A closer look at Boersma and Hayes' Ilokano metathesis test case

Giorgio Magri and Benjamin Storme
CNRS/Univ. Paris 8 and MIT

An *Error-driven* learner maintains a current grammar, which represents its current hypothesis of the target adult grammar. The learner is exposed to a stream of data, one piece of data at the time. Whenever the current grammar is found to be inconsistent with the current piece of data, the current grammar is slightly updated, in a way that takes into account the nature of the failure on the current piece of data. Boersma (1998) develops an implementation of error-driven learning within the framework of stochastic OT, called the *Gradual Learning Algorithm* (henceforth: GLA). Boersma & Hayes (2001) (henceforth: BH) report that the GLA succeeds at learning variation in three complex realistic test cases. Furthermore, they report that variants of the GLA (which differ only for small details of the rule used to update the current grammar) instead fail.

The success of the GLA implementation of error-driven learning on BH's test cases is surprising, as nothing is built into the error-driven learning scheme to guide the learner towards probability matching. Indeed, it is not hard to construct artificial cases of variation where the GLA fails.[1] We thus submit that the proper interpretation of BH's successful simulation results is the following: the patterns of variation in BH's test cases have some special structure (hopefully shared by other cases of variation in Natural Language) which the GLA (but not variants thereof which adopt slightly different update rules) is crucially able to exploit. Thus interpreted, BH's simulation results of course raise the following question: what is this special structure displayed by BH's test cases (and allegedly shared by other cases of variation in Natural Language) which allows the GLA to succeed? In Magri & Storme (in prep.) (henceforth: MS), we address this question through detailed analyses of the behavior of the GLA (and variants thereof) on BH's three test cases. Here, we offer a preview of our analyses, focusing on BH's Ilokano metathesis test case.

1 Stochastic error-driven ranking algorithms

Boersma (1998) develops the *stochastic OT* grammatical framework, a modification of the standard OT framework which is able to model grammatically conditioned language variation. Let us start with a brief review of the notion of a stochastic OT grammar, in order to set the notation used throughout the paper. Assume that the constraint set consists of n constraints $C_1, ..., C_n$. A *ranking vector* is an n-tuple $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)$ of arbitrary numbers; the kth number θ_k is called the *ranking value*

[1] Here is one such case (see MS for details). BH's Finnish genitive test case is based on Anttila (1997). BH (p. 68) write: "we found that we could derive the corpus frequencies using only a subset of [Anttila's] constraints." It turns out that BH's constraint set can be further pruned of the constraints *LAPSE, *Ó ,*Á, *Ĭ without affecting the simulation results. Yet, the GLA fails on a minimal modification of the latter test case, which only differs because it lacks the two forms /luettelo/ and /korjaamo/.

of constraint C_k. A *noise vector* is an n-tuple $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_n)$ of numbers sampled independently from each other according to some continuous distribution $\mathcal{D}$; the kth number ϵ_k is called the *noise value* of constraint C_k. The result of corrupting a ranking vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)$ through a noise vector $\boldsymbol{\epsilon} = (\epsilon_1, ..., \epsilon_n)$ is the new ranking vector $\boldsymbol{\theta} + \boldsymbol{\epsilon} = (\theta_1 + \epsilon_1, ..., \theta_n + \epsilon_n)$ obtained by corrupting each ranking value through the corresponding error value; the kth component $\theta_k + \epsilon_k$ is called the *corrupted ranking value* of constraint C_k.

Since the noise values ϵ_k are sampled according to a distribution $\mathcal{D}$ which is continuous, the probability that the corrupted ranking vector $\boldsymbol{\theta} + \boldsymbol{\epsilon}$ has two identical components is null. Thus, it represents the unique constraint ranking $\gg_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}$ which ranks a constraint C_k above another constraint C_h iff the corrupted ranking value $\theta_k + \epsilon_k$ of the former is larger than the corrupted ranking value $\theta_h + \epsilon_h$ of the latter. The *stochastic grammar* $\mathsf{OT}^{\mathcal{D}}_{\boldsymbol{\theta}}$ corresponding to a ranking vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)$ and a continuous distribution $\mathcal{D}$ is the grammar which takes an underlying form x and returns the candidate $y = \mathsf{OT}_{\gg_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}}(x)$ which is the winner (in the standard OT sense) according to the constraint ranking $\gg_{\boldsymbol{\theta}+\boldsymbol{\epsilon}}$ corresponding to the corrupted ranking vector $\boldsymbol{\theta} + \boldsymbol{\epsilon}$, where the noise $\boldsymbol{\epsilon}$ has been sampled according to the distribution $\mathcal{D}$. Since the winner depends on the random noise vector $\boldsymbol{\epsilon}$, stochastic OT grammars provide a tool to model grammatically conditioned language variation.

Within the framework of stochastic OT, the error-driven learning scheme informally sketched at the beginning of the paper can be formalized as the (*stochastic*) *error-driven ranking algorithm* (EDRA) described in Table 1. The algorithm maintains a current stochastic OT grammar, represented through a current vector $\boldsymbol{\theta}$ of ranking values. These current ranking values are initialized by setting them equal to certain initial values. Following BH, we set the initial ranking values all equal to 100. These initial ranking values are then updated by looping through four steps. At step (I), the algorithm receives a piece of data consisting of an underlying form x together with a corresponding intended winner candidate y. At step (II), the algorithm computes the candidate z predicted to be the winner for the underlying form x according to the stochastic grammar $\mathsf{OT}^{\mathcal{D}}_{\boldsymbol{\theta}}$ corresponding to the current ranking vector $\boldsymbol{\theta}$ and a certain distribution $\mathcal{D}$ used to sample the noise values. Following BH, we assume that $\mathcal{D}$ is a gaussian distribution with zero mean and a small variance, called the *noise* parameter. If the predicted winner z coincides with the intended winner y, the current ranking vector has succeeded on the current piece of data. The EDRA thus has nothing to learn from the current piece of data, loops back to

