Computational Phonology, class 5: Optimality Theory

Adam Albright

CreteLing 2022 — July 2022

Learning with constraints

The elements of most constraint-based approaches

/UR/	C1	C2	C3
☞ a. cand1			*
b. cand2		*	
c. cand3			**

- Grammar assigns probability distribution over outputs (candidate SR's), conditioned on an input form (UR)
  - Discriminative model: chooses among a given set of responses
  - Prince and Smolensky (1993/2004): separate Gen function

The elements of most constraint-based approaches

/UR/	C1	C2	C3
☞ a. cand1			*
b. cand2		*	
c. cand3			**

- Con: Constraints assess violations of competing candidates
    - Markedness: properties of outputs
    - Faithfulness: relation between output and input

The elements of most constraint-based approaches

/UR/	C_1	C_2	C_3
☞ a. cand1			*
b. cand2		*	
c. cand3			**

- Eval: Candidates compete based on harmony, which is a function of their violation profiles
    - Sum over all violations (HG, MaxEnt), or focus on top violations (OT)
    - Categorical (OT, HG), or gradient (MaxEnt, Stochastic OT, Noisy HG)

Learning in constraint-based frameworks

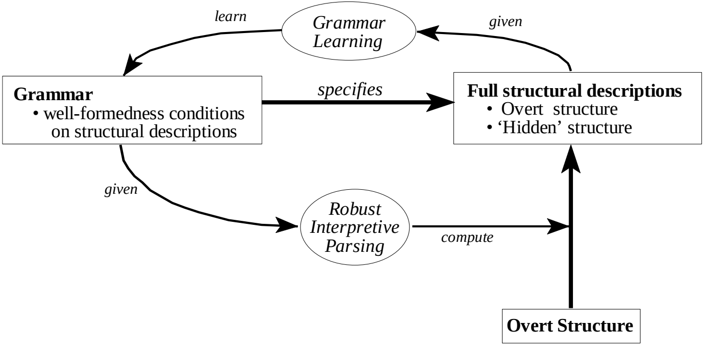
- Inputs (UR's): must be learned
- Candidates: assume invariant Gen (not learned)
- Con
  - All constraints universal? (Prince and Smolensky 1993/2004)
  - Markedness constraints learned? (Hayes and Wilson, 2008)
  - Markedness and faithfulness learned? (Rasin and Katzir, 2015, 2016)
- Eval
  - Violations depend on constraints (though see Doyle et al., 2014 for an approach to learning violation profiles directly)
  - Ranking/weighting must be learned

Learning in constraint-based frameworks

- Inputs (UR's): must be learned
- Candidates: assume invariant Gen (not learned)
- Con
    - All constraints universal? (Prince and Smolensky 1993/2004)
    - Markedness constraints learned? (Hayes and Wilson, 2008)
    - Markedness and faithfulness learned? (Rasin and Katzir, 2015, 2016)
- Eval
    - Violations depend on constraints (though see Doyle et al., 2014 for an approach to learning violation profiles directly)
    - Ranking/weighting must be learned

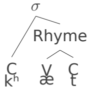
The learning scenario

Tesar and Smolensky (1996), p. 14


learn
Grammar Learning
given
Grammar
• well-formedness conditions on structural descriptions
specifies
Full structural descriptions
• Overt structure
• 'Hidden' structure
given
Robust Interpretive Parsing
compute
Overt Structure

The learning scenario

- Learner receives: [kʰæt]
  - We’ve already said we’ll take for granted that the learner has learned how to analyze continuous acoustic signal into a sequence of symbols (categories)
- Must use the current grammar to parse (hidden structure): underlying and surface form—e.g.,

/kʰæt/	C_1	C_2	C_3
☞ a. 	*		**
b. (Gen provides competing candidates)		*	**

- Based on violations, must adjust grammar to ensure that given form wins
  - Add constraints? (if learned)
  - Change ranking?

Decomposing the problem

- Really hard part:
  - Inferring hidden structure, on basis of overt evidence
  - We'll ignore hidden output structure (syllabification, etc.)
  - UR's: IN=OUT is optimal
- Somewhat hard part:
  - Given those UR's and candidates, find constraints that distinguish them
  - We'll start by assuming Con is given
- The easier part:
  - Given UR's and candidates, find a consistent ranking

Consistent rankings

- You might have thought that the job of the learner is to find the grammar that generated the observed surface forms
  - But this is too strict!

Consistent rankings

- You might have thought that the job of the learner is to find the grammar that generated the observed surface forms
    - But this is too strict!
- Job of the learner: *find a grammar* that generates the observed surface forms

Generating the observed data: intuitively

- For all observed strings, it’s possible to find an input for which that string is the optimal (winning) candidate
  - Make sure [kʰæt] is a possible winner, e.g., for input /kʰæt/
- Often, we care about more
  - For all unobserved strings, no input would have that string as the winning candidate
  - Make sure unobserved [ktʰæ] never wins, for any input (/ktʰæ/ favors something else, like [tʰæ])

/kʰæt/	*[TT	Max(C)	*Coda
☞ a. kʰæt			*
b. kʰæ		*!	

/ktʰæ/	*[TT	Max(C)	*Coda
a. ktʰæ	*!		
☞ b. tʰæ		*	

(How would a model that did this perfectly behave? What are the predictions?)

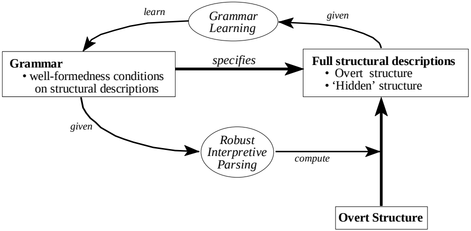
Unobserved strings

- In actuality, adult speakers do accept some unobserved strings
  - Chomsky and Halle: [blɪk]
  - We do want the grammar to be able parse and output these
- Other unobserved strings are rejected
  - Chomsky and Halle: [bnɪk]
  - The grammar should not be able to select these as outputs
- Accidental gaps ([blɪk]) vs. “significant” gaps ([bnɪk])

Achieving this balance

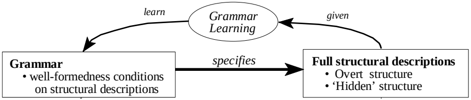
- A grammar that accepts all and only the forms that it has observed is *overfitted*
  - Can’t generalize to unseen forms
- An important tool in avoiding overfitting: coarse constraints
  - No constraint *blɪk
  - Any ranking that could accept [blu] and [tɪk] can also accept [blɪk]
- Since we’re starting by assuming universal Con, we’ll assume that the constraints are appropriately general

More narrowly: the ranking problem


learn
Grammar Learning
given
Grammar
• well-formedness conditions on structural descriptions
specifies
Full structural descriptions
• Overt structure
• 'Hidden' structure
given
Robust Interpretive Parsing
compute
Overt Structure

- Discussion on previous few slides characterized the learning problem of receiving data, and making sure that ranking and UR's are consistent with them tableaus

More narrowly: the ranking problem


learn
Grammar Learning
given
Grammar
• well-formedness conditions on structural descriptions
specifies
Full structural descriptions
• Overt structure
• 'Hidden' structure

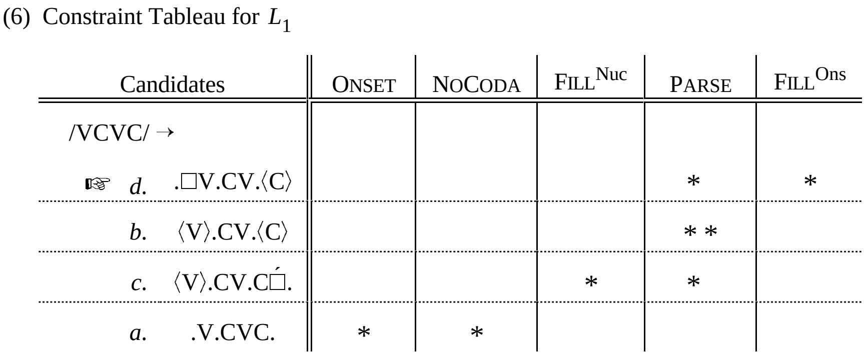
- Simplifying further:
    - For all given/inferred pairs of UR, SR...
    - Find a ranking that favors the observed SR over all competing candidate SR's
- "Finding a consistent ranking" = finding the ranking that gets the right winner for the given tableaus

Positive vs. negative evidence

- This formulation of the ranking problem does not intrinsically specify how UR's are determined
- One possible data set: positive + negative evidence
  - (/kʰæt/, [kʰæt]), (/blu/, [blu]), (/blɪk/, [blɪk]), (/bnɪk/, [bənɪk])
  - This data set contains non-English blɪk and the information that it should surface faithfulness, and non-English bnɪk and the information that it should not surface faithfully
- Another possible data set: positive evidence only
  - (/kʰæt/, [kʰæt]), (/blu/, [blu])
  - Harder but more realistic challenge: generalize to /blɪk/ and /bnɪk/ in the way that English speakers do
- This distinction has important implications for modeling humans, but the ranking problem is the same either way!

