Ranking biases to achieve restrictiveness

24.964—Fall 2004
Modeling phonological learning

Class 7 (21 Oct, 2004)
Review of last time

Recursive Constraint Demotion (RCD):

Construct list of (winner, loser) pairs

Demote all constraints that prefer a loser

Remove all data pairs in which the winner is now correctly preferred

Repeat until no pairs are remaining to be explained
Review of last time

Result of the RCD:

- Every constraint that prefers a loser is ranked immediately below the constraints that prefer the corresponding winner(s)

- Constraints that never prefer losers are ranked on top
Demonstration of the RCD

A simple language, with allophonic alternation:

- /sa/ → [sa]
- /si/ → [ʃi]
- /ʃa/ → [ʃa]
- /ʃi/ → [ʃi]

([s] and [ʃ] not contrastive; distribution governed by vocalic environment)
Demonstration of the RCD

Steps:

• Convert to MDP’s (comparative tableau form is handy!)

• Apply RCD
Problems with the RCD

What data is available to the learner about this language?

- /sa/ → [sa]
- /si/ → [ʃi]
- /ʃa/ → [ʃa]
- /ʃi/ → [ʃi]
Problems with the RCD

Surface [sa], [ʃi] restrict the set of pairs (overtly) available to the learner

- /sa/ → [sa]

- /ʃi/ → [ʃi]

(Why are the other pairs not posited, at least under the RCD as presented by Tesar & Smolensky?)
Problems with the RCD

Surface [sa], [şi] compatible with a variety of languages:

- [s]/[ʃ] completely allophonic
  - [sa], [şi], but no *[şi], *[ʃa]

- [s]/[ʃ] contrastive except /i
  - [sa], [ʃa], [şi], but no *[şi]

- [s]/[ʃ] contrastive everywhere
  - [sa], [ʃa], [şi], *[şi]
The subset principle

Angluin 1980, Berwick 1985

- Always choose the *most restrictive* available analysis
  - [sa], [ʃi], but no *[si], *[ʃa]
Trying to capture the subset principle

How does the RCD do on a language with just [sa], [ji] inputs?

• Unmodified RCD
The idea: rank $\mathcal{F}$ low

- A restrictive grammar is one that doesn't allow stuff to surface unmodified; since $\mathcal{F}$ constraints prefer to let marked structures surface, we want to rank them as low as possible.

- The IN=OUT assumption about learning in OT also tends to underestimate the number of faithfulness violations (by giving the learner pairs that are as close to the identity map as possible). A bias against $\mathcal{F}$ can help correct for this.
A simple idea that doesn’t work

Initial ranking of $M \gg F$

• (What does it yield in this case?)
Another idea: ranking conservatism

Itô & Mester (1999):

- Initial state of $\mathcal{M} \gg \mathcal{F}$

- Learner is biased to preserve current rankings as much as possible

(Does this word on the [sa]/[ʃ] language?)
Trying to capture the subset principle

More sophisticated modifications:

- BCD (Prince & Tesar)
  - Prefer $\mathcal{M}$
  - Among $\mathcal{F}$, prefer those that “free up” $\mathcal{M}$

- LFCD (Hayes)
  - Prefer $\mathcal{M}$
  - Among $\mathcal{F}$, prefer those that are *active*, *specific*, and *autonomous*
A test language: Pseudo-Korean

The basic pattern:

• Aspiration is contrastive before Vs: [ta] vs [tʰa]

• Unaspirated stops voice intervocalically: /ata/ → [ada]
  ○ Aspirated stops do not: [ada] vs [atʰa]

• Aspiration contrast neutralized word-finally: [at] (*[atʰ])
A test language: Pseudo-Korean

The relevant markedness constraints:

- \( *[+\text{voi}][-\text{voi}][+\text{voi}] \) (motivates intervocalic voicing)

- \( *[+\text{voi}, +\text{spread glottis}] \) \( (*[d^h]) \); blocks intervocalic voicing of aspirated stops

- \( *-\text{SON}, +\text{VOI} \) (no voiced obstruents; blocks voicing wherever possible)

- \( *\text{ASPIRATION} \) (motivates de-aspiration wherever possible)
A test language: Pseudo-Korean

The relevant faithfulness constraints:

- IDENT(asp), IDENT(asp)/ V
- IDENT(voi), IDENT(voic)/ V

(Steriade 1997; pre-vocalic (more accurately: pre-sonorant) position is better able to support laryngeal cues)
A test language: Pseudo-Korean

Sample words of Pseudo-Korean

- [ta], [tʰa]
- [ada], [atʰa]
- [at]
- [tada], [tatʰa], [tʰada], [tʰatʰa], [tat], [tʰat]
A test language: Pseudo-Korean

Some crucial rankings:

- $*\text{d}^h \gg *[+\text{voi}][-\text{voi}][+\text{voi}] \gg \text{Ident(voi)}, \text{Ident(voi)}/_V$
  - $/\text{ata}/ \rightarrow [\text{ada}]$, but $/\text{at}^h\text{a}/ \rightarrow [\text{at}^h\text{a}]$

- $\text{Ident(asp)}/_V \gg *\text{asp} \gg \text{Ident(asp)}$
  - $/\text{t}^h\text{a}/ \rightarrow [\text{t}^h\text{a}]$, but $/\text{at}^h/ \rightarrow [\text{at}]$

- $*[+\text{v}][-\text{v}][+\text{v}] \gg *[−\text{son},+\text{voi}], \gg \text{Ident(voi)}, \text{Ident(voi)}/_V$
  - $/\text{ata}/ \rightarrow [\text{ada}]$, but $/\text{da}/ \rightarrow [\text{ta}]$ (presumably)
A test language: Pseudo-Korean

Hayes, pp. 18-19: The RCD fails miserably

- Why? (Does RCD.pl confirm this?)
Trying to do better: LFCD

What principles would guide the ranking algorithm to better choices?