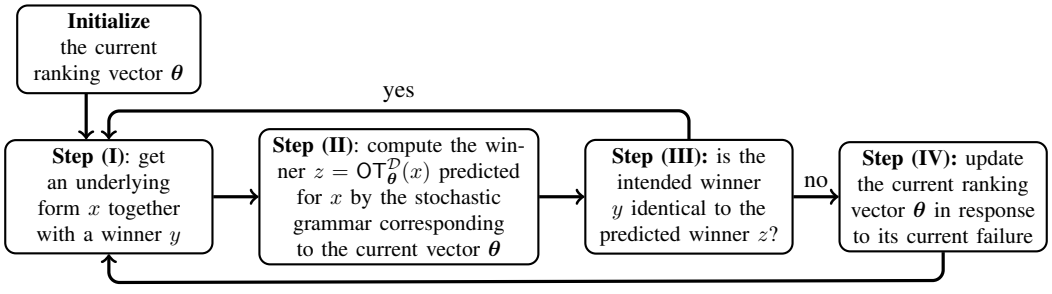


Table 1: Error-driven learning within stochastic OT

step (I), and waits for more data. If instead the predicted winner z differs from the intended winner y, then the current ranking vector needs to be updated at step (IV) in response to that failure.

The current failure says in particular that the constraints which prefer the loser z (namely assign less violations to z than to y) are currently ranked too high while the constraints which instead prefer the intended winner y (namely assign less violations to y than to z) are ranked too low. At step (IV), the algorithm tries to remedy to this situation by slightly modifying the current ranking values. In the general case, this re-ranking rule has two components. It has a *demotion* component, which decreases the ranking values of at least certain loser-preferring constraints by a certain *demotion amount*, say 1 for concreteness. Furthermore, it has a *promotion* component which increases the ranking values of the winner-preferring constraints by a certain *promotion amount* $p \geq 0$, which can be null or positive. The demotion and promotion amounts can be rescaled by a positive small constant, called *plasticity*.

Four implementations of this update scheme have been considered in the literature, summarized in Table 2. The GLA re-ranking rule demotes all loser-preferring constraints. The other three re-ranking rules instead only demote the *undominated* loser-preferring constraints, namely those loser-preferrers that indeed need to be demoted, because their current corrupted ranking values are not smaller than the current corrupted ranking value of some winner-preferring constraint. Furthermore, the EDCD re-ranking rule performs no constraint promotion at all, namely assumes a null promotion amount: $p = 0$. The GLA and minGLA re-ranking rules perform as much constraint promotion as demotion, namely assume a promotion amount equal to the demotion amount: $p = 1$. And the CEDRA re-ranking rule assumes a promotion amount strictly smaller than the inverse of the number w of winner-preferring constraints promoted.[2]

Name of the update rule	Promotion amount	Which constraints are demoted
Error-Driven Constraint Demotion (**EDCD**; Tesar & Smolensky 1998):	$p = 0$	only the *undominated* loser-preferring constraints
Gradual Learning Algorithm (**GLA**; Boersma 1998)	$p = 1$	*all* loser-preferring constraints
minimal Gradual Learning Algorithm (**minGLA**; Boersma 1998)	$p = 1$	only the *undominated* loser-preferring constraints
Calibrated error-driven ranking algorithm (**CEDRA**; Magri 2012):	$p < 1/w$	only the *undominated* loser-preferring constraints

Table 2: Four re-ranking rules considered in the literature

2 The Ilokano metathesis test case

One of BH's test cases concerns two areas of free variation in Ilokano: an optional process of metathesis and variation in the form of the reduplicates. In this paper,

[2]For the CEDRA simulations reported below, we have used the promotion amount $p = \frac{1}{w+1}$.

Underlying forms	Candidates	Probabilities of the candidates	Constraints
/paʔlak/	[paʔ.lak]	0.0	ONSET
	[pa.lak]	**1.0**	MAX_{IO}(V)
	[pa.ʔlak]	0.0	*LOWGLIDE
	[pal.ʔak]	0.0	IDENT_{IO}(LOW)
/ʔajo-en/	[ʔa.jo.en]	0.0	DEP_{IO}(ʔ)
	[ʔaj.wen]	**1.0**	IDENT_{IO}(SYLLABIC)
	[ʔa.jo.ʔen]	0.0	*[ʔC
	[ʔa.jen]	0.0	MAX_{OO}(ʔ)
/basa-en/	[ba.sa.en]	0.0	LINEARITY
	[bas.a͡en]	0.0	*ʔ]
	[bas.͡wen]	0.0	MAX_{IO}(ʔ)
	[ba.sa.ʔen]	**1.0**	
	[ba.sen]	0.0	
/taʔo-en/	ta.ʔo.en	0.0	
	taʔ.wen	**0.5**	
	taw.ʔen	**0.5**	
	ta.wen	0.0	
	ta.ʔwen	0.0	
	ta.ʔo.ʔen	0.0	
	ta.ʔen	0.0	

Table 3: BH's Ilokano metathesis test case

we focus on the metathesis test case, summarized in Table 3.[3] There are four underlying forms, listed in the left column of the table; BH assume they have the same frequency. The corresponding candidates are listed together with their probability of being a winner (conditioned on the underlying form). The underlying form /paʔlak/ illustrates the fact that Ilokano bans glottal stops in coda position, which happen to undergo deletion. The concatenation of the stems /ʔajo/ and /basa/ with the suffix /en/ cannot surface faithfully, because Ilokano does not allow onsetless syllables. When the stem-final vowel is /o/, Ilokano glides that vowel and syllabifies the glide as the missing onset. When the stem-final vowel is instead /a/, Ilokano epenthesizes an onset glottal stop. The processes just described are deterministic: only one candidate wins, with probability 1.0. These processes interact when the onsetless suffix /-en/ is concatenated with the stem /taʔo/ containing a glottal stop. Again, Ilokano repairs the missing onset by gliding the stem-final vowel /o/ and then syllabifies that glide as the missing onset. Unfortunately, this repair strategy stands the glottal stop of the stem in coda position. As seen above, in an underived environment such as /paʔlak/, coda glottal stops are deleted. But in the derived environment /taʔo-en/, Ilokano displays free variation between faithful production of the coda glottal stop and metathesis with the following glide. BH assume that the two candidates [taʔ.wen] and [taw.ʔen] win with equal probability of 0.5. BH offer a detailed account of these phonological patterns within stochastic OT in terms of the constraints listed in the right column of Table 3.

[3] We have made small modifications with respect to BH, which have no impact neither on the simulation results nor on the analysis of those results; see MS for discussion.