Learning in OT

An example



(6) Constraint Tableau for L_1

Candidates		Onset	NoCoda	Fill$^{\text{Nuc}}$	Parse	Fill$^{\text{Ons}}$
/VCVC/ →						
☞ *d.*	.□V.CV.⟨C⟩				*	*
b.	⟨V⟩.CV.⟨C⟩				* *	
c.	⟨V⟩.CV.C□́.			*	*	
a.	.V.CVC.	*	*			

- Assume we're given inputs (with all hidden structure), actual output, and losing candidates

The procedure: an overview

1. Construct *mark-data pairs*
   - For each loser/winner pair, collect all violations
   - If both violate same constraint $\mathbb{C}$ an equal number of times, these marks cancel each other out
   - Identify $\mathbb{C}$ that assess uncancelled marks
2. Start with all $\mathbb{C}$ in a single stratum
3. Look for $\mathbb{C}$ that assign uncancelled marks to winners (that is, all constraints with L). Demote any such $\mathbb{C}$, unless it is already dominated by another constraint $\mathbb{C}'$ that has uncancelled *loser* marks (that is, a higher W)
4. Continue, creating subsequent strata, unless there are no uncancelled winner marks without higher-ranked uncancelled loser marks

Applying RCD

Step 1: assess violations for MDP’s

- Shown here in comparative form (Prince 2000, 2002)
- Faithfulness constraints updated (Correspondence Theory)

/VCVC/ → .□V.CV.⟨C⟩	Ons	*Coda	Dep(V)	Max	Dep(C)
d. ∼ b. ⟨V⟩.CV.⟨C⟩				W	L
d. ∼ c. ⟨V⟩.CV.C□.			W		L
d. ∼ a. .V.CVC.	W	W		L	L

Step 2: apply RCD

Another example

For practice:

Input	Intended	*si	*s	*ʃ	F(ant)
/sa/	☞ sa		*		
	ʃa			*	*
/ʃa/	ʃa			*	
	☞ sa		*		*
/si/	si	*	*		
	☞ ʃi			*	*
/ʃi/	☞ ʃi			*	
	si	*	*		*

RCD with unfaithful mappings

Step 1: construct mark-data pairs

Input	Winner	Loser	*si	*s	*ʃ	$\mathcal{F}(\pm$ant)
/sa/	sa	ʃa		L	W	W
/ʃa/	sa	ʃa		L	W	L
/si/	ʃi	si	W	W	L	L
/ʃi/	ʃi	si	W	W	L	W

- For each constraint, calculate Δ(winner violations, loser violations)
- Constraint preference: $\begin{cases} W & \text{if } \Delta(\text{w,l}) < 1 \\ L & \text{if } \Delta(\text{w,l}) > 1 \\ \text{tie} & \text{otherwise} \end{cases}$

RCD with unfaithful mappings

Step 2: demote

Input	Winner	Loser	*si	*s	*ʃ	$\mathcal{F}$(±ant)
/sa/	sa	ʃa		L	W	W
/ʃa/	sa	ʃa		L	W	L
/si/	ʃi	si	W	W	L	L
/ʃi/	ʃi	si	W	W	L	W

- Constraints that prefer only winners are placed in the current stratum
- Constraints that prefer losers are demoted

RCD with unfaithful mappings

Step 3: remove explained pairs

Input	Winner	Loser	*si	*s	*ʃ	$\mathcal{F}$(±ant)
/sa/	sa	ʃa		L	W	W
/ʃa/	sa	ʃa		L	W	L
/si/	ʃi	si	W	W	L	L
/ʃi/	ʃi	si	W	W	L	W

- Rows with ranked W are removed as explained
- Crucial: relies on strict domination (no lower violations can “de-explain” these mdp’s)

RCD with unfaithful mappings

Step 3: remove explained pairs

Input	Winner	Loser	*si	*s	*ʃ	$\mathcal{F}$(±ant)
/sa/	sa	ʃa		L	W	W
/ʃa/	sa	ʃa		L	W	L
/si/	ʃi	si	W	W	L	L
/ʃi/	ʃi	si	W	W	L	W

- Rows with ranked W are removed as explained
- Crucial: relies on strict domination (no longer violations can “de-explain” these mdp’s)

RCD with unfaithful mappings

Step 3: remove explained pairs

Input	Winner	Loser	*si	*s	*ʃ	$\mathcal{F}(\pm\text{ant})$
/sa/	sa	ʃa		L	W	W
/ʃa/	sa	ʃa		L	W	L

- Rows with ranked W are removed as explained
- Crucial: relies on strict domination (no longer violations can “de-explain” these mdp’s)

RCD with unfaithful mappings

Repeat: demote and remove

Input	Winner	Loser	*si	*s	*ʃ	$\mathcal{F}(\pm\text{ant})$
/sa/	sa	ʃa		L	W	W
/ʃa/	sa	ʃa		L	W	L

RCD with unfaithful mappings

Repeat: demote and remove

Input	Winner	Loser	*si	*ʃ	*s	$\mathcal{F}$(±ant)
/sa/	sa	ʃa		W	L	W
/ʃa/	sa	ʃa		W	L	L

RCD with unfaithful mappings

Repeat: demote and remove

Input	Winner	Loser	*si	*ʃ	*s	$\mathcal{F}$(±ant)
/sa/	sa	ʃa		W	L	W
/ʃa/	sa	ʃa		W	L	L

RCD with unfaithful mappings

Repeat: demote and remove

Input	Winner	Loser	*si	*ʃ	*s	$\mathcal{F}$(±ant)
/sa/	sa	ʃa		W	L	W
/ʃa/	sa	ʃa		W	L	L

RCD with unfaithful mappings

Repeat: demote and remove

Input	Winner	Loser	*si	*ʃ	*s	$\mathcal{F}$(±ant)

*si ≫ *ʃ ≫ *s, $\mathcal{F}$(±ant)

Exercise

Find a ranking consistent with the following tableau:

	C1	C2	C3	C4	C5
Datum 1	W	L	W	W	L
Datum 2		L			W
Datum 3	W	L	W		
Datum 4	L	L		W	
Datum 5		W	L		W

The consequences of demotion

/agap/ → ṯaga	Dep(C)	Ons	Dep(V)	*Coda	Max
taga ∼ agap	L	W		W	L

- (Ons ∨ *Coda) ≫ (Max ∧ Dep(C))
- Demotion: easy to ensure that each L is below a W
  - Dep(C) is “at fault”, and it doesn’t matter which W we use to neutralize it
  - Moving it below Ons maintains all other existing rankings
- Promotion: which M should we move up?
  - Ons? *Coda? both?

