, plasticity 1.0) succeed in learning 99.99 percent of languages randomly generated from totally ordered OT hierarchies. However, if the learner has a tiny amount of evaluation noise, so that tied rankings will never occur, the learner performs at about 97% correct. It also does much worse if tied constraints are allowed in the target grammars. Interestingly, Error-Driven Constraint Demotion (EDCD) succeeds 100% of the time when a tiny amount of noise is added in evaluation, thus producing a totally ordered hierarchy, as in standard OT. That is, tied constraints do not appear to be crucial to the success of EDCD (cf. TS, p. 48).
may be considered unremarkable, in that one could equally create a learner that simply memorizes the correct forms. To counter this worry, we discuss in §5.1 the nature of generalization from a finite set of learning data to an infinite language in HG. This sort of generalization is what distinguishes a learner that acquires a grammar from a learner that lists correct outcomes. We also use this section to shed more light on the nature of HG learning and its similarities and differences from learning in OT. In §5.2, we draw on that discussion to consider how the assumptions of our proof might be weakened.

5.1 Generalization in HG

As in OT, a language in HG is an infinite set of input-output pairs, and as in OT, an HG learner must learn this language from a finite set of data. OT and HG generalize from finite sets of learning data to an infinite set of forms in two ways. First, they generalize through the nature of the constraints. Infinitely many input-output strings may map to the same violation profile. For example, consider the infinite set of input strings /VC\textsubscript{0}V\textsubscript{0}...C\textsubscript{n}V\textsubscript{n}/ (C = consonant, V = vowel), with any number of repeating CV sequences after the initial V. Each of these input strings paired with the output candidate of the form [VC\textsubscript{0}V\textsubscript{0}...C\textsubscript{n}V\textsubscript{n}] will map onto a single violation of \textsc{Onset} (‘assign a violation to a V that is not preceded by C’) and each one paired with [CVC\textsubscript{0}V\textsubscript{0}...C\textsubscript{n}V\textsubscript{n}] will map onto a single violation of \textsc{Dep} (‘assign a violation to a segment in the output that lacks a correspondent in the input’). Since OT and HG are incapable of distinguishing forms that have the same violation profile, if a grammar in either theory chooses either the C-initial output or the V-initial output as optimal with some \(n\) of following CV sequences, it will make the same decision for any other \(n\).

Second, the ranking or weightings of the constraints make predictions for an infinite set of further input-output pairs that have different violation profiles than the input-output pairs that have been seen by the learner. For example, take an HG or OT grammar with the two constraints \textsc{Onset} and \textsc{Dep} that correctly accounts for data showing that leaving a single V onsetless is optimal relative to inserting a C before it. For an input string of any length, this same grammar will choose as optimal the output candidate that leaves every V onsetless whose input correspondent lacks a preceding C, regardless of how many violations of \textsc{Onset} are incurred in that string, insofar as each \textsc{Onset} violation trades off against a single \textsc{Dep} violation (see Prince 2002, Pater \textit{et al.} 2007a, Tesar 2007 for further discussion of such cases).
For a second example of generalization to an infinite set of further violation profiles, note that if we add any number of violations \( n \) to candidate \( o_{52} \) in (40), the outcome of evaluation will always be the same.\(^6\)

(40) An infinite set of violation profiles with the same outcome under a weighting

<table>
<thead>
<tr>
<th>( i_5 )</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1.5 )</td>
<td>( 1 )</td>
<td></td>
</tr>
<tr>
<td>( \sqrt{e} ) ( o_{51} )</td>
<td>(-1)</td>
<td></td>
</tr>
<tr>
<td>( o_{52} )</td>
<td>(-2-n)</td>
<td></td>
</tr>
</tbody>
</table>

These examples show that from a finite set of learning data, an HG learner will generalize to a language consisting of an infinite set of input-output pairs, due to the nature of the constraints, and the weightings that are acquired.

We have emphasized that the basic nature of generalization is similar in OT and HG. However, the exact way in which an HG and an OT learner generalize may be different, even with the same constraint set and learning data (see Prince 2007 for related discussion). From the following correct input-output pair, and its accompanying incorrect candidate, an HG and an OT learner may generalize to different languages.

