Learning Anttila Grammars

Floris van Vugt
florisvanvugt@ucla.edu

March 14, 2010

1 Introduction

In optimality theory, the underlying form is taken as input and turned into an infinite variety of possible output forms by a function generally called Gen. To decide which of these candidates is actually uttered, we compare how well the candidates satisfy a number of constraints. The constraints are ordered and given two candidates, one is more harmonic than the other if it has fewer violations on some constraint than the other, and equal violations on all higher ranked constraints. Finally, given a particular input, the winning candidate is the one that is more harmonic than all the others.

If we now assume that no two candidates have equal violations on all constraints (i.e. for any two candidates there will be a constraint that distinguishes them by assigning a different number of violations) then it follows that there will be one and only one such winner.

But this does not always reflect phonological reality. Some words can be pronounced in different ways without there being any factors in the context predicting which. In such a case, we say that these multiple outputs occur in free variation.

Several proposals have been made as to how optimality theory (OT) could account for free variation without assuming that two candidates have exactly the same violations on all constraints. The proposal that will be discussed in this paper is due to Arto Anttila and henceforth called *Anttila grammars*Anttila (1997).

The idea is as follows. Constraints are not totally ordered, but they come in groups called *strata*, which are totally ordered with respect to each other.

The constraints within each stratum, however, do not have any ordering at all. The central hypothesis is then that each of the possible rankings of the constraints in a stratum are *equiprobable*.

The combination of all orderings of constraints within each stratum of the grammar yields a list of possible total orderings of the all constraints. By hypothesis these are equiprobable and therefore we predict that the frequency of occurrence of an output form is the fraction of these orderings under which it wins.

1.1 An example Anttila grammar

It may be helpful to investigate a representative example of an Anttila grammar. Table 1 shows such an example with seven constraints that come in five strata. The first stratum contains only constraint A, the second contains B and C, the third only D, the fourth E and F and the last contains only G.

Hence, there are two possible rankings in the second stratum: BC and CB. Similarly there are two possible rankings in the fourth stratum: EF and FE. Since all other strata contain only one member, there is only one ordering in them. Combining the orderings of the strata this yields a total of $1 \times 2 \times 1 \times 2 \times 1 = 4$ possible orderings.

These orderings are listed in table 2. We find that for input 1 and 2 the candidates a and b each win in half of the orderings.

1.2 Learnability

The obvious appeal of Anttila grammars is that they provide an account of free variation without introducing any further conceptual apparatus: the assumption is that the inside of a stratum is simply underspecified and hence the speaker randomly puts the constraints into an order and utters the winning candidate.

But one wonders whether these grammars are learnable. Part of the appeal of optimality theory in its original formulation is that if the constraints and underlying forms are known, the grammar is learnable by a straight–forward algorithm called Constraint DemotionTesar & Smolensky (2000). In other words, given n constraints, the learner does not have to try out all of the astronomically many $n!$ rankings of constraints in order to learn the grammar.

Table 1: The test grammar we submit to our algorithm. The dashed lines separate constraints that are in the same stratum; the solid lines separate the strata. On the basis of the constraint violations we calculated the frequencies for each of the candidates.

input	cand.	freq.	A	B	C	D	E	F	G
1	a	.5			*				
	b	.5		*					
	c	0	*						
2	a	1				*			
	b	0		*					
3	a	1				*			
	b	0			*				
4	a	.5						*	
	b	.5					*		
	c	0				*			
5	a	1							*
	b	0					*		
6	a	1							*
	b	0						*	

Table 2: Possible orderings of the constraints in table 1 and the winners for each input candidate.

