Class 4: Model selection continued; Probability distributions over OT rankings I

To do for Monday

- Try making a regression model in R for your own data.
  - I recommend starting early in case you run into frustrations with your input file or with R.
  - If you don't have real data yet, make a fake input file.
  - Remember that each column should be a variable and each row should be one observation.
  - Get rid of any spaces or punctuation in your input file.
  - Save the input file as a .txt file
  - Turn in a **brief** description of what you got, including your R output for `summary(your_model.lm)`.
- Read Boersma & Hayes
- Install OTSoft on a computer that you have access to.
  - Run the Gradual Learning Algorithm in the normal way on the Anttila data (no initial rankings specified, "number of times to go through forms" is 1000 or more)
  - Turn in a **brief** report on what happened, including your OTSoft output file:
    - did the constraints that Anttila puts in the same stratum get the same ranking value?
    - where did the GLA fit the data worse or better than the Anttilan model did?

Overview: More about choosing among models; our first type of quantitative Optimality Theory

1 Summary of model selection last time

- We discussed the problem of how to choose between models
  - On the one hand, it should **fit** the observed data well
  - On the other hand, it should not be so exactly fitted to the observed data that it makes poor predictions about new data—it should not **overfit**
    - The more **complex** the model, the more danger of overfitting
- We saw two popular methods in statistics for balancing these factors
  - Akaike Information Criterion (AIC): a function of the number of independent variables (complexity) and the "likelihood" (fit). Smaller numbers are better.
  - If one model is a subset of the other, we can look up the ratio of their likelihoods[1] (how much better the complex model's fit is) and the difference in number of independent variables (how much more complex the complex model is) in a table to get a *p*-value.
    - lower *p*-values are evidence against the simpler model
    - R can do this easily using the `anova()` function.

- Issue: counting independent variables is a crude measure of complexity. Imagine:

underlying form	*final C*	*complex final coda?*	*# of syllables*	*dep. var.: final C deletes?*
ard	d	1	1	1
lit	t	0	1	1
folt	t	1	1	0
med	d	0	1	0
sabild	d	1	2	1
trod	d	0	1	1
sit	t	1	1	1
klont	t	0	1	1
slad	d	0	1	0
marikt	t	1	2	1

[1] or rather 2 * that ratio's log

  - simple model: uses just *final C*
  - complex model #1: *final C* and *complex final coda?*
  - complex model #2: *final C* and *# of syllables*
  - When we compare each of the more-complex models to the simple model, they both incur an additional-complexity penalty of 1 (1 additional independent variable)
  - But, *complex final coda* is much more powerful than *# of syllables*
    - because disyllables are rare in this language, having a different rate of C deletion for disyllables doesn't add that much complexity to the model
    - adding *# of syllables* also doesn't increase the fit that much
  - So we might unfairly reject the model with *# of syllables* because it adds an additional independent variable without buying much improvement in fit
- There are also model-selection approaches that don't count the number of independent variables but instead look at how big their (normalized) coefficients are.
  - including an independent variable but giving it a small coefficient incurs less complexity penalty
- Some regression functions even incorporate these penalties directly, making the coefficients a bit smaller (in absolute value—that is, closer to 0) than the best fit.
  - We'll get into the math and how to actually do this when we talk about MaxEnt OT next week.

2 Machine-learning approach: training set vs. cross-validation set vs. test set

- Now we depart from traditional statistics. If you already know statistics and Tuesday and Wednesday were too much review for you, here is something new!
- Here's a method that doesn't rely on estimating complexity, to estimate how badly the model would do on new data. Instead, it tests directly how badly the model would do on new data.
- I got attracted to this approach by Andrew Ng's wonderful free online course in machine learning. I've included a link on the course web page.
- The idea is to test how well the model does on new data by *holding back some data* and then treating it as new.

Training set:
  - Randomly select part of the data, say 60%. Fit each model to these data.

Cross-validation set:
  - Of the remaining 40%, randomly select some, say half. See how well each model fits *those* data.
- Repeat until you've chosen your final model.

Test set:
  - See how well your final model fits the remaining 20% of the data. Report this as your accuracy.

- In engineering applications, such as natural language processing, the last step (test set) is very important because we need to estimate the true accuracy.
- In theoretical linguistics, we might be able to skip it (and thus use more data for training and cross-validation)
  - we're less interested in quantifying how well the model performs and more interested in which factors are really active in cognition.

3 Diagnostics

- If your model fits the training set well (low error), but fits the cross-validation set poorly, you probably have overfitting.
  - Consider omitting some independent variables (or constraints, or interaction terms, or...)
  - Or consider putting greater penalties on large coefficients
- If your model fits both training and cross-validation data poorly, you probably have underfitting
  - Consider adding some independent variables (or constraints, or interaction terms, or...)
  - Or consider putting lesser penalties on large coefficients

4 Multiple cross-validation

- Especially if you have not very much data, people sometimes repeat the training-development steps:
  - E.g., divide your training-and-development data into 10 (or some other number) equal subsets. ("10-fold cross-validation")
    - Train on sets 1-9, evaluate on 10
    - Train on sets 1-8 and 10, evaluate on 9
    - Train on sets 1-7 and 9-10, evaluate on 8
    - etc.
  - Choose the model that does the best in overall on the 10 evaluations.
  - Or, for every single data item, train on the rest and then test on it. ("leave-one-out cross-validation")

- What I like about this approach is that it doesn't matter what kind of model we use: regression, stochastic OT, MaxEnt OT...
  - all we need is a measure of how well a model's predictions fit the data
  - for a continuous dependent variable, we can sum the squared errors (difference between actual and predicted)
  - for a binary dependent variable, we can count the number of incorrect classifications (model says 0 when it should be 1 or vice-versa)
    - or, better: -log(model's predicted probability) if actual value is 1
    - –log(1-model's predicted probability) if actual value is 0

5 Cross-validation for regression in R

- Fortunately, there are some packages in R that will automate cross-validate for us, at least for regression.
- One is the `boot` package (Davison & Hinkley 1997, Canty & Ripley 2012)
  - `Packages > Install packages...`
  - Choose Austria (or elsewhere) from the menu that appears
  - Choose `boot` from the menu that appears
  - Add this command to your script: `library(boot)`
  - For examples, type `help("cv.glm")`
- I had some code to show you for Hungarian but I seem to have a bug (I'm getting the same number for every model), so I'll show you on-screen but I don't want to put it in the handout because I don't want to infect you with my mistake.