- Initial constraint set:

<table>
<thead>
<tr>
<th>$\mathcal{M}$</th>
<th>$\mathcal{F}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>*d^h</td>
<td>Ident(asp)</td>
</tr>
<tr>
<td>[+voi][−voi][+voi]</td>
<td>Ident(asp) / _ V</td>
</tr>
</tbody>
</table>
Trying to do better: LFCD

Step 1: identify set of NoLosers

- \([+\text{voi}][-\text{voi}][+\text{voi}]\) dislikes \([at^h a]\), prefers \([ad^h a]\)
- \([-\text{son},+\text{voi}]\) dislikes \([ada]\), prefers \([ata]\)
- aspiration dislikes \([t^h a]\), prefers \([ta]\)

So NoLosers includes:

- \([d^h, \text{Ident(asp)}, \text{Ident(asp)}/\_ V, \text{Ident(voi)}, \text{Ident(voi)}/\_ V]\)
Trying to do better: LFCD

Favoring markedness:

- $*d^h >>$ everything else

Explains all mdp’s involving $[d^h]$
Trying to do better: LFCD

Step 2: identify new set of NoLosers

- *[+voi][−voi][+voi] still dislikes [atʰa], prefers *[adʰa]
- *[−son,+voi] still dislikes [ada], prefers *[ata]
- *aspiration still dislikes [tʰa], prefers *[ta]

Now NoLosers includes just $\mathcal{F}$:

- Ident(asp), Ident(asp)/_V, Ident(voi), Ident(voi)/_V
Trying to do better: LFCD

Favor specificity:

- Intuition is that we want to admit as few new marked structures as possible

- Accomplish this by employing $\mathcal{F}$ constraints that are as specific as possible (allow marked structures in a narrow range of contexts)

Here: favor Ident(asp)/_ V, Ident(voi)/_ V over Ident(asp), Ident(voi)  
(BUT: which one???)
Some problems with specificity

Prince & Tesar, section 6 (p. 23)

“We are reluctant to take this step, because it does not solve the general problem. There are at least two areas of shortfall…: First, two constraints that have only partial overlap in their domain of application can, under the right circumstances, end up in a special to general relationship. Second, the special/general relation that is relevant to restrictiveness can hold between constraints that seem quite independent of each other.”
Trying to do better: LFCD

Favor autonomy:

- Similar to principle of “free up markedness”: we want to shift the burden of explanation to markedness constraints as much as possible. So, if there’s a possibility that a $\mathcal{M}$ constraint might be able to do the work down the line, don’t “steal its thunder” by installing a $\mathcal{F}$ that does it already

(Example: Hayes p. 24)
Trying to do better: LFCD

One other principle: Favor activeness

- Discussed also by Prince & Tear: if a $F$ constraint doesn’t hurt, but also doesn’t help (by favoring a winner somewhere), then ranking it above other constraints will do no good (and could hurt on other inputs, not yet seen)

- Delay ranking such constraints until the very end

- Example: faithfulness for ejectives in English
Trying to do better: LFCD

Putting it together: ordered decisions (see LFCD.pl)

- Eliminate losers
- If both $\mathcal{M}$ and $\mathcal{F}$, eliminate $\mathcal{F}$
- If only $\mathcal{F}$, eliminate inactive ones
- If still multiple possibilities, eliminate more specific ones
- If still multiple possibilities, choose the one with greatest autonomy
What these algorithms have in common

- Preference for constraints that generate the right outputs (obviously)

- 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)
Another way in which grammars may fail

Discussion up to this point has focused on “unimagined inputs that surface faithfully”

- That is, inputs that are not part of the actual language (or, at least, are absent from the initial learning data)

- Difficulty arises when grammar accidentally predicts that they should occur

Another large source of trouble: “unimagined candidates”

- Example in [sa]/[ʃi] language: fixing input /si/ by changing to [sa]

What constraint/ranking is needed to rule this out? What learning pair is needed to learn this?
Where this is leading

- In all of these approaches, the idea is to make faithfulness constraints justify their position in the ranking.

- This requires estimating which one is “truly” responsible for the pattern, and which ones happen to apply to the current learning data.

- Various unresolved issues (how to favor specificity? how to implement a lasting preference so $\mathcal{F}$ constraints “sink” if their inputs are later reanalyzed?)

- Perhaps a more important issue, though: what counts as “good” evidence for demoting? Simply favoring a loser or being the wrong kind of $\mathcal{F}$ constraint? Is there some better way to reason about the relation between pairs of constraints?
Where this is leading

Next time, we will discuss the following paper(ette):

• Albro (2000) *A probabilistic ranking learner for phonotactics*

It is sketchy, and I don’t actually understand it at present; the method described here is an attempt to introduce some important techniques to constraint ranking, however, so it is worth trying to make sense of.
For next week

A short computer assignment:

- Prince & Tesar (1999) discuss the problematic azba language (section 6). Prepare an input file of tableaus demonstrating the azba language, that can be run in RCD.pl and LFCD.pl. Hayes (1999) claims that LFCD.pl works on the azba language. Does it? How?

Readings:

- The LFCD.pl implementation of Hayes’ proposal (in this week’s perlscripts directory)


- Necessary background for the preceding: an introduction to Bayes’ rule