Boersma’s Gradual Learning Algorithm

Simplifying assumptions
- Child knows input-output mappings (doesn’t have to figure out what the inputs are)
- Child knows or has already learned the constraint set
- The (finite) set of relevant competitors is known

Advantages over Constraint Demotion
- Robust to errors, because gradual
- Can handle variation (we’ll take this up in Week 8)
- Realistically models learning stages (see Suzanne Curtin’s presentation in lab today)

Stochastic constraints
(Boersma 1998, Hayes & MacEachern 1998)
For Boersma, the ranking of constraints is neither fixed nor freely variable, but probabilistic.

- Each constraint in an individual’s grammar has a ranking value, given in arbitrary units.
- In each utterance, the speaker generates selection points (“disharmony” for Boersma) for each constraint

\[
\text{selectionPoint} = \text{rankingValue} + \text{rankingSpreading} * z
\]

- rankingSpreading (“noise” in OTSoft) is some constant, typically 2
- \(z\) is a Gaussian random variable with mean 0 and standard deviation 1. Gaussian random variables are what give us “bell curves” (normal distributions):
Example of selection points for 10 runs of a 2-constraint grammar (Boersma p. 331)

<table>
<thead>
<tr>
<th>trial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>50.5</td>
<td>51.2</td>
<td>50.2</td>
<td>49.1</td>
<td>52.9</td>
<td>52.9</td>
<td>52.7</td>
<td>53.8</td>
<td>55.4</td>
<td>54.3</td>
</tr>
<tr>
<td></td>
<td>(ranking value = 50)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>50.8</td>
<td>48.3</td>
<td>50.7</td>
<td>51.2</td>
<td>48.9</td>
<td>48.8</td>
<td>48.2</td>
<td>50.3</td>
<td>48.1</td>
<td>48.7</td>
</tr>
<tr>
<td></td>
<td>(ranking value = 50)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Constraints are ranked, for the utterance in questions, according to their selection points. We can call this the “effective ranking”.

We can think of each constraint as being associated with a probability density function centered on the ranking value:

![Probability Density Function Diagram]

*How it works*

1. Starting with some default set of rankings values.

Perhaps all ranking values are 100, or perhaps Markedness constraints start at 500, Faithfulness at 100.

2. Pick a word.

Word choice can be completely random, or as we’ll see in Week 8, it can be weighted for frequency.

This could be something children do in their spare time or while asleep, or it could be an action they perform every time they hear a word.
3. Generate an effective ranking and make a tableau for how you would say the word under that ranking.

<table>
<thead>
<tr>
<th>/k’at’a/</th>
<th>IDENT-IO (PLACE)</th>
<th>IDENT-IO (LARYNGEAL)</th>
<th>-LARYNGEAL SIMILARITY</th>
<th>BE IDENTICAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>k’ata</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>b</td>
<td>k’at’a</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>c</td>
<td>t’at’a</td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. If your output was correct, do nothing. Otherwise compare the real winner to your winner, canceling any marks that they share.

<table>
<thead>
<tr>
<th>/k’at’a/</th>
<th>IDENT-IO (PLACE)</th>
<th>IDENT-IO (LARYNGEAL)</th>
<th>-LARYNGEAL SIMILARITY</th>
<th>BE IDENTICAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>k’ata</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>b</td>
<td>k’at’a</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

5. Demote the constraints for which the winner has uncancelled marks, and promote the constraints for which the loser has uncancelled marks.

<table>
<thead>
<tr>
<th>/k’at’a/</th>
<th>IDENT-IO (PLACE)</th>
<th>IDENT-IO (LARYNGEAL)</th>
<th>-LARYNGEAL SIMILARITY</th>
<th>BE IDENTICAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>k’ata</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>b</td>
<td>k’at’a</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Demotion and promotion here mean adjusting the ranking value up or down by some small amount (the “plasticity”). Typically, the plasticity might start out at 2 and gradually decrease to 0.002.

6. Repeat from 2. for the rest of your life.

As the ranking values are adjusted, mistakes will become less and less frequent, and adjustments will happen rarely or never.

(The idea of maintaining a lifelong ability to adjust your grammar is appealing: accents can drift when people move, for example.)
Turkel’s Genetic Algorithm

*Genetic Algorithms*
GAs use the mechanisms of mutation and natural selection to solve problems. They’re used in engineering, artificial life…all over the place.