We can now study the behavior of the stochastic EDRA on the Ilokano metathesis test case. In Table 4, we report the final ranking values learned by the algorithm, for each of the four re-ranking rules listed in Table 2. We have used the same implementation details used by BH (p. 79): a learning simulation consists of three stages of 7,000 iterations each, with deceasing plasticity (2.0, 0.2, and 0.02) and decreasing noise (10.0, 2.0, and 2.0). The final ranking vector reported here for the GLA is almost identical to the one reported by BH. BH note that multiple runs of the GLA yield almost identical final ranking values; the same holds for the other update rules; the values reported are thus characteristic of the behavior of the algorithm.

The quality of the final ranking vectors reported in Table 4 is evaluated in Table 5. In the first two columns of the table, we list all pairs of an underlying form and a corresponding candidate together with their actual probabilities. In the four remaining columns, we provide the frequencies of that mapping predicted by the final ranking vectors reported in Table 4 (noise: 2.0; number of repetitions: 100,000). We see that all four algorithms manage to learn the stochastic behavior of the underlying form /taʔo-en/. Furthermore, all four algorithms also manage to learn the deterministic behavior of the underlying forms /ʔajo-en/ and /basa-en/. The critical test case is the underlying form /paʔlak/ at the top of the table: both the GLA and GLAmin succeed at learning its deterministic behavior, while the CEDRA comes short on the task and EDCD fails. What makes /paʔlak/ hard to learn? How do the GLA and GLAmin succeed? Why is it that EDCD and CEDRA instead fail? In the rest of this paper, we provide an analytical answer to these questions.

3 Simplifying the Ilokano metathesis test case

To start, we describe the Ilokano metathesis test case in ERC (*elementary ranking condition*; Prince 2002) notation, as in Table 6a. We consider all possible triplets of an underlying form, *a* corresponding winner (namely any candidate for that underlying form which has a non-null probability of winning) and any other candidate different from that winner, which therefore counts as a loser. For instance, the first triplet (/paʔlak/, [pa.lak], ~~[pa.ʔlak]~~) consists of the underlying form /paʔlak/, its unique winner [pa.lak], and one of its loser candidates, in this case ~~[pa.ʔlak]~~. We adopt the convention of striking out the loser in each triplet, in order to distinguish it from the winner. The remaining triplets in the first block are obtained by considering all possible loser candidates for this underlying form. The next two blocks corresponding to the underlying forms /ʔajo-en/ and /basa-en/ are constructed analogously. Finally, the underlying form /taʔo-en/ comes with two winners, namely [taw.ʔen] and [taʔ.wen]. Hence, it corresponds to two blocks of triplets, each corresponding to this underlying form, one of the two winners and all remaining loser candidates. Each such underlying/winner/loser form triplet sorts the constraints into winner-preferring (i.e. those which assign less violations to the winner than to the loser), loser-preferring (i.e. those which assign less violations to the loser than to the winner), and even (i.e. those which assign the same number of violations to the loser and to the winner). We thus classify each constraint accordingly, by writing W (or L) in correspondence of winner-preferring (or loser-preferring, respectively) constraints (while the entry corresponding to even constraints is left empty). To illustrate, the entry corresponding to the first ERC (/paʔlak/, [pa.lak], ~~[pa.ʔlak]~~) and

GLA		**GLAmin**	
IDENT$_{IO}$([LOW])	142.0	MAX$_{IO}$(V)	148.0
MAX$_{IO}$(V)	140.0	*LOWGLIDE	146.0
ONSET	138.0	ONSET	144.0
*LOWGLIDE	138.0	IDENT$_{IO}$([LOW])	144.0
*[ʔC	114.0	*[ʔC	118.0
MAX$_{OO}$(ʔ)	110.0	MAX$_{OO}$(ʔ)	116.0
DEP$_{IO}$(ʔ)	98.0	DEP$_{IO}$(ʔ)	106.0
LINEARITY	67.0	LINEARITY	76.1
*ʔ]	66.9	*ʔ]	75.8
MAX$_{IO}$(ʔ)	24.0	IDEN$_{IO}$([SYL])	64.0
IDEN$_{IO}$([SYL])	24.0	MAX$_{IO}$(ʔ)	34.0

EDCD		**CEDRA**	
ONSET	100.0	ONSET	113.0
MAX$_{OO}$(ʔ)	100.0	IDENT$_{IO}$([LOW])	111.3
MAX$_{IO}$(V)	100.0	MAX$_{IO}$(V)	111.0
IDENT$_{IO}$([LOW])	100.0	*LOWGLIDE	108.6
*[ʔC	100.0	*[ʔC	103.0
*LOWGLIDE	100.0	MAX$_{OO}$(ʔ)	100.0
DEP$_{IO}$(ʔ)	50.0	DEP$_{IO}$(ʔ)	64.0
IDEN$_{IO}$([SYL])	10.0	IDEN$_{IO}$([SYL])	22.0
*ʔ]	-897.9	*ʔ]	-304.3
LINEARITY	-898.0	LINEARITY	-304.5
MAX$_{IO}$(ʔ)	-900.8	MAX$_{IO}$(ʔ)	-309.1

Table 4: Ranking values learned by the EDRA trained on the Ilokano test case

the markedness constraint *[ʔC is set equal to W because that constraint is winner-preferring, as it is violated by the loser [pa.ʔlak] but not by the winner [pa.lak]. The ERC matrix thus obtained provides a summary of all the actions that the EDRA can take. In fact, each update is triggered by one of these ERCs and the update can be described in terms of the corresponding pattern of W's and L's: promote constraints which have a W and demote constraints which have an (undominated) L.

The constraints corresponding to the six left-most columns of Table 6a are special in the sense that they only have W's but no L's. This means that these constraints will never be demoted and thus will never drop below their initial ranking value. Focus on the ERCs that have a W corresponding to one of these constraints. The following proposition I guarantees that these ERCs can trigger only very "few" updates (see MS for a more explicit formulation of this proposition and a proof). In other words, this proposition guarantees that the learning dynamics is governed in the long run by the remaining ERCs, which are listed in Table 6b.