(41) A learning datum

<table>
<thead>
<tr>
<th>( i_6 )</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sqrt{\ } ) ( o_{61} )</td>
<td>(-1)</td>
<td></td>
</tr>
<tr>
<td>( o_{62} )</td>
<td>(-2)</td>
<td></td>
</tr>
</tbody>
</table>

From this learning datum, the OT learner would acquire a ranking \( C_2 \gg C_1 \), and would therefore pick the first candidate in the tableau in (42) below. The HG learner, on the other hand, might deal with (41) by acquiring a weighting like the one in tableau (40) above, which would make \( o_{72} \) optimal, or one in which the weight of \( C_2 \) is greater than that of \( C_1 \), in which case \( o_{71} \) would be optimal.

---

\(^6\) Tesar (2007) uses this sort of infinite set of violation profiles to make the uncontroversial point that OT can generate languages, that is, infinite sets of input-output pairs, which HG cannot represent. In the “tableau de tableaux” in (40), OT, but not HG, can represent a language in which \( C_2 \) is optimal regardless of the size of \( n \). The consequences of this difference are subtle. In any finite set of data presented to a real or simulated learner, \( n \) will be defined, and an HG grammar will be capable of making all of the observed correct forms optimal, and will generalize to the infinite set of further possible violation profiles. The difference is in the ways the HG and OT learners generalize, as we discuss in the text.
Thus, HG and OT make different predictions for generalization. As far as we know, these predictions have never been tested in human language acquisition.

From the perspective of learning theory, the importance of this observation is that the nature of the informative data for a single target language may be different in the two frameworks. If the target language in fact contains the first candidate in (42), \(i_7, o_{71}\), then the datum in (41) will be sufficient to guarantee that the OT learner converges on a correct grammar, but it will be insufficiently informative for the HG learner. Again, as far as we know, the consequences of this difference remain to be investigated.

### 5.2 Generalizing the perceptron proof

Proofs of convergence following Gold (1967), including the CD convergence proof in TS, assume that the learning data are directly sampled from the infinite set of possible forms defined by the target language. Our adaptation of the perceptron proof, on the other hand, involves sampling from a predefined finite subset of the target language. Generalization to a correct target language is guaranteed insofar as the finite dataset contains all of the necessary informative data, in the sense illustrated just above. This assumption is not completely dissimilar from Tesar and Smolensky’s (1998: 246) assumption for the CD proof that we mentioned in §3.3 – that informative input-output pairs “are not maliciously withheld from the learner”.

Nonetheless, it is an interesting challenge to attempt to relax the assumptions of the perceptron proof so that it can be generalized to sampling directly from the target language. To do so, one would need to be able to define a margin of separation (§3.2) on this infinite set. The margin of separation can be estimated from \(D_{\text{max}}\), a measure of the maximum difference between the violation profiles of a correct target language form and any other form in the candidate set that is potentially optimal.

Convergence, then, is guaranteed if \(D_{\text{max}}\) is finite. However, in standard OT, there is no upper bound on the number of constraint violations: for example, since there is no upper bound on word or utterance length, there is no upper bound on the number of Onset and Dep violations. This problem would not obtain under an alternative view of constraints, such as proposed in finite-state formalizations of OT, in which constraints do have an upper bound on violation number (Frank and Satta 1998, Karttunen 1998, Riggle 2004). Given such an upper bound, \(D_{\text{max}}\) can be defined for

\[
\begin{array}{c|c|c}
 i_7 & C_1 & C_2 \\
\hline
 o_{71} & -1 & \\
\hline
 o_{72} & -1 & \\
\end{array}
\]
any finite constraint set. The problem also might disappear under an alternative view
of candidate generation, such as Harmonic Serialism (Prince and Smolensky
1993/2004: §2, McCarthy 2000, 2007bc), in which steps of a derivation, and hence
candidate sets at each step, are limited to a single change of some aspect of the
representation. This itself imposes upper bounds on the number of possible violations
of some, if not all, constraints (see Pater et al. 2007a for related discussion).
Furthermore, even if standard OT’s parallel evaluation with unbounded degree of
constraint violation is maintained, it might be possible to weaken the $D_{\text{max}}$ assumption
by formalizing equivalences across different violation profiles.