	Winners					
Ordering	1	2	3	4	5	6
A.BC.D.EF.G	a	a	a	a	a	a
A.CB.D.EF.G	b	a	a	a	a	a
A.BC.D.FE.G	a	a	a	b	a	a
A.CB.D.EF.G	b	a	a	b	a	a

Table 3: A comparison of the number of classical OT grammars and Anttila grammars given n constraints

n	1	2	5	10	13
Classical OT grammars	1	2	120	3628800	6227020800
Anttila grammars	1	3	541	102247563	526858348381

How is the case of learning an Anttila grammar? Clearly introducing strata we have dramatically increased the number of possible grammars. Given n constraints, there are $\sum_{1 \leq i \leq n} S(i, n) \cdot i!$ where S is the second order Stirling number. Table 3 lists some concrete numbers for comparison.

This means that it is even more unrealistic for the learner to generate all such divisions into strata and check whether they generate the correct frequencies, than it would be to calculate all classical OT rankings. There is, however, no obvious more efficient algorithm for learning the Anttila grammars.

In this paper, I will investigate a candidate algorithm for learning Anttila grammars that is based on the Constraint Demotion procedure and assess its effectiveness.

2 Parallel Constraint Demotion

The algorithm takes as input a number of input forms and their associated candidates, as well as a number of constraints. Each of the candidates is associated with an observed frequency of occurrence. This frequency of occurrence is assumed to have been detected by the learner. We further assume to know the violations of all candidates on each of the constraints. The algorithm will output a division into strata, in other words an Anttila grammar. This grammar will be built gradually and referred to as *target grammar*.

1. The target grammar is initially empty: it has no constraints and no strata.

2. We initialise our *focus stratum* to the set of all constraints. Their ordering in this stratum is irrelevant. This focus stratum is our working space and we will be moving constraints from it into our target grammar.

3. We initially mark all candidates as *unaccounted for*. The purpose of this will become clear in what follows.

4. We single out, for each input, the pairs of its candidates (w, l) such that w has a nonzero frequency of occurrence and l has a zero frequency of occurrence. Of these, we select only those pairs where both elements are unaccounted for and remove the others from further analysis.

5. For each constraint in the focus stratum we determine whether that constraint prefers the occurring (w) or non–occurring (l) candidate of each pair. In the former case, i.e. when the constraint assigns strictly less violations to the occurring candidate, we mark the constraint as a *winner–preferer* (relative to that pair of candidates). If the constraint assigns strictly more violations to the occuring candidate, we mark it as a *loser–preferer* (again, relative to the pair of candidates). If neither is the case, it is a *neutral* constraint.

6. We select the constraints that are not loser–preferer for any of the pairs, i.e. they must always be neutral or winner–prefer. This set will form a stratum that we add to our target grammar below all strata that are already in there (if any).

7. The remaining constraints (i.e. those that for at least one pair prefer the loser) form the focus stratum for the next cycle.

8. We mark as *accounted for* those candidates l that have zero frequency and for which there is an occurring candidate w such that one of the constraints in our focus set prefers w. The reason we can mark this candidate as accounted for is that it will not be able to win due to the occurrence of that constraint in that place in the grammar. This is the counterpart of "culling" in regular Constraint Demotion.

9. We repeat the steps 4 to 8 until one of the following obtains:

    - *All of the constraints in the focus set are demoted.* That is, we are entering a loop in which nothing further will happen. We finish our algorithm by placing the current focus set as a whole as the lowest stratum in our grammar.
    - *The focus set is empty.* In this case we have placed all constraints in a stratum of the grammar, hence we are done.

Table 4: Cycle 1 (**left**) and 2 (**right**) of the Parallel Constraint Demotion algorithm.