Proposition I. *Consider an arbitrary run of the stochastic EDRA on the Ilokano metathesis text case. Assume that all constraints start with the same initial ranking value, that the additive error is sampled according to a gaussian distribution with small variance, and that the re-ranking rule is one of those listed in Table 2. With high probability, each of the ERCs in Table 6a which has a* W *corresponding to the six left-most constraints can trigger only very few updates.* ■

We can now repeat the same reasoning a second time. Constraint DEP$_{IO}$(ʔ) is

	actual	GLA	GLAmin	EDCD	CEDRA
(/paʔlak/, [pa.lak])	1.0	1.0	1.0	**0.7558**	**0.9128**
(/paʔlak/, [paʔ.lak])	0.0	0.0	0.0	**0.1201**	**0.0372**
(/paʔlak/, [pal.ʔak])	0.0	0.0	0.0	**0.1241**	**0.05**
(/paʔlak/, [pa.ʔlak])	0.0	0.0	0.0	0.0	0.0
(/ʔajo-en/, [ʔaj.wen])	1.0	1.0	1.0	1.0	1.0
(/ʔajo-en/, [ʔa.jen])	0.0	0.0	0.0	0.0	0.0
(/ʔajo-en/, [ʔa.jo.ʔen])	0.0	0.0	0.0	0.0	0.0
(/ʔajo-en/, [ʔa.jo.en])	0.0	0.0	0.0	0.0	0.0
(/basa-en/, [ba.sa.ʔen])	1.0	1.0	1.0	1.0	1.0
(/basa-en/, [bas.a͡en])	0.0	0.0	0.0	0.0	0.0
(/basa-en/, [ba.sen])	0.0	0.0	0.0	0.0	0.0
(/basa-en/, [ba.sa.en])	0.0	0.0	0.0	0.0	0.0
(/basa-en/, [bas.wen])	0.0	0.0	0.0	0.0	0.0
(/taʔo-en/, [taʔ.wen])	0.5	0.4958	0.5464	0.4929	0.4547
(/taʔo-en/, [taw.ʔen])	0.5	0.5042	0.4536	0.5071	0.5453
(/taʔo-en/, [ta.ʔo.en])	0.0	0.0	0.0	0.0	0.0
(/taʔo-en/, [ta.ʔen])	0.0	0.0	0.0	0.0	0.0
(/taʔo-en/, [ta.wen])	0.0	0.0	0.0	0.0	0.0
(/taʔo-en/, [ta.ʔwen])	0.0	0.0	0.0	0.0	0.0
(/taʔo-en/, [ta.ʔo.ʔen])	0.0	0.0	0.0	0.0	0.0

Table 5: Stochastic OT grammars corresponding to the ranking values in Table 4

winner-preferring but never loser-preferring according to the ERCs listed in Table 6b. Thus, this constraint is never demoted. Proposition II guarantees that those ERCs that have a W corresponding to this constraint can only trigger a very small number of updates (see MS for a more explicit formulation of this proposition and a proof). In other words, this proposition guarantees that the learning dynamics is governed in the long run by the remaining ERCs, listed in Table 6c (here, we have gotten rid of the seven constraints that are always even in the ERCs which are not guaranteed to only trigger very few updates by the two propositions I and II).

Proposition II. *Consider an arbitrary run of the stochastic EDRA on the Ilokano metathesis text case. Assume that all constraints start with the same initial ranking value, that the additive error is sampled according to a gaussian distribution with small variance, and that the re-ranking rule is one of those listed in Table 2. With high probability, each of the three ERCs in Table 6b which has a* W *corresponding to the constraint* $\text{DEP}_{\text{IO}}(\text{ʔ})$ *can trigger only very few updates.* ■

In conclusion, Table 6c displays the computational core of the Ilokano metathesis test case. This test case requires the two constraints LINEARITY and *ʔ] to have the same ranking values, in order to model the free variation between the two forms [taw.ʔen] and [taʔ.wen]. Furthermore, the constraint $\text{MAX}_{\text{IO}}(\text{ʔ})$ must have a smaller ranking value, in order to prevent the additive noise from being able to switch the relative ranking between $\text{MAX}_{\text{IO}}(\text{ʔ})$ and the other two constraints, thus ensuring that [pal.ʔak] never beats [pa.lak]. In the next two sections, we analyze the behavior of the stochastic EDRA on this case study, depending on the choice of the re-ranking rule. As the ERC matrix in Table 6c contains ERCs that all have a single L, the GLA and GLAmin re-ranking rules collapse; we thus ignore the latter.

(a)

	Onset	Max$_{IO}$(V)	*LowGlide	Ident$_{IO}$(low)	*[ʔC	Max$_{OO}$(ʔ)	Dep$_{IO}$(ʔ)	Ident$_{IO}$(syl)	Linearity	*ʔ]	Max$_{IO}$(ʔ)
(/paʔlak/, [pa.lak], ~~[pa.ʔlak]~~)					W						L
(/paʔlak/, [pa.lak], ~~[pal.ʔak]~~)									W		L
(/paʔlak/, [pa.lak], ~~[paʔ.lak]~~)										W	L
(/ʔajo-en/, [ʔaj.wen], ~~[ʔa.jen]~~)		W						L			
(/ʔajo-en/, [ʔaj.wen], ~~[ʔa.jo.en]~~)	W							L			
(/ʔajo-en/, [ʔaj.wen], ~~[ʔa.jo.ʔen]~~)							W	L			
(/basa-en/, [ba.sa.ʔen], ~~[bas.a͡en]~~)			W				L	W			
(/basa-en/, [ba.sa.ʔen], ~~[bas.a͡wen]~~)				W			L	W			
(/basa-en/, [ba.sa.ʔen], ~~[ba.sen]~~)		W					L				
(/basa-en/, [ba.sa.ʔen], ~~[ba.sa.en]~~)	W						L				
(/taʔo-en/, [taw.ʔen], ~~[taʔ.wen]~~)									L	W	
(/taʔo-en/, [taw.ʔen], ~~[ta.wen]~~)						W			L		W
(/taʔo-en/, [taw.ʔen], ~~[ta.ʔen]~~)		W						L	L		
(/taʔo-en/, [taw.ʔen], ~~[ta.ʔo.en]~~)	W							L	L		
(/taʔo-en/, [taw.ʔen], ~~[ta.ʔo.ʔen]~~)							W	L	L		
(/taʔo-en/, [taw.ʔen], ~~[ta.ʔwen]~~)					W				L		
(/taʔo-en/, [taʔ.wen], ~~[taw.ʔen]~~)									W	L	
(/taʔo-en/, [taʔ.wen], ~~[ta.wen]~~)						W				L	W
(/taʔo-en/, [taʔ.wen], ~~[ta.ʔen]~~)		W						L		L	
(/taʔo-en/, [taʔ.wen], ~~[ta.ʔo.en]~~)	W							L		L	
(/taʔo-en/, [taʔ.wen], ~~[ta.ʔo.ʔen]~~)							W	L		L	
(/taʔo-en/, [taʔ.wen], ~~[ta.ʔwen]~~)					W					L	