The consequences of demotion (*cont.*)

- This alone doesn’t seem like a big problem; could just take highest ranked and promote
- However, this still yields a different answer from demotion
  - Suppose there’s another constraint K between Dep(C) and Ons: Dep(C) $\gg$ K $\gg$ Ons
  - Demoting Dep(C) preserves K $\gg$ Ons
  - Promoting Ons reverses both rankings

A python script

RCD.py

- Takes text file of tableaus in OTSoft format
  - Inputs
  - Winners and competitors
  - Constraint violations of candidates
- Applies RCD according to Tesar’s definition
- Outputs final (partial) hierarchy

Virtues of RCD

- Converges correctly and efficiently: for N data pairs and K constraints
  - Maximally K strata $\rightarrow$ maximally K demotion steps
  - At each step, we have maximally K constraints to consider demoting (really, maximally $K - \mathit{current\ stratum}$)
  - To see if a constraint needs demoted, we must check maximally N as-yet-unexplained mdp's
  - Total: $K^2 \cdot N$
  - "Efficient" (Achilles heel: how many losing candidates?)
- That is, if there's a consistent ranking, it will find it
  - If there are multiple consistent rankings, it will find one of them (or more than one: partial hierarchy via strata)

Efficiency of error-driven learning

- Demotion only in response to error
- Error only when L not strictly dominated by a W
- L in stratum k can only generate errors/be demoted $k-1$ times

Efficiency of RCD

- Demo: RCDPayoff.pl
- ‘Real world’ efficiency depends on configuration of violations
  - Confounded

mdp	1	2	3	4	5	6
a	W	L				
b		W	L			
c			W	L		
d				W	L	
e					W	L

  - Orthogonal

mdp	1	2	3	4	5	6
a	W	L				
b			W	L		
c					W	L

Consistent ranking, but not necessarily identical

Ways in which RCD fails to arrive at grammar that generated input data

- Total hierarchy vs. partial hierarchy (strata)
    - RCD learns stratified hierarchies, but cannot (in the general case) learn languages generated by stratified hierarchies (why?)
    - Simply assume total ranking imposed later, by independent means
- Ambiguities
    - Constraints with same violation patterns (have to be placed together by RCD)
    - Constraints with just W’s and ties
    - This point is taken up in the Prince and Tesar, and Hayes (2004) papers

The RCD and strict domination

- RCD is efficient because of strict domination:
  - Once a mdp has a ‘W-preferrer’ installed, it’s guaranteed to be explained, so can be removed from consideration
- Interesting to note that under Tesar’s formulation, there are some types of consistent ranking that RCD will never find (why?)

/UR/	C1	C2	C3
☞ cand1		*	
cand2	*!		
cand3		*	*!

/UR/	C1	C2	C3
☞ cand1		*	
cand2	*!		
cand3		*	

Where do input/output pairs come from?

- Assumption so far: both are given by an omniscient being
- Relaxing this: receive just the SR, infer UR and ranking
- This can be tricky! Wrong choices could lead to dead ends

Unfortuitous choice of UR’s

Example: given SR [sa]...

- Candidate UR’s: /sa/, /ʃa/, /sap/, etc.
- In principle, given just this datum, any of these is possible (given the appropriate ranking)
- Danger: incorrect selection of UR’s creates ranking paradoxes
  - E.g., [sa] ← /sap/, [mata] ← /mat/
- A possible, but inefficient approach:
  - Hypothesize UR’s, in relatively unconstrained fashion (/UR/ → [SR] must not be harmonically bound, but free to select among possible mappings)
  - Construct mdp’s
  - Attempt to learn a consistent ranking
  - If learning terminates with no consistent ranking, randomly modify hypothesized UR’s until something

A more conservative approach

Breaking into the system

- On hearing [sa], we can be pretty sure that at the very least, the grammar must be able to map /sa/ to [sa][1]
- Less certain: does the grammar also map /sap/, /ʃa/ → [sa]?
- Start modestly: assume /sa/ (IN = OUT)
    - Mapping /sa/ → [sa] violates fewer Faithfulness constraints than any other X → [sa], and all the same Markedness constraints
    - So, this would be the "best" guess at a UR, in terms of harmony (lexicon optimization)

[1] Major caveat: counterfeeding opacity

What this buys us

How does the sa/ʃi learning scenario change when we are restricted to learning from (IN = OUT) pairs?

Input	Intended	*si	*s	*ʃ	$\mathcal{F}$([±ant])
/sa/	☞ sa		*		
	ʃa			*	*
/ʃa/	ʃa			*	
	☞ sa		*		*
/si/	si	*	*		
	☞ ʃi			*	*
/ʃi/	☞ ʃi			*	
	si	*	*		*

What this buys us

How does the sa/ʃi learning scenario change when we are restricted to learning from (IN = OUT) pairs?

Input	Intended	*si	*s	*ʃ	$\mathcal{F}([\pm\text{ant}])$
/sa/	☞ sa		*		
	ʃa			*	*
/ʃi/	☞ ʃi			*	
	si	*	*		*

What this buys us

How does the sa/ʃi learning scenario change when we are restricted to learning from (IN = OUT) pairs?

Input	Intended	*si	*s	*ʃ	$\mathcal{F}([\pm\text{ant}])$
/sa/	☞ sa		*		
	ʃa			*	*
/ʃi/	☞ ʃi			*	
	si	*	*		*

- Learning converges efficiently
- But arrives at a different answer! (why?)

The challenge of positive evidence

- Grammar distinguishes between what maps faithfully (grammatical) vs. unfaithfully (ungrammatical)
- Positive evidence tells us only what maps faithfully
- Extremely ambiguous! Faithful mappings obey all $\mathcal{F}$
    - Tied or winner-preferring for all mdp's (why?)
- Applying RCD based on positive evidence will never have reason to demote $\mathcal{F}$
    - Unintended consequence: grammar typically allows quite a bit more than was seen in input data
- The subset problem (e.g., Angluin 1980)
    - Data is ambiguous: consistent with grammars that produce many different languages
    - Claim: we want a grammar that produces the smallest possible language (allow attested forms, as little else as possible)

The subset principle

- Assumption 1: human learners do indeed prefer maximally restrictive analyses
  - I.e., they solve the subset problem, somehow
  - At a first pass, this is probably more right than it is wrong, but worth evaluating
  - We can revisit this, but for now we'll stick to the way the problem is framed in this literature
- Assumption 2: the way to solve the subset problem is by having the learner prefer the most restrictive analysis at each stage of learning
  - Premise: more permissive hypothesis can never be falsified based on positive evidence
  - So, once you adopt a more permissive analysis, you're stuck there (no counterevidence)
  - Here too, one might question the premise, but we'll grant it for now in order to see the kind of approaches

Positive evidence as fixed points

- If a form can surface unfaithfully, it should be able to surface faithfully as well

/ba/		*ɓ	$\mathcal{F}$(ɓ)	$\mathcal{F}$([±voi])	*[+voi]
☞	ba				*
	pa			*!	

/ɓa/		*ɓ	$\mathcal{F}$(ɓ)	$\mathcal{F}$([±voi])	*[+voi]
☞	ba		*		*
	pa		*	*	
	ɓa	*!			

  - Faithful mapping has subset of violations of unfaithful mapping
  - Once again, a caveat: counterfeeding opacity
- Number of forms that can surface faithfully is a metric of permissiveness of grammar

Something to keep in mind at the outset

- Even among languages that have same set of fixed points, there are many possible grammars/mappings
- E.g., among lgs that allow CV, CVC, CVCV, CVCVC
  - /prat/ → pat
  - /prat/ → pərat
- Hayes/Prince&Tesar assume that knowledge of unfaithful mappings will be refined while learning alternations
- A plausible further criterion to keep in mind:
  - Favor correct set of *unfaithful* mappings, even in absence of explicit evidence from alternations (seen, for example, in loanword phonology)

The logic of markedness and faithfulness, again

- Markedness constraints wish to ban structures
  - If input contains marked structures, prefer candidates that eliminate them
- Faithfulness constraints license structures
  - Allow forms to pass through the grammar unmodified
- Ranking of $\mathcal{M}$ and $\mathcal{F}$ determines restrictiveness of grammar
  - $\mathcal{M} \gg (\mathcal{M} \gg \ldots)\ \mathcal{F}$: neutralization
  - $\mathcal{F} \gg \mathcal{M}$: contrast