	Constraints						
pair	A	B	C	D	E	F	G
(1a, 1c)	w	-	l	-	-	-	-
(1b, 1c)	w	l	-	-	-	-	-
(2a, 2b)	-	w	-	l	-	-	-
(3a, 3b)	-	-	w	l	-	-	-
(4a, 4c)	-	-	-	w	-	l	-
(4b, 4c)	-	-	-	w	l	-	-
(5a, 5b)	-	-	-	-	w	-	l
(6a, 6b)	-	-	-	-	-	w	l

	Constraints						
pair	A	B	C	D	E	F	G
(2a, 2b)	-	w	-	l	-	-	-
(3a, 3b)	-	-	w	l	-	-	-
(4a, 4c)	-	-	-	w	-	l	-
(4b, 4c)	-	-	-	w	l	-	-
(5a, 5b)	-	-	-	-	w	-	l
(6a, 6b)	-	-	-	-	-	w	l

2.1 An example

We can now investigate how this procedure works for the grammar given in table 1. We mark the frequencies for each candidate and remove the division into strata. In the ideal case our algorithm will find back the division into strata.

- **initialisation**. Our focus set is {A,B,C,D,E,F,G}. All candidates are unaccounted for.
- **cycle 1**. We identify the occurring/non–occurring pairs and decide which of the two each constraint prefers, which is illustrated in table 4. We see that only constraints A never prefers a loser and hence we demote all other constraints. This means our target grammar at the end of the first cycle has one stratum which contains only A. Since A prefers the winner in the pairs (1a,1c) and (1b,1c) we mark 1c as accounted for.
- **cycle 2**. Our focus set is {B,C,D,E,F,G} (the demoted constraints from the previous step). Since we have marked (1c) as accounted for, constraints B and C now never prefer a loser. Hence we add a stratum with B and C to our grammar. Futher, candidates 2b and 3b are marked as accounted for.
- **cycle 3**. Our focus set is {D,E,F,G}. Since we eliminated the two candidates that D wrongly preferred, it now prefers only winners. All

other candidates still prefer at least one loser and thus are demoted. We add a stratum with only D to our grammar. We mark 4c as accounted for.

- **cycle 4**. From {E,F,G} both E and F now only prefer winners (the only pairs that remain unaccounted for are (5a,5b) and (6a,6b)). We add a stratum with E and F to the bottom of the grammar.

- **cycle 5**. The only remaining constraint is G. All candidates are accounted for, so G never prefers a loser, hence we demote it. That is, we have demoted all constraints in our focus set, hence we terminate the algorithm and put G in the last stratum of the target grammar.

It has become clear our algorithm has correctly found back the division into strata that we started out with, that is, it has succesfully learned this grammar.

2.2 How real are zero frequencies of occurrence?

This algorithm has the curious property of calling on stage the candidates with zero frequency of occurrence as the counterparts to the "losers" in regular Constraint Demotion. In our context, these are the candidates that lose under any of the possible rankings. But can we assume that a language learner can identify which are these forms that never occur?

Firstly of course there is the general inductive problem that there is no series of observations that will establish that a form does not occur. Never having seen an atom being split in my entire life, I nevertheless do not infer that it is impossible. This problem is also faced by the Constraint Demotion algorithm for original OT. Under the assumption that there is no free variation, however, the occurrence of a particular candidate will allow the learner to infer that all the other candidates are less harmonic and proceed from this assumption with the sorting of the constraints.

But when forms do occur in free variation, how can the learner decide between forms that occur at very low frequencies and forms that do not occur at all? There seems to be no straightforward answer to this objection. This might be taken to reflect a shortcoming of the Anttila grammars that predict a sharp, binary distinction between forms that occur and forms that do not. This is different in, e.g. MAXENT grammars, where all candidates are assigned nonzero (but possible very close to zero) frequencies.

Table 5: A set of candidates and frequencies that cannot obtain in an Antilla grammar with the given constraints.

input	cand.	freq.	A	B
1	a	.4	*	
	b	.6		*

In practice, a learner that uses the algorithm proposed here might simply use a variable frequency threshold beneath which forms are treated as non–occurring.[1]

2.3 Assessing effectiveness of the algorithm

One can now ask how one should assess the effectiveness of the algorithm. One criterion could be to assess whether the algorithm, given any set of constraints and candidates with their associated frequencies, can learn a division into strata that yields precisely these frequencies. However, this criterion is too strict, since one can well imagine cases in which there is simply no division into strata which will yield the observed frequencies. An example is given in table 5. The reason is that if one of the constraints is in a separate stratum, it will cause the candidate the violates it to occur with zero frequency. Furthermore, if all constraints are in the same stratum, the frequencies should be equal. So our algorithm cannot be blamed for not find a division into strata that yields the correct frequencies, for there simply is none.