(b)

	Onset	Max$_{IO}$(V)	*LowGlide	Ident$_{IO}$([low])	*[ʔC	Max$_{OO}$(ʔ)	Dep$_{IO}$(ʔ)	Ident$_{IO}$([syl])	Linearity	*ʔ]	Max$_{IO}$(ʔ)
(/paʔlak/, [pa.lak], ~~[pal.ʔak]~~)								W			L
(/paʔlak/, [pa.lak], ~~[paʔ.lak]~~)									W		L
(/ʔajo-en/, [ʔaj.wen], ~~[ʔa.jo.ʔen]~~)							W	L			
(/taʔo-en/, [taw.ʔen], ~~[taʔ.wen]~~)								L	W		
(/taʔo-en/, [taw.ʔen], ~~[ta.ʔo.ʔen]~~)							W	L	L		
(/taʔo-en/, [taʔ.wen], ~~[taw.ʔen]~~)								W	L		
(/taʔo-en/, [taʔ.wen], ~~[ta.ʔo.ʔen]~~)							W	L			

(c)

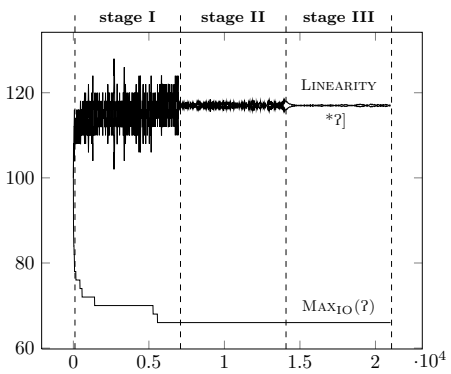
	Linearity	*ʔ]	Max$_{IO}$(ʔ)
ERC1 = (/paʔlak/, [pa.lak], ~~[pal.ʔak]~~)	W		L
ERC 2 = (/paʔlak/, [pa.lak], ~~[paʔ.lak]~~)		W	L
ERC 3 = (/taʔo-en/, [taw.ʔen], ~~[taʔ.wen]~~)	W	L	
ERC 4 = (/taʔo-en/, [taʔ.wen], ~~[taw.ʔen]~~)	L	W	

Table 6: The Ilokano metathesis test case in ERC notation: **(a)** original description; **(b)** description after the first round of simplifications; **(c)** description after the second round of simplifications.

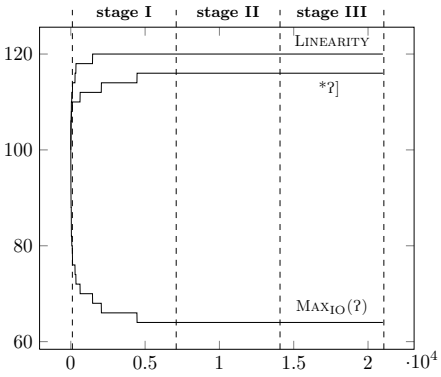
(a)

Constraint	final RV
LINEARITY	117.06
*ʔ]	116.94
MAX$_{IO}$(ʔ)	66.0

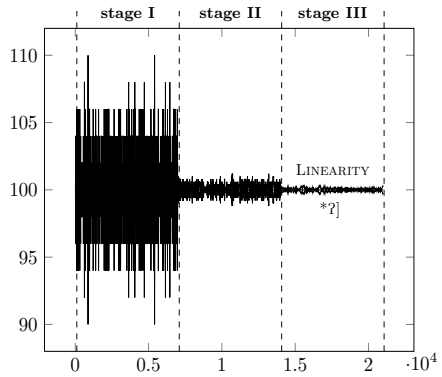
(b)



(c)



(d)



(e)

ERCs	#U	LU	LI
ERC1 = (/paʔlak/, [pa.lak], [pal.ʔak])	11	1329	5278
ERC 2 = (/paʔlak/, [pa.lak], [paʔ.lak])	6	1390	5562
ERC 3 = (/taʔo-en/, [taw.ʔen], [taʔ.wen])	2631	5228	20992
ERC 4 = (/taʔo-en/, [taʔ.wen], [taw.ʔen])	2581	5229	20999

Table 7: Behavior of the GLA on the core Ilokano metathesis test case in Table 6c

4 Why the GLA succeeds

We report in Table 7a the final ranking values learned by the GLA in a simulation on the simplified Ilokano metathesis test case described in Table 6c. Again, we use the same implementation details as in BH: a simulation consists of three stages of 7,000 iterations each, with deceasing plasticity (2.0, 0.2, and 0.02) and decreasing noise (10.0, 2.0, and 2.0). These ranking values enforce the desired ranking conditions. The corresponding dynamics of the three ranking values is plotted in Table 7b. The two constraints LINEARITY and *ʔ] raise to roughly their final ranking value in the very initial portion of the first learning stage and then just keep oscillating without moving away from that value. In the meanwhile, the constraint MAX$_{IO}$(ʔ) drops quickly within the first learning stage and then stays put, well separated underneath the other two constraints. We want to understand this learning dynamics and thus explain how the GLA manages to succeed on this test case.