- Start with a set of solutions
- Allow the fittest ones to “mate” and mutate; kill the rest
- Repeat above step with “offspring” solutions

*Constraints as chromosomes*

\[ A >> B >> C >> D >> E >> F \]

- One-point crossover with mark cancellation:
  \[ \begin{align*}
    A >> B >> C >> D >> E >> F & \quad \rightarrow \quad A >> B >> C >> F >> E >> D \\
    F >> E >> C >> A >> D >> B & \quad \rightarrow \quad F >> E >> C >> A >> B >> D
  \end{align*} \]

- Mutation
  - Swap adjacent
    \[ \begin{align*}
    A >> B >> C >> D >> E >> F & \quad \rightarrow \quad A >> C >> B >> D >> E >> F
    \end{align*} \]
  - Swap nonadjacent
    \[ \begin{align*}
    A >> B >> C >> D >> E >> F & \quad \rightarrow \quad A >> E >> C >> D >> B >> F
    \end{align*} \]
  - Reverse segment
    \[ \begin{align*}
    A >> B >> C >> D >> E >> F & \quad \rightarrow \quad A >> D >> C >> D >> E >> F
    \end{align*} \]
  - Rotate left
    \[ \begin{align*}
    A >> B >> C >> D >> E >> F & \quad \rightarrow \quad B >> C >> D >> E >> F >> A
    \end{align*} \]
  - Rotate right
    \[ \begin{align*}
    A >> B >> C >> D >> E >> F & \quad \rightarrow \quad F >> A >> B >> C >> D >> E
    \end{align*} \]

*How it works*

1. Randomly generate a set of constraint rankings

\[ \begin{align*}
    \text{Id[PLACE]} & >> \text{*LARSIM} >> \text{BEIDENT} >> \text{Id[LAR]} \\
    \text{Id[PLACE]} & >> \text{*LARSIM} >> \text{Id[LAR]} >> \text{BEIDENT} \\
    \text{*LARSIM} & >> \text{Id[PLACE]} >> \text{BEIDENT} >> \text{Id[LAR]} \\
    \text{BEIDENT} & >> \text{Id[LAR]} >> \text{Id[PLACE]} >> \text{*LARSIM} \\
    \text{Id[LAR]} & >> \text{Id[PLACE]} >> \text{*LARSIM} >> \text{BEIDENT}
    \end{align*} \]
2. Assess the fitness of each ranking

Assume every constraint is binary. Then we can describe each candidate’s constraint violations as a binary number (“M-ranking”):

<table>
<thead>
<tr>
<th>/k’at’a/</th>
<th>-LARYNGEAL SIMILARITY</th>
<th>IDENT-IO (LARYNGEAL)</th>
<th>IDENT-IO (PLACE)</th>
<th>BE IDENTICAL</th>
<th>harmony</th>
</tr>
</thead>
<tbody>
<tr>
<td>a k’ata</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0101</td>
</tr>
<tr>
<td>b k’at’a</td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td>1001</td>
</tr>
<tr>
<td>c t’at’a</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td>1010</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/t’ata/</th>
<th>-LARYNGEAL SIMILARITY</th>
<th>IDENT-IO (LARYNGEAL)</th>
<th>IDENT-IO (PLACE)</th>
<th>BE IDENTICAL</th>
<th>harmony</th>
</tr>
</thead>
<tbody>
<tr>
<td>d t’at’a</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td>1100</td>
</tr>
<tr>
<td>e t’ata</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td>0001</td>
</tr>
</tbody>
</table>

Low numbers represent greater harmony (zeros at the beginning = satisfaction of high-ranked constraints).

The fittest ranking is the one that assigns the lowest average number to the winning candidates

\[ \text{ID[PLACE]} \gg \text{*LARSIM} \gg \text{BEIDENT} \gg \text{ID[LAR]} \]
\[ \text{ID[PLACE]} \gg \text{*LARSIM} \gg \text{ID[LAR]} \gg \text{BEIDENT} \]
\[ \text{ID[LAR]} \gg \text{BEIDENT} \gg \text{ID[PLACE]} \gg \text{*LARSIM} \]
\[ \text{BEIDENT} \gg \text{ID[LAR]} \gg \text{ID[PLACE]} \gg \text{*LARSIM} \]
\[ \text{ID[LAR]} \gg \text{ID[PLACE]} \gg \text{*LARSIM} \gg \text{BEIDENT} \]