Since the interest of this paper is to assess pure learnability of Anttila grammars and not their descriptive accuracy, we will discard this criticism as a criticism of Anttila grammars as a whole and not of our learning algorithm. The question of effectiveness of the algorithm will thus be: which languages that can be modelled by Anttila grammars can be learned by the algorithm?

[1]That is, he or she might start out considering all forms that are heard in less than 1% of the cases as non–occuring. If this fails to yield a correct grammar, he or she could adapt the threshold to become more or less strict.

Table 6: An example that shows that the proposed algorithm is not able to implement fine–tuned strata divisions necessary to obtain the correct frequencies (though an Anttila grammar can capture the frequencies with the indicated division into strata)

input	cand.	freq.	A	B	C	D
1	a	1/3	*			*
	b	2/3		*	*	

2.4 The exact value of nonzero frequencies is ignored

We can further observe that our algorithm has treated all nonzero frequencies as being equal. In other words, all that our procedure takes as input is whether a frequency is zero or nonzero (i.e. whether it will be classified as w or l), but the exact value of the frequency is not used in the computation. So, for better or worse, a candidate with .6 frequency of occurrence is treated the same as a candidate with .05 frequency.

This has one immediate consequence: the algorithm will not be able to make divisions into strata when these are not motivated by the need to rule out a particular candidate that we know always loses, but rather by the desire to fine–tune the relative frequencies of candidates that sometimes win.

The following example will illustrate this point. In table 6 there are no constraints that prefer a loser (for the simple reason that there is no candidate with zero frequency of occurrence). Therefore our algorithm will return all the constraints in one stratum. But taking all possible permutations of constraints in that stratum, candidates 1a and 1b win equally often. In other words, the frequencies calculated according to the output grammar of our algorithm do not match those that we started out to model. There is, however, an alternative division into strata which yields the correct output. The grammar which puts A,B and C in the topmost stratum and D in a stratum below that will generate precisely the observed frequencies.

In sum, there is an example of a language that can be captured by an Antilla grammar, but which is not learnable by our algorithm. Perhaps this is not as serious an objection as it seems. Our algorithm returns in this caes a division into strata that is too "coarse" in the sense that it fails to make a subdivision into strata based on the frequencies of the occurring candidates.

So one could imagine that after the application of this algorithm an auxiliary algorithm would be executed to determine if any subdivisions of the strata of the resulting grammar are necessary to fine–tune the predicted frequencies.

Therefore our question of the effectiveness of the algorithm will be further narrowed: we are now curious if our algorithm might return a division into strata that is not just too coarse, but plainly wrong, making divisions where there should be none. Such cases would be worse, since a hypothetical fine–tuning mechanism could not repair them without redoing the entire problem.

2.5 A quantitative approach to assess effectiveness

2.5.1 Explanation of the simulation

Now that our question is made precise we turn to a quantitative method to assess the effectiveness of the learning algorithm.

Using a Java program, random Anttila grammars were generated in the following fashion (henceforth these are called the *original* grammars). It generated 2 to 4 strata containing 1 to 4 constraints each. Next, between 1 and 5 input forms were created with each 2 to 5 associated candidates. Finally, each candidate was assigned between 0 and 2 violations on each constraint. Since these fully specify an Anttila grammar, one can then calculate the predicted frequencies of occurrence for each candidate by investigating all possible permutations of the constraints in the strata.