The $\mathcal{M} \gg \mathcal{F}$ bias

- Each $\mathcal{F} \gg \mathcal{M}$ ranking allows a particular structure to surface faithfully
  - E.g., $\mathcal{F}$([$\pm$voi]) $\gg$ [$+$voi]: /ba/ → [ba]
- The fewer the $\mathcal{F} \gg \mathcal{M}$ rankings we have, the fewer the number of structures that the grammar will allow
- Bias: $\mathcal{M} \gg \mathcal{F}$

Initial $\mathcal{M} \gg \mathcal{F}$ bias is not enough

Going back to the *sa/ʃi* allophonic language: try $\mathcal{M} \gg \mathcal{F}$ bias

Input	Winner ∼ Loser	*si	*s	*ʃ	$\mathcal{F}$([±ant])
/sa/	sa ∼ *ʃa		L	W	W
/ʃi/	ʃi ∼ *si	W	W	L	W
	Stratum	1	1	1	2

- Stage 1: *si, *s, *ʃ ≫ $\mathcal{F}$([±ant])
  - *si is the only unranked constraint that doesn't prefer a loser
  - $\mathcal{F}$ prefers all winners, but starts out ranked below stratum 1

Initial $\mathcal{M} \gg \mathcal{F}$ bias is not enough

Going back to the *sa/ʃi* allophonic language: try $\mathcal{M}\gg\mathcal{F}$ bias

Input	Winner , Loser	*si	*s	*ʃ	$\mathcal{F}$([±ant])
/sa/	sa , ʃa		L	W	W
/ʃi/	ʃi , si	W	W	L	W
	Stratum	1	2	2	2

- Stage 1: *si, *s, *ʃ $\gg$ $\mathcal{F}$([±ant])
    - *si is the only unranked constraint that doesn't prefer a loser
    - $\mathcal{F}$ prefers all winners, but starts out ranked below stratum 1
- Stage 2: *si $\gg$ *s, *ʃ, $\mathcal{F}$([±ant])
    - *ʃ, $\mathcal{F}$([±ant]) both prefer winner
    - *s demoted to next stratum

Initial $\mathcal{M} \gg \mathcal{F}$ bias is not enough

Going back to the *sa/ʃi* allophonic language: try $\mathcal{M}\gg\mathcal{F}$ bias

Input	Winner , Loser	*si	*ʃ	$\mathcal{F}$([$\pm$ant])	*s
/sa/	sa , ʃa		W	W	L
/ʃi/	ʃi , si	W	L	W	W
	Stratum	1	2	2	3

$\mathcal{M} \gg \mathcal{F}$ as a persistent bias

- In previous example, we need to maintain *ʃ $\gg$ $\mathcal{F}$([$\pm$ant])
- That is, given ambiguous datum /sa/ → [sa], prefer markedness-based explanation
- Persistent bias: wherever possible, rank $\mathcal{M} \gg \mathcal{F}$
  - Give $\mathcal{M}$ ‘first crack’ at ambiguous data
  - Deploy $\mathcal{F}$ only where truly needed
- Similar observations by many authors
  - Ito and Mester (1999); Prince and Tesar (2004); Hayes (2004)

A metric of degree of $\mathcal{M} \gg \mathcal{F}$

Prince and Tesar (2004, p. 7): The r-measure
“The r-measure for a constraint hierarchy is determined by adding, for each faithfulness constraint in the hierarchy, the number of markedness constraints that dominate that faithfulness constraint.”

- Does not attempt to calculate set of possible output forms; estimates instead by more holistic measure of number of $\mathcal{M} \gg \mathcal{F}$ rankings
- In general, more restrictive $=$ higher r-measure
- Max r-measure $= |\mathcal{M}| \times \ |\mathcal{F}|$
- Does not favor a single unique grammar, but a set of (equally restrictive??) grammars
- MaxEnt: maximize w($\mathcal{M}$) $-$ w($\mathcal{F}$)

Implementing the persistent bias: BCD

Biased constraint demotion (BCD)

- $\mathcal{F}$ delay

  "On each pass, among those constraints suitable for membership in the next stratum, if possible place only markedness constraints. Only place faithfulness constraints if no markedness constraints are available to be placed in the hierarchy."

  - This alone is sufficient in our simple sa/Si case

Implementing the persistent bias: BCD (***cont.***)

- Don't place inactive $\mathcal{F}$ constraints

  "When placing faithfulness constraints into the hierarchy, if possible only place those that prefer some winner"

  - Consequence: prefer placing $\mathcal{F}$ constraints that free up $\mathcal{M}$ constraints for ranking on next iteration
  - If none of remaining $\mathcal{F}$ prefers a winner, place them all

Implementing the persistent bias: BCD

F-gangs

- Sometimes more than one active $\mathcal{F}$ constraint is needed before another $\mathcal{M}$ constraint can be placed

	F1	F2	F3	M1
a. W1 $\sim$ L1	W		W	L
b. W2 $\sim$ L2		W		L
c. W3 $\sim$ L3	W			L

- {F1, F2}, or {F3, F2, F1}

☞ Prefer smaller set (the smaller “F-gang”), in this case leaving out gratuitous F3

Implementing the persistent bias: BCD

M-cascades

- Sometimes choice of which $\mathcal{F}$ to rank has additional consequences for r-measure

	F1	F2	M1	M2
a. W1 $\sim$ L1	W		L	
b. W2 $\sim$ L2		W	W	L

- M1 and M2 both out at moment, since both prefer L
- Pick F1 or F2?
  - F1 $\gg$ M1 $\gg$ M2 $\gg$ F2 (r$=$2)
  - F2 $\gg$ M2 $\gg$ F1 $\gg$ M1 (r$=$1)
- Favor $\mathcal{F}$ that permits longest "$\mathcal{M}$-streak" until next $\mathcal{F}$

Summary so far

Biased Constraint Demotion

- Set of learning preferences, applied during ranking
    - Easy: Favor $\mathcal{M}$, favor active $\mathcal{F}$
    - Costlier: Favor smaller $\mathcal{F}$-gangs, and longer $\mathcal{M}$-cascades
- Local goal: delay ranking $\mathcal{F}$ as long as possible
    - Increases r-measure
    - Correspondingly, increases restrictiveness of grammar
- Reassuringly, works at least for the simple sa/ʃi case

Unfortunately...

Hayes (2004)

- BCD does not always achieve the most restrictive analysis, either
- Hayes proposes an alternative (LFCD = Low Faithfulness Constraint Demotion), which incorporates a number of different biases
- This is more successful, though Prince and Tesar point out cases where it will not work, either

What these algorithms have in common

- Preference for constraints that generate the right outputs (prerequisite for convergence)
- Preference for markedness constraints, as more restrictive
- Preference for faithfulness constraints that clearly and uniquely explain sets of forms
- Some type of preference for more restricted faithfulness constraints (directly through specificity, or indirectly through examining consequences for freeing up markedness constraints)

The nature of the approach

- “Tread cautiously”
  - Hayes: “To learn that (say) *[ad] is ill-formed, Pseudo-Korean infants must use a conservative strategy, plausibly along the lines of ‘if you haven’t heard it, or something like it, then it’s not possible.”
- Constraint demotion is non-probabilistic:
  - If current ranking generates error, re-rank; else keep current grammar
  - No attempt to assess predicted probability distribution; model doesn’t “pay” for over-allowing possible structures
- Constraint demotion is brittle
  - Errors, exceptions, etc., can create also forms outside the language
  - Don’t want just a single [at^h] token during training to change adult language too radically

Another issue: RCD is brittle

Training data

tʰa	tʰa	da	da	tʰa	da
tʰa	da	tʰa	tʰa	tʰa	tʰa
da	da	tʰa	ta	tʰa	da
da	da	da	tʰa	da	tʰa

- *D, *Tʰ/Dʰ (no voiced, aspirated obstruents)
- *T/#___ (no initial unaspirated stops)
- $\mathcal{F}$([±voi]) (faithfulness to voicing)
- $\mathcal{F}$([±s.g.]) (faithfulness to aspiration)

RCD is brittle

- Desired:
  - *T/#___, $\mathcal{F}$([$\pm$voi]) $\gg$ *D, $\mathcal{F}$([$\pm$s.g.])