Generalizing to sampling from an infinite language would also involve making
precise the conditions under which it is possible to uniquely identify an optimum and
find some second-best candidate, which is also necessary to guarantee a finite margin
of separation. It appears that for any system in which the optimum can always be
defined, there will also be a second-best candidate. One class of systems with
undefined optima involve instances in which there is no upper bound on harmony in
an infinite candidate set. As in OT, positively satisfied constraints in HG can yield
such a scenario, if as in standard OT the candidate set is infinite. A simple solution
would be for constraints to assess only negative violation scores, with strictly positive
weights, in which case the maximum harmony score will always be zero.

The infinite candidate set also poses challenges to the definition of $D_{\text{max}}$, since the
violation count can increase without bound with a single candidate set. One possibility
may be to define a finite set of contenders, that is, possible optima in the candidate set
(Riggle 2004). Again, this would depend on restricting weights to positive values,
since harmonic bounding is only guaranteed under this restriction (see §3.5 above).

Since we are leaving this weakening of assumptions to further research, we would
like to emphasize that the application of the perceptron convergence proof to HG with
its current set of assumptions is of no small interest, since for all ‘practical’ purposes,
we have shown that the HG learner will succeed in finding a correct constraint
weighting, given full knowledge of structure.

6. Learning with imperfect knowledge of right and wrong

We now turn to a set of learning problems for which there is no convergence proof for
either CD or HG-GLA. These are cases in which the learner is not supplied with full
knowledge of the structure of the correct forms, that is, when there is hidden structure.
We provide an initial examination of the convergence properties of the learners
discussed here by testing them on a set of hidden structure problems that have already
been investigated for OT learners.
These learning simulations build on work by Tesar (1997a) and TS, who propose an account of the acquisition of hidden grammatical structure based on Expectation Maximization approaches to unsupervised machine learning, termed Robust Interpretive Parsing/Constraint Demotion. The learner is provided with a correct input-output pair that lacks some aspect of its structure, and uses its grammar to choose the optimal full structure that is consistent with the observed structure. In the cases we discuss here, the learner is provided with the position of stress in the output form of a word, and chooses the optimal metrical structure that is consistent with that placement of stress. The learner then uses the full structure as the basis for learning, which follows the procedure discussed above. When the learner makes an error, the grammar is adjusted on the basis of a comparison of its incorrect optimum with the correct form, where some of the structure of the latter has been provided by the current state of its own grammar. The situation is similar to the concept of a given input representation (the underlying form), a given output representation (the overt form without metrical structure), and an intermediate hidden representation (the full surface structure).

Using this approach to the learning of hidden structure (or a hidden intermediate representation), TS test the error-driven version of CD (EDCD) on 124 target languages with different patterns of stress placement that can be generated by a set of 12 metrical constraints in categorical OT. Their simulations are replicated by Boersma (2003), who also tests the Stochastic OT with OT-GLA on the same set of languages. To compare the HG learners with the OT learners, we replicate both of these earlier sets of simulations, and also test learners that operate with HG-GLA and four grammar models: categorical HG, Noisy HG, categorical Positive HG, and Noisy Positive HG. Here we only report the results, for further details of the simulations, see TS and Boersma (2003). TS ran simulations with various assumptions about the initial ranking, our simulations always have the constraints start out at an equal ranking or weighting (a value of 10.0). The categorical learners had an evaluation noise of zero and a plasticity of 1.0; the noisy learners had an evaluation noise of 2.0 and a plasticity of 0.1.

We tested each combination of grammar type and learning algorithm three times on each of the 124 languages, providing it with a distribution of the 62 overt forms of the language. TS provided the 62 overt forms in order in increasing length; in our simulations each piece of data was a random choice from this set. We judged a learner successful if after a maximum of 1000 cycles of 62 randomly drawn data its grammar renders uniquely optimal all of the target language correct forms. The results are shown in (43).
(43) Results of learning simulations with hidden metrical structure