To gain some intuition into the GLA's learning dynamics in Table 7b, suppose that the GLA were only trained on the underlying form /taʔo-en/. The two corresponding ERCs 3 and 4 completely disagree with each other: the former promotes

LINEARITY and demotes *ʔ] while the latter does just the opposite. Crucially, the GLA promotes and demotes winner- and loser-preferring constraints by exactly the same amount. This means that one of these two ERCs will completely undo the work of the other: one of these two ERCs promotes LINEARITY and demotes *ʔ] and the other ERC will displace them back to their original position. If the two constraints start with the same initial ranking value, then these two ERCs will never be able to displace them away from that initial ranking value and will just keep them oscillating around that value. This is illustrated by the ranking dynamics plotted in Table 7c, which is indeed the ranking dynamics obtained when the GLA is trained only on the underlying form /taʔo-en/ corresponding to the two ERCs 3 and 4. Of course, the amplitude of the oscillations of the two constraints LINEARITY and *ʔ] decreases through the three learning stages as a consequence of the decreasing plasticities, perfectly matching the oscillations of these two constraints LINEARITY and *ʔ] plotted in the original ranking dynamics in Table 7b.

Suppose next that the GLA were only trained on the other underlying form /paʔlak/. The two corresponding ERCs 1 and 2 agree with each other: they both promote the winner-preferring constraints LINEARITY and *ʔ] and demote the loser-preferring constraint $\text{MAX}_{\text{IO}}(\text{ʔ})$. After only a few updates, the two former winner-preferring constraints will be separated from the latter loser-preferring constraint by a distance large enough that the additive error will not be able to swap their relative ranking (the gaussian distribution is concentrated around zero and thus the additive noise values are small with high probability). This is illustrated be the ranking dynamics plotted in Table 7d, which is indeed the ranking dynamics obtained when the GLA is trained only on the underlying form /paʔlak/ corresponding to the two ERCs 1 and 2. Ignoring the oscillations, the shape of the original ranking dynamics plotted in Table 7b coincides with the ranking dynamics obtained in Table 7d by training the GLA on the underlying form /paʔlak/ only.

We are now ready to put the pieces together. Training on the two ERCs 3 and 4 corresponding to the stochastic underlying form /taʔo-en/ contributes to the oscillations of the original ranking dynamics in Table 7b but not to its shape. Training on the two ERCs 1 and 2 corresponding to the deterministic underlying form /paʔlak/ of course does not contribute to the oscillations, but determines the shape of the original ranking dynamics. In other words, the original ranking dynamics obtained in Table 7b when the GLA is trained on both underlying forms /taʔo-en/ and /paʔlak/ simultaneously is the "sum" of the two ranking dynamics obtained in Tables 7c-d when the GLA is trained on the two underlying forms separately. This is not an obvious fact. Indeed, the two ERCs 1 and 2 corresponding to the underlying form /paʔlak/ partially disagree with the two other ERCs 3 and 4 corresponding to the underlying form /taʔo-en/: ERCs 1 and 4 (ERCs 2 and 3) disagree on the constraint LINEARITY (on the constraint *ʔ], respectively). When the GLA is trained simultaneously on both underlying forms, we might thus expect complex interactions between the forces exerted by the two underlying forms. Because of these interactions, the ranking dynamics of the GLA trained simultaneously on both underlying forms might in principle be quite different from the "sum" of the two ranking dynamics obtained when the GLA is trained on the two underlying forms separately. As we will see in the next section, that is indeed the case for EDCD and the CEDRA. Intuitively, the reason for why the case of the GLA is different

is that in the case of the GLA (but not in the case of EDCD and the CEDRA) the promotion and demotion amounts coincide and thus ERCs 3 and 4 do not displace the constraints and therefore do not contribute to the shape of the ranking dynamics. This intuition is formalized in the proof of the following proposition III, provided in MS. Indeed, this proposition says that the two ERCs 1 and 2 corresponding to the underlying form /paʔlak/ trigger few updates, exactly as in the case where the GLA is trained on that underlying form alone. In other words, the presence of the two ERCs 3 and 4 corresponding to the stochastic underlying form /taʔo-en/ has no effects on the deterministic underlying form /paʔlak/.

Proposition III. *Consider a run of the GLA on the core Ilokano metathesis test case described in Table 6c. Assume that all constraints start with the same initial ranking value and that the additive error is sampled according to a gaussian distribution with small variance. With high probability, the two ERCs 1 and 2 corresponding to the underlying form /paʔlak/ can trigger only very few updates.* ■

Proposition III is confirmed by a closer look at the simulation results plotted in Table 6b. The column headed by "#U" in Table 6e provides the number of updates triggered by each of the four ERCs, showing that ERCs 1 and 2 indeed trigger just a few updates. These updates furthermore happen towards the beginning of the run, as shown by the last two columns of the table, which provide respectively the number of updates (column headed "LU" for "last update") by and the number of iterations (column headed by "LI" for "last iteration") at the time when each single ERC has triggered its last update.

In conclusion, proposition III provides the key to a formal understanding of the behavior of the GLA on the core Ilokano metathesis test case. Since ERCs 1 and 2 corresponding to the underlying form /paʔlak/ can trigger only few updates, their two winner-preferring constraints LINEARITY and *ʔ] raise and their loser-preferring constraint $\text{MAX}_{\text{IO}}(ʔ)$ drops quickly, ensuring the needed separation. From that moment on, the underlying form /paʔlak/ cannot trigger any further update. The learning dynamics is thus driven by the other underlying form /taʔo-en/. Since the promotion and demotion amounts coincide, its two corresponding ERCs 3 and 4 do not displace their active constraints LINEARITY and *ʔ], but just keep them oscillating up and down. The ranking dynamics in Table 7b is thus completely explained.