    $=$ “Initial aspiration” (plus one anomalous token)
- BCD answer:
  - $\mathcal{F}$([$\pm$voi]) $\gg$ *D and[2] $\mathcal{F}$([$\pm$s.g.]) $\gg$ *T^{h}/#___

    $=$ “Three-way contrast”
- Problematic under both batch and on-line EDCD
- Idea: make demotion less responsive to small amounts of data

[2] Ranking: $\mathcal{M} \gg \mathcal{F} \gg \mathcal{M} \gg \mathcal{F}$, to maximize r-measure

Slower reranking could also help with other things

Recall argument for persistent $\mathcal{M} \gg \mathcal{F}$ bias

Input	Winner , Loser	*si	*ʃ	$\mathcal{F}$([±ant])	*s
/sa/	sa , ʃa		W	W	L
/ʃi/	ʃi , si	W	L	W	W
	Stratum	1	2	2	3

- Stage 1: *si, *s, *ʃ $\gg \mathcal{F}$([±ant])
  - *si is the only $\mathcal{M}$ constraint that doesn't prefer a loser
  - $\mathcal{F}$ prefers winners, but starts ranked below stratum 1
- Stage 2: *si $\gg$ *s, *ʃ, $\mathcal{F}$([±ant])
  - *ʃ, $\mathcal{F}$([±ant]) both prefer winner
  - *s demoted to next stratum
- Stage 3: *si $\gg$ *ʃ, $\mathcal{F}$([±ant]) $\gg$ *s
  - Two compatible refinements: $\mathcal{F} \gg \mathcal{M}$, $\mathcal{M} \gg \mathcal{F}$
  - Initial bias doesn't guarantee complementary distribution

A solution usinng slower reranking?

- Premise is that ‘initial two-stratum’ approach is insufficient
    - Stratum 2 by initial bias
    - Stratum 2 by demotion
- This particular example would not be a problem if we had some way of making the bias ‘greater’
    - $\mathcal{M}$: Stratum 1
    - $\mathcal{F}$: Stratum 3
- Requires numerical interpretation of strata
    - The Gradual Learning Algorithm (GLA; Boersma, 1997; Boersma and Hayes, 2001)

The Gradual Learning Algorithm

The Gradual Learning Algorithm (GLA; Boersma, 1997; Boersma and Hayes, 2001)

- Reranking does not proceed stratum by stratum, creating strict rankings at each stage
- Rather, constraints are moved in small increments, getting closer and closer together and finally switching places

/ta/	*Dʰ	$\mathcal{F}$(±s.g.)	*T/#___
tʰa $\sim$ *ta		L	W

The Gradual Learning Algorithm

The Gradual Learning Algorithm (GLA; Boersma, 1997; Boersma and Hayes, 2001)

- Reranking does not proceed stratum by stratum, creating strict rankings at each stage
- Rather, constraints are moved in small increments, getting closer and closer together and finally switching places

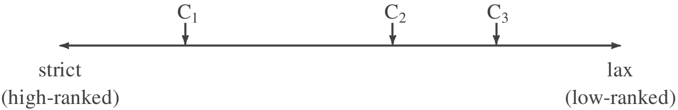
/ta/	$*D^h$	$\mathcal{F}(\pm\text{s.g.})$	*T/#___
$t^ha \sim *ta$		$\rightarrow$L	W$\leftarrow$

Implementing small steps

Encoding gradient rankings

- Instead of constraint strata (partial hierarchy), we need something that will let us express degrees of distance
- Boersma proposes a ranking scale: constraints are given numerical values corresponding to their importance

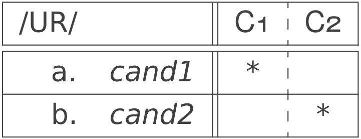
(1) *Categorical ranking of constraints (C) along a continuous scale*



(Boersma and Hayes 2001, p. 47)

Dealing with ties

Two different interpretations of tied constraints

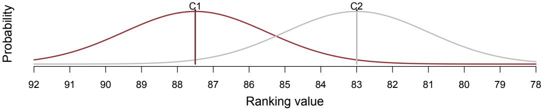


/UR/	C_1	C_2
a. *cand1*	*	
b. *cand2*		*

- Tesar and Smolensky (1996/2001): cancelation within the stratum
  - In order to avoid this, they impose a restriction: RCD assumes that data has been generated by a *total hierarchy*
- Anttila, Boersma, others: optionality/free variation
  - $C_1 > C_2$: 100% *cand1*
  - $C_1 = C_2$: 50% *cand1*, 50% *cand2*
  - $C_1 < C_2$: 100% *cand2*

Stochastic ranking to assign probabilities

- (Boersma, 1997; Boersma and Hayes, 2001)
    - Rankings are sampled from distributions at time of evaluation
    - Relative frequency of patterns is determined by ranking distributions



- Independent of choice of ranking algorithm or framework
    - Boersma proposed gradual reranking, stochastic values, OT evaluation
    - Tesar proposed recursive reranking, fixed values, OT evaluation
    - Also possible to use gradual weighting, fixed/stochastic values, HG evaluation

Part 1: gradual constraint demotion

- Scenario: start with initial state of $\mathcal{M} \gg \mathcal{F}$
  - Ranking value of $\mathcal{M} = 100$
  - Ranking value of $\mathcal{F} = 0$
- Input to the algorithm: as before, triples of underlying form, surface form, losing candidate (mdp’s)
- If current grammar does not predict correct surface form, demote loser-preferring constraints by a certain amount
  - Amount of adjustment $=$ plasticity (η)
  - For simplicity, let’s assume $\eta = 1$ for now

Gradual constraint demotion

- Given sufficiently small steps (low plasticity), a few erroneous training items will not be enough to lead to a grammar in which such forms are considered legal
- Thus, gradual demotion successfully increases the robustness of the algorithm
- It also creates an interesting situation:
  - As constraint C_1 moves below C_2, they may momentarily tie
  - What should happen in this case?

Learning in the GLA

- Constraints initialized to default weights
  - All same, $\mathcal{M} \gg \mathcal{F}$ bias, etc.
- Input to the algorithm: as before, triples of underlying form, surface form, losing candidate (mdp’s)
- Invoke current grammar (with noise): if correct surface form not predicted correctly, nudge winner-preferrers up slightly and loser-preferrers down slightly
  - Again, amount of adjustment $=$ plasticity (η)
  - Plasticity may decrease over time, according to some schedule
  - Symmetric adjustments (demotion and promotion); see Boersma (1997) for discussion

(See how this is accomplished in gla.py)

Part 2: probabilistic interpretation of ranking distance

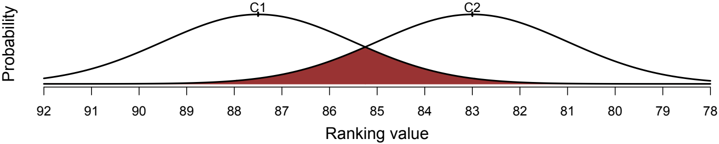
Intuitively:

- C_1 much higher than C_2: *cand1* always wins
- C_1 slightly above C_2: *cand1* usually wins, sometimes *cand2*
- $C_1 = C_2$: equal probability
- C_1 slightly below C_2: *cand2* usually wins, sometimes *cand1*
- C_1 much lower than C_2: *cand2* always wins

/UR/	C_1	C_2
a. cand1	*	
b. cand2		*

Modeling variation with variable rankings

- Ranking values are not merely points along a continuum; they are probability distributions
- For example, the Gaussian distributions below:

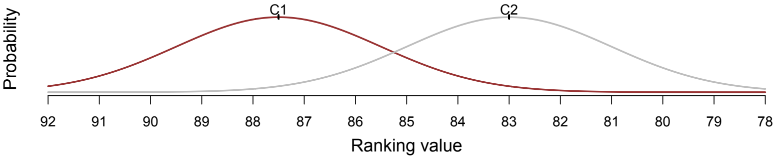


(Boersma and Hayes 2001, p. 49)