<table>
<thead>
<tr>
<th>Learner grammar type</th>
<th>Learning algorithm</th>
<th>Percent successful</th>
</tr>
</thead>
<tbody>
<tr>
<td>OT with strata</td>
<td>EDCD</td>
<td>63.71%</td>
</tr>
<tr>
<td>OT with strata</td>
<td>OT-GLA</td>
<td>79.03%</td>
</tr>
<tr>
<td>Stochastic OT</td>
<td>OT-GLA</td>
<td>78.76%</td>
</tr>
<tr>
<td>Categorical HG</td>
<td>HG-GLA</td>
<td>80.11%</td>
</tr>
<tr>
<td>Noisy HG</td>
<td>HG-GLA</td>
<td>88.71%</td>
</tr>
<tr>
<td>Categorical Positive HG</td>
<td>HG-GLA</td>
<td>72.85%</td>
</tr>
<tr>
<td>Noisy Positive HG</td>
<td>HG-GLA</td>
<td>79.03%</td>
</tr>
</tbody>
</table>

‘OT with strata’ refers to the version of OT used in EDCD, which allows constraints to have equal rank, that is, to be placed in strata. The success rates of EDCD and of OT-GLA with Stochastic OT replicate the findings of Tesar and Smolensky (2000) and Boersma (2003). The highest success rate in our testing was obtained by HG-GLA coupled with the Noisy HG grammar model.

We emphasize that this is only an initial investigation of the hidden structure problem in HG. Even for just this instance, further research is required to understand the sources of the relative success rates of the learners in (43), and the nature of the problems posed by these metrical systems. Nonetheless, for the case of purely on-line learning of hidden structure problems (cf. Tesar 1997b et seq.), it seems that the ‘symmetric’ OT-GLA and HG-GLA algorithms, which both promote and demote constraints, have an advantage over demotion-only EDCD, and that a learner with an HG grammar may have an advantage over one using OT ranked constraints (though the Positive HG convergence rates do not exceed those of OT-GLA). Finally, we note that failures of learning algorithms may not necessarily be negative results, insofar as they can be shown to reflect the behavior of human learners (TS, p. 48, Boersma 2003).

7. The severity of the subset problem

The subset problem for OT and HG acquisition (Smolensky 1996) can be illustrated with the case of ONSET and DEP interaction discussed in §5.1. If a learner has the ranking DEP >> ONSET, and is presented only with correct forms in which ONSET is satisfied (e.g. /CV/ → [CV]), there would be no evidence that the ranking should be changed. The problem is that with this ranking an input with an onsetless syllable would also surface faithfully (e.g. /V/ → [V]): the grammar is insufficiently restrictive to characterize the difference between attested and unattested forms. To avoid this outcome, Smolensky (1996) and TS (§5.1) propose that in the initial
ranking, Markedness constraints like \textsc{onset} are placed above Faithfulness constraints like \textsc{dep}.

Hayes (2004) and Prince and Tesar (2004) show that the initial M $\gg$ F ranking is insufficient to solve all restrictiveness problems. The issue can be illustrated by splitting \textsc{dep} into two separate constraints: one that penalizes an output consonant in word-initial position that lacks an input correspondent (\textsc{dep-initial}), and one that penalizes such a consonant in word-medial position (\textsc{dep-medial}). Given an initial ranking $\{\text{onset} \gg \text{dep-initial}, \text{dep-medial}\}$, and a learning datum with a word initial onsetless syllable (e.g. /VCV/ $\rightarrow$ [V.CV]), the ranking returned by \textsc{edcd} would be $\{\text{dep-initial}, \text{dep-medial} \gg \text{onset}\}$. The problem is that this ranking would also allow medial onsetless syllables (e.g. /CVV/ $\rightarrow$ [CV.V]). To deal with this sort of problem, Hayes (2004) and Prince and Tesar (2004) propose versions of recursive CD that maintain biases for low faithfulness throughout learning. These algorithms are considerably more elaborate than \textsc{edcd}.

Boersma and Levelt (2003) point out that subset problems are less severe for \textsc{ot-gla} than \textsc{edcd}, due to the fact that \textsc{ot-gla} both promotes and demotes constraints. For example, if we give \textsc{onset} an initial value of 10, and \textsc{dep-initial} and \textsc{dep-medial} initial values of 0, then repeatedly exposing \textsc{ot-gla} to the /VCV/ $\rightarrow$ [V.CV] mapping will produce a final state in which \textsc{dep-initial} = 6, \textsc{onset} = 4 and \textsc{dep-medial} = 0 (if plasticity = 1). Because \textsc{dep-initial} is promoted, \textsc{onset} remains above \textsc{dep-medial}, and unlike the final state grammar produced by \textsc{edcd}, this one rules out medial onsetless syllables (e.g. /CVV/ $\rightarrow$ [CV.CV]). \textsc{hg-gla} yields exactly the same result as \textsc{ot-gla}.