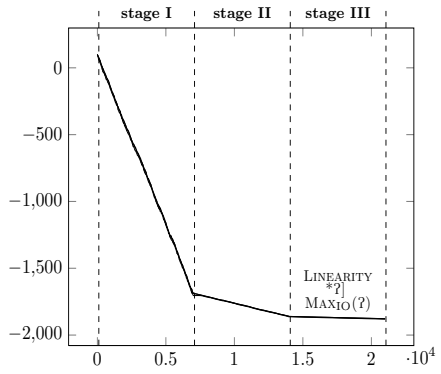
4.1 Why EDCD and the CEDRA fail

The case of EDCD and the CEDRA is completely different. As we will see in this section, this is crucially due to the fact that these algorithms do not promote by the same amount they demote, contrary to the GLA. We report in Table 8a the final ranking values learned by EDCD in a simulation on the core Ilokano metathesis test case described in Table 6c. These ranking values are all roughly the same. This means that the algorithm has failed to learn that the constraint $\text{MAX}_{\text{IO}}(ʔ)$ needs to be ranked at a safe distance underneath both constraints LINEARITY and *ʔ]. As shown by the ERC matrix in Table 6c, the latter ranking condition is needed in order to account for the fact that the underlying form /paʔlak/ is deterministically mapped to [pa.lak]. The fact that EDCD fails to learn this ranking condition explains the

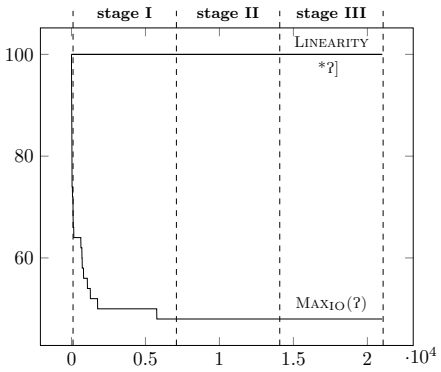
(a)

Constraint	final RV
LINEARITY	-1877.38
*ʔ]	-1877.72
MAX$_{IO}$(ʔ)	-1880.34

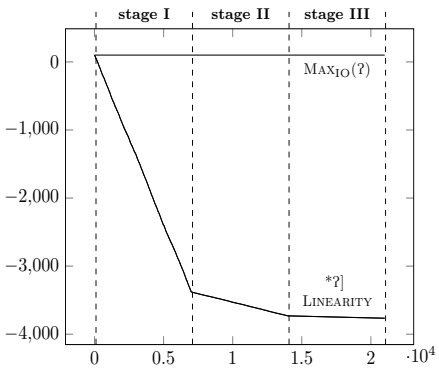
(b)



(c)



(d)



(e)

ERC	#U	LU	LI
ERC1 = (/paʔlak/, [pa.lak], [pal.ʔak])	1282	7791	20998
ERC 2 = (/paʔlak/, [pa.lak], [paʔ.lak])	1282	7786	20991
ERC 3 = (/taʔo-en/, [taw.ʔen], [taʔ.wen])	2614	7784	20983
ERC 4 = (/taʔo-en/, [taʔ.wen], [taw.ʔen])	2613	7790	20996

Table 8: Behavior of EDCD on the core Ilokano metathesis test case in Table 6c

failure of EDCD on this underlying form, diagnosed in Table 5. The corresponding ranking dynamics is plotted in Table 8b. All three constraints drop in a free fall, without EDCD being able to enforce any separation between LINEARITY and *ʔ] on the one hand and MAX$_{IO}$(ʔ) on the other hand. We want to understand in detail this learning dynamics and thus explain why EDCD fails on this test case.

Again, it is useful to start by investigating the learning dynamics of EDCD when trained on the two underlying forms separately. Let me start with the case where EDCD is trained on the deterministic underlying form /paʔlak/ alone. Since EDCD performs no constraint promotion, the two corresponding ERCs 1 and 2 do not re-rank the two winner-preferring constraints LINEARITY and *ʔ]. But they both demote the loser-preferring constraint MAX$_{IO}$(ʔ). After a few updates, the latter loser-preferring constraint has dropped underneath the two former winner-preferring constraints at a distance large enough that the additive error cannot swap the constraints. This is illustrated by the ranking dynamics plotted in Table 8c, which is indeed the ranking dynamics obtained when EDCD is trained only on the

underlying form /paʔlak/ corresponding to the two ERCs 1 and 2. Overall, EDCD's ranking dynamics in Table 8c is not substantially different from the GLA's ranking dynamics in Table 7c: when trained on the underlying form /paʔlak/ alone, the two algorithms behave roughly in the same way.

The crucial difference between the GLA and EDCD comes up when the two algorithms are trained on the stochastic underlying form /taʔo-en/. The two corresponding ERCs 3 and 4 completely disagree with each other: LINEARITY and *ʔ] are winner-preferring in one of the two ERCs and loser-preferring in the other. Crucially, EDCD performs constraint demotion but no constraint promotion. This means that when trained on these two ERCs 3 and 4, EDCD will force the two constraints LINEARITY and *ʔ] in a free fall, dragging them away from their initial ranking value indefinitely, for as long as learning continues. This is illustrated by the ranking dynamics plotted in Table 8d, which is indeed the ranking dynamics obtained when EDCD is trained only on the underlying form /taʔo-en/ corresponding to the two ERCs 3 and 4. The difference with the ranking dynamics in Table 7d obtained when the GLA is trained on this underlying form is striking. Since the GLA promotes and demotes by the same amount, it keeps the two constraints LINEARITY and *ʔ] oscillating up and down without effectively displacing them. Since EDCD instead performs constraint demotion but no promotion, it forces these two constraints LINEARITY and *ʔ] into a free fall.

It is now easy to put the pieces together. In the case of EDCD, the two ERCs 3 and 4 corresponding to the underlying form /taʔo-en/ force the two constraints LINEARITY and *ʔ] into a free fall. Since the two underlying forms are sampled with the same frequency, these two constraints fall too fast in order for the other two ERCs 1 and 2 corresponding to the other underlying form /paʔlak/ to be able to slide the constraint $\text{MAX}_{\text{IO}}(\text{ʔ})$ underneath them. Indeed in the case of the GLA, ERCs 1 and 2 trigger few updates and only in the initial segment of the run, as reported in Table 7e. The situation is very different in the case of EDCD, as reported in Table 8e: all ERCs trigger a very high number of updates and they all keep triggering updates until the very end of the run.

The case of the CEDRA is completely analogous to that of EDCD just considered. Instead of performing *no* constraint promotion as EDCD, the CEDRA performs *little* promotion: crucially it promotes less than it demotes. Hence, we expect exactly the same free-fall ranking dynamics, only slower. And this is what we get, as illustrated in Table 9b. As the demotion amount is larger than the promotion amount, the fight between *ʔ] and LINEARITY triggered by ERCs 3 and 4 again forces them in a free fall, only a slower fall than in the case of EDCD. And again ERCs 1 and 2 hardly cope with keeping $\text{MAX}_{\text{IO}}(\text{ʔ})$ underneath *ʔ] and LINEARITY, as reveled by the final ranking values reported in Table 9a. Again, all ERCs remain active until the end of the run, as revealed by Table 9c.