- Each time the grammar is invoked, ranking values are chosen from the distribution (selection points)

Example

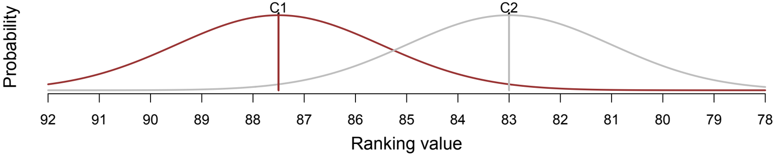
Two constraints, partially overlapping distributions



- On average, C_1 (red) $\gg$ C_2 (gray)

Example

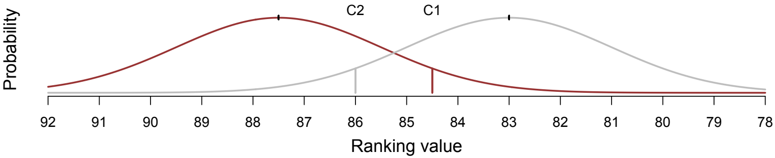
Two constraints, partially overlapping distributions



- On average, C_1 (red) $\gg$ C_2 (gray)
- Most choices of selection points will respect this ranking

Example

Two constraints, partially overlapping distributions



- On average, C_1 (red) $\gg$ C_2 (gray)
- Most choices of selection points will respect this ranking
- Occasionally, selection points might reverse preference

Relation between overlap and production probability

- Constraints far apart: vanishingly small chance of reversal
  - Indistinguishable from noise/errors
- As constraints get closer, probability of competition/variability increases

Symmetrical adjustments

With noise/variation, symmetrical promotion/demotion is crucial

- In cases of variation, data poses a ranking paradox:
  - Variant 1: requires $C_1 \gg C_2$
  - Variant 2: requires $C_2 \gg C_1$
- Demotion only: constraints will continue to leapfrog in perpetuity
- Symmetric adjustment: creates a “stalemate”
  - Balancing act/tug of war between the two variants

The road not taken

In principle, distributions have other parameters that could be varied

- Spread (size of standard deviation): allows constraints to interact at greater distances
  - All GLA applications that I know of hold the size of a standard deviation constant across all constraints
- Skewing of distribution
  - I know of no attempts to vary this in the GLA

Gradualism and acquisition

- As we saw with Dutch, incremental updates provide one way of modeling time course of acquisition
  - Initial state unlike target ⇒ systematic errors
  - Gradual reranking to achieve necessary ranking conditions ⇒ milestones in mastering adult structures
- Prediction: children may pass through stages that are typologically attested, but make distinctions that appear “unmotivated” by the input data

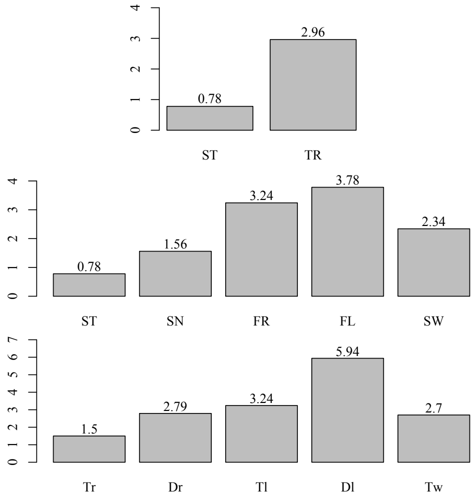
Albright and Magri (2013)

Albright and Magri (2013) Perceptually motivated epenthesis asymmetries in the acquisition of consonant clusters. GLOW

- Investigated order of acquisition of English consonant clusters, in database of normally developing American English children
- Iowa-Nebraska articulatory norms project (Smit et al. 1990): elicited productions of fixed set of target words

ST:	**SN:**	**SL, FR:**	**TL:**		**STR:**
3	3	5	14		5
spoon	snake	frog	twins	clock	straw
star	snail	three	queen	block	screen
skate	smoke	swing	dress	bridge	spray

Epenthesis, by cluster type

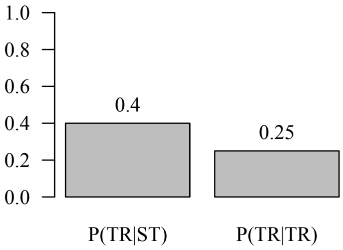


Entailments

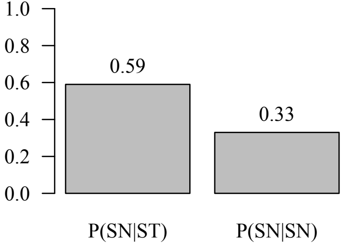
In general:

	ST		**SN**		**SR**		**TR**
a.	ST	$\Longrightarrow$	$\Longrightarrow$	$\Longrightarrow$	$\Longrightarrow$	$\Longrightarrow$	TR
b.	ST	$\Longrightarrow$	SN				
c.	ST	$\Longrightarrow$	$\Longrightarrow$	$\Longrightarrow$	SR		
d.			SN	$\Longrightarrow$	SR		

Entailments

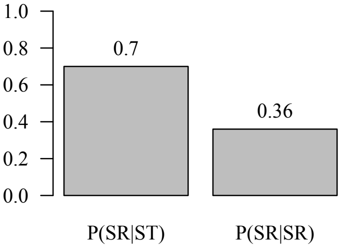


$$\frac{P(\text{TR}|\text{ST})}{P(\text{TR}|\text{TR})} = 1.6 > 1$$

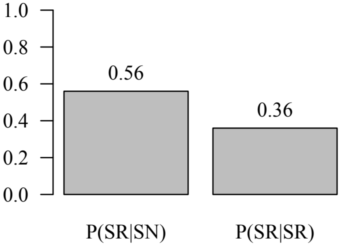


$$\frac{P(\text{SN}|\text{ST})}{P(\text{SN}|\text{SN})} = 1.7 > 1$$

Entailments



$$\frac{P(\mathrm{SR}|\mathrm{ST})}{P(\mathrm{SR}|\mathrm{SR})} = 1.9 > 1$$



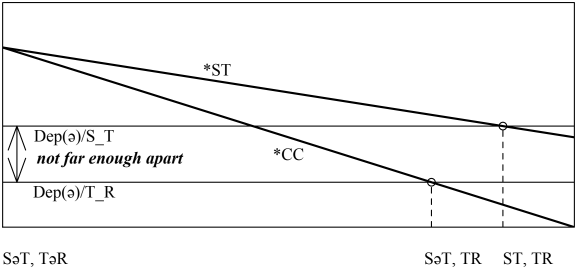
$$\frac{P(\mathrm{SR}|\mathrm{SN})}{P(\mathrm{SR}|\mathrm{SR})} = 1.5 > 1$$

Capturing these asymmetries

- Dep(V)/S___ T $\gg$ Dep(V)/T___ R
- Consistent with P-Map hypothesis (Fleischhacker 2001, 2005)

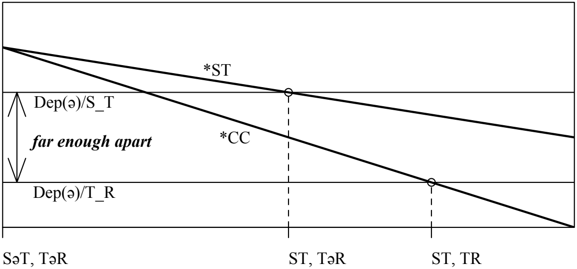
Initial ranking by itself is not enough

Dep constraints not far enough apart: less marked TR mastered first



Initial ranking + distance

DEP constraints sufficiently far apart: more marked ST mastered first



Crucial pieces for modeling this

- Gradual reranking
- Numeric values (define what it means to be 'gradual',

The GLA and restrictiveness

- There is nothing built into the GLA to solve the problem of settling on the most restrictive grammar compatible with the data
- Initial $\mathcal{M} \gg \mathcal{F}$ ranking helps, assuming sufficient distance
- However,
    - In fact, the GLA can have exactly the opposite of the desired effect for things like “Prefer specific” (since general constraints get nudged more often, by wider variety of input forms)

(See how the GLA performs on the sa/ʃi languages, including one with variation)

Unfortunately, the GLA can fail to converge (Pater, 2008)

/UR$_1$/	C$_1$	C$_2$	C$_3$	C$_4$	C$_5$
☞ a. cand1		*			
b. cand2	*!		*		

/UR$_2$/	C$_1$	C$_2$	C$_3$	C$_4$	C$_5$
☞ a. cand1			*		
b. cand2		*!		*	

/UR$_3$/	C$_1$	C$_2$	C$_3$	C$_4$	C$_5$
☞ a. cand1				*	
b. cand2			*!		*

/UR$_4$/	C$_1$	C$_2$	C$_3$	C$_4$	C$_5$
☞ a. cand1					*
b. cand2				*!	