3. Choose the fittest constraints, and allow them to mutate and combine

\[ \text{ID[PLACE]} \gg \text{*LARSIM} \gg \text{BEIDENT} \gg \text{ID[LAR]} \]
\[ \text{ID[PLACE]} \gg \text{*LARSIM} \gg \text{ID[LAR]} \gg \text{BEIDENT} \]
\[ \text{ID[LAR]} \gg \text{ID[PLACE]} \gg \text{*LARSIM} \gg \text{BEIDENT} \]

*mutations and combinations:*

\[ \text{ID[LAR]} \gg \text{*LARSIM} \gg \text{ID[LAR]} \gg \text{BEIDENT} \] (crossover)
\[ \text{*LARSIM} \gg \text{BEIDENT} \gg \text{ID[LAR]} \gg \text{ID[PLACE]} \] (rotate left)

4. Repeat from 2. until grammar converges

Not sure what the criterion for convergence is: none of the mutations are better than the previous generation? How many generations does that have to go on for?
**Plausibility**
Is this a realistic model of what kids do?
Boersma points out the problem of having to keep many hypothesized grammars in mind at once.
How do you choose which one to use in production?

It would be fun to write an implementation of this.

**Next time**
- Learnability: Gold paradigm and PAC
- The initial state
- Lexicon Optimization
- Richness of the Base

**For next time**
- Readings (available on ROA)
- Homework
  - Using the same OT paper as in last week’s Constraint Demotion assignment, use OTSoft to run the Gradual Learning Algorithm. Turn in the final constraint ranking, the results of testing the grammar, and a graph of the ranking history. You should run the GLA at least twice, changing plasticity, noise, or initial ranking values. Discuss the differences, if any, between the results that you got on each run—can you tell what caused the differences?
Week 5 Lab: Genetic Algorithms and GLA

UAMIS AI Group
Non-phonological Genetic Algorithm example
http://ai.bpa.arizona.edu/~mramsey/ga.html

OTSoft
1. Create Excel file

<table>
<thead>
<tr>
<th>inputs</th>
<th>outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>zero for losers, 1 for winners</td>
</tr>
<tr>
<td></td>
<td>constraint names (full)</td>
</tr>
<tr>
<td></td>
<td>constraint names (short)</td>
</tr>
<tr>
<td></td>
<td>number of violations (blank = 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Ident[place]</th>
<th>BEldentical</th>
<th>Ident[laryngeal]</th>
<th>LaryngealSimilarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>k'at'a</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>k'ata</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>t'a'ta</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>t'ata</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>t'ata</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If you want a constraint to be treated as a faithfulness constraint, its name must begin with “Id”, “Max”, “Dep”, or “Fai”. If you want a constraint to be treated as a markedness constraint, its name must begin with something else.

I’ll refer to this file as myfile.xls, but you should name it something more memorable.

2. Download OTSoft
http://www.linguistics.ucla.edu/people/hayes/otsoft/

Follow directions on web page, except that you’ll probably need to unpack everything to the User folder.

3. Run the GLA
Double-click on otsoft.exe, then click “Work with different file” and find your Excel file. Choose the GLA button on the left side of the screen, and click on “Rank myfile.xls”. Play around with the options on the screen that appears, then click “Run GLA”.

LING 599: Computation in OT, Kie Zuraw, Spring 2001
4. **View the results**
Click “View results” back on the main screen. At the top you’ll see the final ranking values that the algorithm achieved, then the results of applying that ranking to all the inputs. If there weren’t enough learning trials, you’ll see some error.
If you want to cut and paste this information, look for a file called myfile.out, in the same folder as myfile.xls

5. **View the ranking history**
In the same folder as myfile.xls, you should see a file myfileHistory.xls. Open that file. You’ll see the constraint names across the top row; if they’re shifted over to the right, move them over so that they line up with the columns of numbers. Each row of numbers represents the ranking value of each constraint. You’ll see that most of the ranking values change over time.

To make a graph of the ranking history, click in the top left cell, then click on the Chart Wizard button on the toolbar (or go to the Insert menu and select Chart…). Choose the chart type Line, and chart sub-type Line (the upper-left option). Click Finish.