Jesney and Tessier (2007) point out that when faithfulness constraints are in a specific-to-general relation, the \textsc{hg-gla} yields a more restrictive end-state than \textsc{ot-gla}. As an example, we can use \textsc{onset} with the specific \textsc{dep-initial} and the general \textsc{dep}, with the same initial state values as above (\textsc{onset} = 10 and \textsc{dep}, \textsc{dep-initial} = 0). If the /VCV/ $\rightarrow$ [V.CV] mapping is then given to either \textsc{ot-gla} or \textsc{hg-gla}, \textsc{onset} will be demoted by 1, and both faithfulness constraints will be promoted by 1. The difference in outcomes between \textsc{ot-gla} and \textsc{hg-gla} comes from the fact that \textsc{hg-gla} will stop making errors when the sum of the weights of \textsc{dep} and \textsc{dep-initial} exceeds that of \textsc{onset}. Given the assumptions about noise and plasticity from the last paragraph, the \textsc{hg-gla} final state grammar will be \textsc{onset} = 6, \textsc{dep} = 4 and \textsc{dep-initial} = 4. The correct /VCV/ $\rightarrow$ [V.CV] is optimal because insertion of a word-initial consonant violates both faithfulness constraints, and is with this weighting worse than an \textsc{onset} violation. Because \textsc{onset} remains above \textsc{dep}, medial onsetless syllables are ruled out. \textsc{ot-gla}, on the other hand, would end up promoting both faithfulness constraints above \textsc{onset}, thus yielding a grammar that
accepts medial onsetless syllables. EDCD would also produce this less restrictive outcome, as would the biased version of CD in Prince and Tesar 2004 (cf. Hayes 2004, Tessier 2006).

In sum, the structure of both the learning algorithm and the grammar module in HG-GLA can lead to more restrictive learning outcomes than either EDCD or OT-GLA, and even a biased version of CD. This provides reason to be optimistic that the overall severity of subset problems is reduced in this approach to learning, and that complexities analogous to those introduced in the revisions to CD in Hayes (2004) and Prince and Tesar (2004) can be avoided.

8. Conclusions

This paper provides a set of basic learning results for HG. We have shown that a simple on-line gradual learning algorithm, the HG-GLA, is guaranteed to converge on a correct set of weights for categorical choices of optima, when it is supplied with full knowledge of the structure of the language data. We have also shown that the learner continues to be successful in this regard when it is used with a modified version of the HG grammar model, Noisy HG. Like Stochastic OT, Noisy HG provides a grammar model that can represent variation, and allows for learning in noise, and for the modelling of gradual acquisition. These results provide a foundation for further research on the formal properties of this approach to learning, as well as research on the modelling of human language acquisition. The further development of HG learning theory has a large body of extant research to draw upon, since HG’s linear model is widely used in computational approaches to learning. Our results draw on that strength, especially in the adaptation of the perceptron convergence proof. In addition to these core results, we have discussed initial investigations into the hidden structure problem and the subset problem in HG learning, which provide some indications that there may be advantages for HG over OT in these domains.

References


Gold, Mark. 1967. Language identification in the limit. Information and Control, 10: 447-474


Karttunen, Lauri. 1998. The proper treatment of optimality in computational phonology. Ms., Xerox Research Centre Europe, Meylan, France. [ROA 258].


Pater, Joe, Rajesh Bhatt and Christopher Potts. 2007a. Linguistic Optimization. Ms, University of Massachusetts, Amherst. [ROA-924]

Pater, Joe, Christopher Potts and Rajesh Bhatt. 2007b. Harmonic Grammar with Linear Programming. Ms, University of Massachusetts, Amherst. [ROA-827]


Prince, Alan. 2007. Let the decimal system do it for you: a very simple utility function for OT. Ms, Rutgers University. [ROA-943].


Smolensky, Paul. 1996. The initial state and ‘Richness of the Base’ in Optimality Theory. Ms, Johns Hopkins University. [ROA-154].