- Try this case out with the GLA script and see what

Non-convergence in the GLA

- Pater (2008): suggests a “credit problem”
  - This language is plagued by particularly many ambiguous violations: is UR_1, cand1 explained by C_1, or C_3?
- Magri (2012) Convergence of error-driven ranking algorithms. *Phonology* 29:213–269
  - No combination of error-driven updates can possibly lead to ranking $C1 \gg C2 \gg C3 \gg C4 \gg C5$
  - Solution is outside search space of GLA

Non-convergence in the GLA

$/UR_1/$	C1	C2	C3	C4	C5
☞ a. cand1		*			
b. cand2	*!		*		

$/UR_2/$	C1	C2	C3	C4	C5
☞ a. cand1			*		
b. cand2		*!		*	

$/UR_3/$	C1	C2	C3	C4	C5
☞ a. cand1				*	
b. cand2			*!		*

$/UR_4/$	C1	C2	C3	C4	C5
☞ a. cand1					*
b. cand2				*!	

Non-convergence in the GLA

Representative updates in response to this data

Datum	C1	C2	C3	C4	C5
Start	100	100	100	100	100
UR_1	W: 101	L: 99	W: 101	100	100
UR_3	101	99	W: 102	L: 99	W: 101
UR_2	101	W: 100	L: 101	W: 100	101
UR_3	101	100	W: 102	L: 99	W: 102
UR_1	W: 102	L: 99	W: 103	99	102
UR_4	102	99	103	W: 100	L: 101
etc.					

Non-convergence in the GLA

Representative updates in response to this data

Datum	C1	C2	C3	C4	C5
Start	100	100	100	100	100
UR$_1$	W: 101	L: 99	W: 101	100	100
UR$_3$	101	99	W: 102	L: 99	W: 101
UR$_2$	101	W: 100	L: 101	W: 100	101
UR$_3$	101	100	W: 102	L: 99	W: 102
UR$_1$	W: 102	L: 99	W: 103	99	102
UR$_4$	102	99	103	W: 100	L: 101
etc.					

Final total of updates

$$\theta = \alpha_1 \begin{bmatrix} +1 \\ -1 \\ +1 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ +1 \\ -1 \\ +1 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 0 \\ +1 \\ -1 \\ +1 \end{bmatrix} + \alpha_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ +1 \\ -1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 - \alpha_1 \\ \alpha_1 + \alpha_3 - \alpha_2 \\ \alpha_2 + \alpha_4 - \alpha_3 \\ \alpha_3 - \alpha_4 \end{bmatrix}$$

Non-convergence in the GLA

- Only one compatible ranking: four conditions

1: C1 $\gg$ C2 $\Rightarrow \alpha_1 > \alpha_2 - \alpha_1$

2: C2 $\gg$ C3 $\Rightarrow \alpha_2 - \alpha_1 > \alpha_1 + \alpha_3 - \alpha_2$

3: C3 $\gg$ C4 $\Rightarrow \alpha_1 + \alpha_3 - \alpha_2 > \alpha_2 + \alpha_4 - \alpha_3$

4: C4 $\gg$ C5 $\Rightarrow \alpha_2 + \alpha_4 - \alpha_3 > \alpha_3 - \alpha_4$

- Can't all be satisfied at once! e.g.,

a (2−4): $-\alpha_1 - \alpha_4 + \alpha_3 > \alpha_1 - \alpha_2 + \alpha_4 \Rightarrow -2\alpha_1 + \alpha_2 + \alpha_3 - 2\alpha_4 > 0$

b (3+4): $\alpha_1 + \alpha_4 > \alpha_2 \Rightarrow \alpha_1 - \alpha_2 + \alpha_4 > 0$

c (1+4): $\alpha_1 + \alpha_2 + \alpha_4 - \alpha_3 > \alpha_2 + \alpha_3 + -\alpha_4 - \alpha_1 \Rightarrow \alpha_1 - \alpha_3 + \alpha_4 > 0$

d (a+b): $2\alpha_1 - \alpha_2 - \alpha_3 + 2\alpha_4 > 0 \Rightarrow -2\alpha_1 + \alpha_2 + \alpha_3 - 2\alpha_4 < 0$

- No combination of updates will lead to the desired ranking!

Magri’s solution

Part 1: a response to the credit problem

/UR/	C1	C2	C3
a. mdp1	L	W	W

- C1 must be dominated by one of {C2, C3}
- Be cautious: demote L by 1, promote W’s by $\frac{1}{|W|}$
- Or, equivalently, demote L by $|W|$, promote W’s by 1,

Magri’s solution

Part 2: multiple L’s

/UR/	C1	C2	C3
a. mdp1	L	L	W

- Both of C1, C2 must be dominated by C3
- Completely equivalent decomposition, in OT terms:

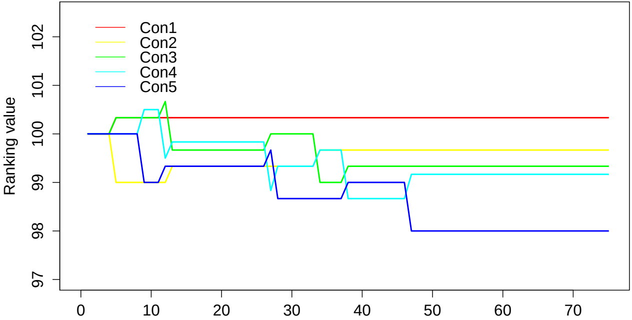
/UR/	C1	C2	C3
a. mdp1	L		W
a′. mdp2		L	W

- Promote W’s by number of undominated L’s

Putting this together

When the grammar generates an error for a mdp...

- Demote *undominated* L's by 1 (or $n \propto$ number of W's)
- Promote W's by $\dfrac{\text{number of undominated L's}}{1 + \text{number of W's}}$
- As before, actual amount can be scaled by plasticity η
- Implementation: `GLAMagri.py`



An OT–MaxEnt link

Magri (2013): a link between OT compatibility and linear models

- As previously discussed, grammars of weighted constraints can produce different effects than OT strict domination (ganging up)
    - Not in general possible to translate analyses back and forth
- However, it is possible to achieve a more narrow equivalence
    - Given a particular OT-compatible comparative tableau, transform it so that any compatible weighting will correspond to an OT-compatible ranking

Illustration

Consider again Pater’s example

/UR$_1$/	C1	C2	C3	C4	C5
☞ a. cand1		*			
b. cand2	*!		*		

/UR$_3$/	C1	C2	C3	C4	C5
☞ a. cand1				*	
b. cand2			*!		*

/UR$_2$/	C1	C2	C3	C4	C5
☞ a. cand1			*		
b. cand2		*!		*	

/UR$_4$/	C1	C2	C3	C4	C5
☞ a. cand1					*
b. cand2				*!	

- OT-compatible ranking: C1 $\gg$ C2 $\gg$ C3 $\gg$ C4 $\gg$ C5
- Similar linear weighting would work (C1 $>$ C2 $>$ C3 $>$ C4 $>$ C5), but so would many others
- What would a MaxEnt solution look like?