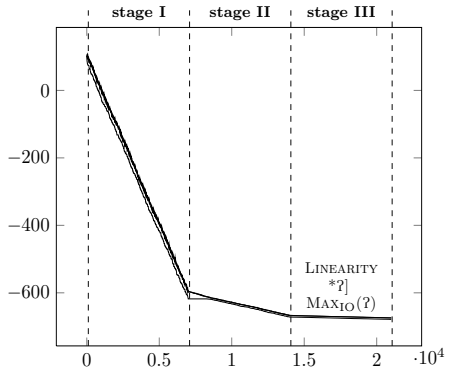
5 Conclusion

BH report that the GLA implementation of error-driven learning within stochastic OT succeeds at learning variation on three complex, naturalistic test cases. Minimal variants of the GLA (namely, EDCD, the minGLA, and the CEDRA; see Table 2) instead fail on these test cases. As suggested at the beginning of the paper, we

(a)

Constraint	final RV
LINEARITY	-674.37
*ʔ]	-674.39
MAX_{IO}(ʔ)	-678.68

(b)



(c)

ERC	#U	LU	LI
ERC1 = (/paʔlak/, [pa.lak], [pal.ʔak])	496	6166	20900
ERC 2 = (/paʔlak/, [pa.lak], [paʔ.lak])	476	6190	20987
ERC 3 = (/taʔo-en/, [taw.ʔen], [taʔ.wen])	2621	6196	21000
ERC 4 = (/taʔo-en/, [taʔ.wen], [taw.ʔen])	2603	6194	20996

Table 9: Behavior of the CEDRA on the core Ilokano metathesis test case

interpret these simulation results as showing that BH's test cases have some special structure which the GLA is crucially able to exploit, while variants of the GLA are not. What is this special structure? In MS, we address this question through a detailed analysis of BH's simulations. As a preview, in this paper we have looked at one of BH's test cases, namely the Ilokano metathesis test case. Our analysis shows that the crucial property of this test case is the fact that variation boils down to a pair of ERCs with the shape in (1).

(1)

$$\begin{array}{lc} & \begin{array}{ccccc} & C_h & & C_k & \end{array} \\ \begin{array}{l}(/x/,[y],[\cancel{z}]) \\ (/x/,[z],[\cancel{y}])\end{array} & \left[\begin{array}{ccccc} \ldots & \text{W} & \ldots & \text{L} & \ldots \\ \ldots & \text{L} & \ldots & \text{W} & \ldots \end{array}\right] \end{array}$$

The winner and the loser are swapped in the two ERCs. Thus, a constraint is winner-preferring relative to one of the two ERCs if and only if it is loser-preferring relative to the other ERC. In other words, these two ERCs are one the negation of the other (one has a W corresponding to a certain constraint if and only if the other has an L).

The GLA promotes and demotes by the same amount. Hence, the mutually contradicting stochastic ERCs (1) keep "fighting each other" (namely force the active constraints to oscillate up and down), without effectively "getting anything done" (namely without effectively displacing their active constraints and thus without contributing to the ranking dynamics). Since variation in the Ilokano metathesis test case reduces to a pair of these mutually contradicting stochastic ERCs (1) and since these contradicting ERCs get nothing done in the case of the GLA, then the GLA is in some sense "insensitive" to variation: if the stochastic ERCs are dropped, the oscillations are of course lost, but the shape of the ranking dynamics is unaffected.

The case of EDCD and the CEDRA is completely different. As these algorithms perform less constraint promotion than demotion (EDCD actually performs

no promotion at all), then these stochastic contradicting ERCs (1) get "a lot done", namely they manage to force their active constraints into a free fall. And the algorithms thus fail at sliding any constraint underneath the constraints forced into a free fall. These stochastic ERCs thus have a drastic effect on the ranking dynamics in the case of EDCD and the CEDRA.

In MS, we develop an analogous account for the success of the GLA on BH's Finnish genitive test case and the English dark [l] test case. Also in these test cases, stochasticity boils down to pairs of mutually contradicting stochastic ERCs corresponding to the same pair of candidates $[y]$ and $[z]$. Again, the GLA is not substantially affected by these stochastic ERCs, because it promotes and demotes by the same amount. While in the case of EDCD and the CEDRA we get a free falling ranking dynamics, which washes away any effort of the algorithm at establishing any ranking condition.

Now that we have been able to pinpoint at the structure which allows the GLA to succeed on BH's test case, we are in a better position to investigate whether that structure might indeed be characteristic of patterns of grammatically-induced variation in Natural Language. Here is a specific strategy to probe into this question, that we are currently investigating. Part of what is crucially helping the GLA in BH's test cases is the fact that variation is always between **two** candidates $[y]$ and $[z]$ for a given underlying form, leading to pairs of ERCs such as (1), which completely mutually contradict each other because of the fact that the winner and the loser are swapped. What about cases of variation among **three** candidates for a given underlying form? Do these cases lead to stochastic ERCs with a different structure from the one in (1)? How does the GLA cope with that structure?

References

Anttila, Arto. 1997. Deriving variation from grammar: A study of Finnish genitives. In *Variation, change and phonological theory*, ed. by Frans Hinskens, Roeland van Hout, & Leo Wetzels, 35–68. Amsterdam: John Benjamins. Rutgers Optimality Archive ROA-63, http://ruccs.rutgers.edu/roa.html.

Boersma, Paul. *Functional Phonology*. University of Amsterdam, The Netherlands dissertation. The Hague: Holland Academic Graphics.

——, & Bruce Hayes. 2001. Empirical tests for the Gradual Learning Algorithm. *Linguistic Inquiry* 32.45–86.

Magri, Giorgio. 2012. Convergence of error-driven ranking algorithms. *Phonology* 29.213–269.

——, & Benjamin Storme, in prep. A closer look at Boersma and Hayes' (2001) simulations.

Prince, Alan, 2002. Entailed ranking arguments. Ms., Rutgers University, New Brunswick, NJ. Rutgers Optimality Archive, ROA 500. Available at http://www.roa.rutgers.edu.

Tesar, Bruce, & Paul Smolensky. 1998. Learnability in Optimality Theory. *Linguistic Inquiry* 29.229–268.