These frequencies, the candidates and constraints (but crucially of course not the division into strata) was then fed into the algorithm which proceeded to output a division into strata. We refer to this output as the *derived* grammar. We then generate all possible orderings for these strata and calculate the corresponding winners, thus counting the predicted frequencies for each candidate. A grammar is then considered to be succesfully learnt if the predicted frequencies of the derived grammar equal those of the original grammar.

Notice that this does not mean that the division into strata is the same. For example, in the grammar in table 6 we could replace A with D and obtain exactly the same frequencies. So what matters is not whether the algorithm outputs the same strata but whether these strata result in the same frequencies as the original grammar.

Table 7: An example that shows that the proposed algorithm fails to generate the correct frequencies, and even a further subdivision of the strata will not remedy this (**left**). For each occurring/non–occurring pair the candidate which the constraint prefers (**right**). The output of the algorithm (**bottom**), yielding incorrect predicted frequencies.

input	cand.	freq.	A	B	C
1	a	.5	**		
	b	.5		*	
	c	0		*	*

	Constraints		
pair	A	B	C
(1a, 1c)	l	w	w
(1b, 1c)	-	-	w

input	cand.	freq.	B	C	A
1	a	1			**
	b	0	*		
	c	0	*	*	

2.5.2 Classification results of the simulation

The algorithm was tested on 100,000 such randomly generated grammars. It correctly learned 57,495 of these (57.5%). In the grammars that were not correctly learned, a further 8035 (8.0%) simply yielded too coarse a division into strata (cf. discussion in section 2.4) which means they might be solvable by an extension of the algorithm. This implies that the remaining 34.5% are the more serious failures that will be investigated in what follows.

2.5.3 Investigation of incorrectly learned grammars

What grammars are not correctly learned by this algorithm and what is it about them that causes the procedure to make the wrong stratal divisions?

The grammar in table 7 is one of the simplest and most intriguing examples of grammars that were incorrectly learned and whose output strata are not just too coarse. The algorithm detects that constraint A is the only that ever prefers a non–occurring candidate and thus demotes it to a lower stratum, and then the procedure ends. The output grammar is shown in the table and predicts that candidate 1a is the only winner. In other words, the algorithm has reduced the free variation grammar to a deterministic one with one winner.

Table 8: Overview of the action of each stratum in the original grammar of table 7.

	Winners after each stratum	
ordering	Stratum (A,B)	Stratum (C)
AB.C	{b,c}	{b}
BA.C	{a}	{a}

At this point it is unclear how the algorithm could be amended to eliminate this incorrect classification. The procedure, it seems, fails to see that though constraint A by itself prefers a loser when comparing candidate 1a to 1c, it nevertheless is necessary to pair with constraint B in a stratum to allow both 1a and 1b to win. Then constraint C can instead be called in afterwards to eliminate the illegal 1c (or it can even occur before constraints A and B, as long as it is in a stratum of its own).

This seems to then point at an interesting layer of complexity that Anttila grammars have added to regular Optimality Theory.

Table 8 shows that in one of the possible orderings of the original grammar, the first stratum (containing only A and B) lets through 1b and 1c, that is, too many candidates. The second stratum remedies this by selecting only 1b. This is to say, the fact that constraint A prefers a candidate, 1c, which always loses is not problematic since constraint C in a separate stratum is sufficient to rule it out.

3 Conclusion

We have investigated a possible learning algorithm for Anttila grammars which is a generalisation of the Constraint Demotion procedure in regular optimality theory. The learning algorithm proceeds by demoting constraints that prefer candidates that never win.

The algorithm, though surprisingly effective in our quantitative simulation, proves to miss the essential interactions that Anttila grammars allow between the constraints. As such, it is a good heuristic a language learner might use but by no means an effective procedure that is guaranteed to find the suitable Anttila grammar for a set of observed data.

References

Anttila, Arto. 1997. *Variation in finnish phonology and morphology.* Ph.D. thesis, Stanford University.

Tesar, Bruce, & Smolensky, Paul. 2000. *Learnability in optimality theory.* Cambridge, Massachusetts: The MIT Press.