Illustration

MaxEnt solution to the Pater example

Con1	5.15
Con2	11.23
Con3	15.26
Con4	12.05
Con5	4.33

- C3 $>$ C4 $>$ C2 $>$ C1 $>$ C5
- Definitely not OT-compatible

Illustration

Magri (2013, p. 583)

- Given: OT-compatible comparative tableau of W's, L's, e's
  - $|W|$ = number of W's in a given row
  - Only one L per row (decompose as necessary)
- Transform: numerical differences
  - W $\Rightarrow$ 1
  - e $\Rightarrow$ 0
  - L $\Rightarrow -|W|$

  (or, L=1 and W=1/$|W|$)
- Claim: any order of linear weights that works for the transformed tableau will be an OT-compatible ranking

The logic of this equivalence

- Multiple W’s

	C1	C2	C3
winner $\sim$ loser	W	W	L

- Ranking/weighting conditions
  - OT: one of {C1,C2} $\gg$ C3
  - HG: w(C1) $+$ w(C2) $>$ w(C3)
- Compare transformed tableau

	C1	C2	C3
winner $\sim$ loser	1	1	-2

  - w(C1) + w(C2) $>$ 2·w(C3)
  - Only satisfied if at least one of w(C1) $>$ w(C3) or w(C2) $>$ w(C3) (cf. ranking condition)

HG compatible ⇒ OT compatible

$$0 \quad < \quad \sum_{c=1}^{n} \theta_c \bar{a}_c$$

Weighted violation differences (loser−winner) prefer the winner, in aggregate (definition of HG compatible weighting)

$$< \quad \sum_{i \in W(a)} \theta_i \bar{a}_i + \sum_{j \in L(a)} \theta_j \bar{a}_j + \sum_{k \notin W(a) \cup L(a)} \theta_k \bar{a}_k$$

Partition into W's, L's, e's (also covers rows with multiple L's)

$$< \quad \sum_{i \in W(a)} \theta_i - |W| \sum_{j} \theta_j + 0$$

$\bar{a}$ of W's is 1 (by definition on prev. slide), and $\bar{a}$ of L's is $-|W|$; e's contribute nothing

HG compatible ⇒ OT compatible

$$0 < \sum_{i \in W(a)} \theta_i - |W| \sum_{j} \theta_j + 0$$

$$< |W| \max_{i \in W(a)} \theta_i - |W| \sum_{j \in L(a)} \theta_j$$

Upper bound on sum of *n* items is $n \times$max

$$< |W| \max_{i \in W(a)} \theta_i - |W| \theta_j$$

Since all weights are non-negative, $\sum_{j \in L(a)} \theta_j \geq$ any arbitrary θj

$$max\theta_i > \theta_j$$

There’s a W above all L’s!

OT compatible ⇒ HG compatible

- An OT-compatible tableau can be rearranged thusly:

$$
\begin{array}{c|ccccccc|}
 & C_1 & C_2 & \ldots & C_d & C_{d+1} & \ldots & C_n \\
\hline
 & \mathrm{W} & & & & & & \\
dec(C_1) & \vert & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
 & \mathrm{W} & & & & & & \\
 & \mathrm{E} & \mathrm{W} & & & & & \\
dec(C_2) & \vert & \vert & \ldots & \ldots & \ldots & \ldots & \ldots \\
 & \mathrm{E} & \mathrm{W} & & & & & \\
 & \mathrm{E} & \mathrm{E} & & & & & \\
\vdots & \vert & \vert & \ldots & \ldots & \ldots & \ldots & \ldots \\
 & \mathrm{E} & \mathrm{E} & & & & & \\
 & \mathrm{E} & \mathrm{E} & & \mathrm{W} & & & \\
dec(C_d) & \vert & \mathrm{E} & \ldots & \vert & \ldots & \ldots & \ldots \\
 & \mathrm{E} & \mathrm{E} & & \mathrm{W} & & & \\
\hline
\end{array}
$$

- And a ranking vector constructed as follows:

$$
\begin{aligned}
\theta_{d+1} = \ldots = \theta_n \quad &\doteq \quad 0 \\
\theta_k \quad &\doteq \quad \max\left\{\theta_{k+1}, \max_{\mathbf{a}\in dec(C_k)} \frac{1}{\overline{a}_k}\left(1 - \sum_{h=k+1}^{n} \theta_h \overline{a}_h\right)\right\}, \quad k = d, d-1, \ldots, 1
\end{aligned}
$$

OT compatible ⇒ HG compatible

$$\sum_{h=1}^{n} \theta_h \bar{a}_h \geq 1 \quad \Leftrightarrow \quad \sum_{h=1}^{k-1} \theta_h \bar{a}_h + \theta_k \bar{a}_k + \sum_{h=k+1}^{n} \theta_h \bar{a}_h \geq 1$$

For arbitrary row $\bar{a}$, partition at decisive constraint k

$$\Leftrightarrow \quad \theta_k \bar{a}_k + \sum_{h=k+1}^{n} \theta_h \bar{a}_h \geq 1$$

Cells above k are e's, $\bar{a}_{>k} = 0$

$$\Leftrightarrow \quad \theta_k \geq \frac{1}{\bar{a}_k} \left(1 - \sum_{h=k+1}^{n} \theta_h \bar{a}_h \right)$$

$$\Leftarrow \quad \theta_k \geq \max \; \frac{1}{\;} \left(1 - \sum^{n} \theta_h \bar{a}_h \right)$$

The upshot

- Any ranking that yields the correct outputs for the given OT tableau corresponds to a set of weights that yields the right outcome in HG for the derived tableau
- Of course, these formalisms still have different powers in general
  - Could easily construct new rows that would have different outcomes under OT vs HG evaluation
- Potential benefit: can explore use of learning algorithms for linear/weighted constraint models for learning OT-compatible rankings

Illustration

Transformed tableau:

/UR$_1$/			C_1	C_2	C_3	C_4	C_5
☞	a.	cand1		2			
	b.	cand2	1		1		

/UR$_2$/			C_1	C_2	C_3	C_4	C_5
☞	a.	cand1			2		
	b.	cand2		1		1	

/UR$_3$/			C_1	C_2	C_3	C_4	C_5
☞	a.	cand1				2	
	b.	cand2			1		1

/UR$_4$/			C_1	C_2	C_3	C_4	C_5
☞	a.	cand1					1
	b.	cand2				1	

Illustration

MaxEnt solution for the transformed tableau

Con1	30.4
Con2	16.9
Con3	7.6
Con4	2.5
Con5	0.0

- C1 $>$ C2 $>$ C3 $>$ C4 $>$ C5
- This one is OT-compatible!

References

Boersma, Paul. 1997. How we learn variation, optionality, and probability. Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam 21, 43–58. http://fon.hum.uva.nl/paul/.

Boersma, Paul and Bruce Hayes. 2001. Empirical Tests of the Gradual Learning Algorithm. Linguistic Inquiry 32(1), 45–86.

Doyle, Gabriel; Klinton Bicknell; and Roger Levy. 2014. Nonparametric learning of phonological constraints in Optimality Theory. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 1094–1103. Baltimore, Maryland:

References (*cont.*)

Association for Computational Linguistics. URL https://aclanthology.org/P14-1103.

Hayes, Bruce. 2004. Phonological acquisition in Optimality Theory: The early stages. In Fixing Priorities: Constraints in Phonological Acquisition, ed. by René Kager; Joe Pater; and Wim Zonneveld, 158–203. Cambridge: Cambridge University Press.

Hayes, Bruce and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. Linguistic Inquiry 39, 379–440.

Pater, Joe. 2008. Gradual learning and convergence. Linguistic Inquiry 39, 334–345.

References (*cont.*)

Prince, Alan and Paul Smolensky. 2004. Optimality Theory: Constraint Interaction in Generative Grammar. Blackwell Publishing.

Rasin, Ezer and Roni Katzir. 2015. Compression-based learning for OT is incompatible with Richness of the Base. In Proceedings of NELS 45, ed. by Thuy Bui and Deniz Özyıldız, volume 2, 267–274.

—. 2016. On evaluation metrics in Optimality Theory. Linguistic Inquiry 47, 235–282.