Part II of today: quantitative Optimality Theory
Probability distributions over classic OT rankings I: Anttila's partial rankings

6 Multiple grammars

- One way to think about within-speaker variation is that the speaker has multiple grammars to choose between
  - There's quite a bit of work under this view on language change (Niyogi 2006).
- Some examples:
  - {S → NP VP, VP → V NP } and {S → VP NP, VP → NP V} (old-fashioned syntactic grammars)
  - {V → Ø / __V}, {} [i.e., make no change to VV], and {Ø → t / V__V}
  - MAX-C >> *θ >> IDENT(continuant) and MAX-C >> IDENT(continuant) >> *θ

7 Probability distributions over grammars

- But this class is about modeling variation quantitatively, so we want something more like:
  - {S → NP VP, VP → V NP }$_{60\%}$ and {S → VP NP, VP → NP V}$_{40\%}$
  - {V → Ø / __V}$_{20\%}$, {}$_{10\%}$ and {Ø → t / V__V}$_{70\%}$
  - MAX-C >> *θ >> IDENT(continuant)$_{15\%}$ and MAX-C >> IDENT(continuant) >> *θ$_{85\%}$
- Labeling each grammar with how often it should be used is a form of **probability distribution**
  - a probability distribution over a variable tells us how probable each value of the variable is
  - if the variable is "outcome of rolling a die [주사위]", there are 6 possible values (1,2,3,4,5,6), and the probability distribution is (for a fair die):
    - {1: 0.17, 2: 0.17, 3: 0.17, 4: 0.17, 5: 0.17, 6: 0.17}
  - if the variable is "grammar chosen", then in our OT example above there are 2 possible values, one with probability 0.15 and one with 0.85
    - a candidate's probability of winning is the probability of choosing one of the rankings that generates it.

8 Putting restrictions on probability distributions

- If we have *n* constraints, we have *n*! possible rankings:
  - MAX-C >> IDENT(place) >> *θ >> IDENT(continuant) >> *DENTAL
  - MAX-C >> IDENT(place) >> *θ >> *DENTAL >> IDENT(continuant)
  - MAX-C >> IDENT(place) >> IDENT(continuant) >> *θ >> *DENTAL
  - MAX-C >> IDENT(place) >> IDENT(continuant) >> *DENTAL >> *θ
  - MAX-C >> IDENT(place) >> *DENTAL >> *θ >> IDENT(continuant)
  - MAX-C >> IDENT(place) >> *DENTAL >> IDENT(continuant) >> *θ
  - MAX-C >> *θ >> IDENT(place) >> IDENT(continuant) >> *DENTAL
  - MAX-C >> *θ >> IDENT(place) >> *DENTAL >> IDENT(continuant)
  - MAX-C >> *θ >> IDENT(place) >> IDENT(continuant) >> *θ >> *DENTAL
  - MAX-C >> *θ >> IDENT(place) >> IDENT(continuant) >> *DENTAL >> *θ
  - MAX-C >> *θ >> IDENT(place) >> *DENTAL >> *θ >> IDENT(continuant)
  - MAX-C >> *θ >> IDENT(place) >> *DENTAL >> IDENT(continuant) >> *θ
  - ... and 108 more
- Some of these rankings produce the same candidate.
  - So, to find a candidate's probability of winning, we have to add up the probabilities of the rankings that choose it.

- In principle, we could allow any probability distribution over these rankings:
  - **3% : MAX-C >> IDENT(place) >> *θ >> IDENT(continuant) >> *DENTAL**
  - **0% : MAX-C >> IDENT(place) >> *θ >> *DENTAL >> IDENT(continuant)**
  - 2% : MAX-C >> IDENT(place) >> IDENT(continuant) >> *θ >> *DENTAL
  - 2% : MAX-C >> IDENT(place) >> IDENT(continuant) >> *DENTAL >> *θ
  - 0% : MAX-C >> IDENT(place) >> *DENTAL >> *θ >> IDENT(continuant)
  - 1% : MAX-C >> IDENT(place) >> *DENTAL >> IDENT(continuant) >> *θ
  - **0% : MAX-C >> *θ >> IDENT(place) >> IDENT(continuant) >> *DENTAL**
  - **3% : MAX-C >> *θ >> IDENT(place) >> *DENTAL >> IDENT(continuant)**
  - 5% : MAX-C >> *θ >> IDENT(place) >> IDENT(continuant) >> *θ >> *DENTAL
  - 1% : MAX-C >> *θ >> IDENT(place) >> IDENT(continuant) >> *DENTAL >> *θ
  - 2% : MAX-C >> *θ >> IDENT(place) >> *DENTAL >> *θ >> IDENT(continuant)
  - 1% : MAX-C >> *θ >> IDENT(place) >> *DENTAL >> IDENT(continuant) >> *θ
  - ... and 108 more, adding up to 100%
- But it seems strange to say that in the first two grammars, **IDENT(continuant) >> *DENTAL** is preferred, but in the other bolded pair, ***DENTAL >> IDENT(continuant)** is preferred.
- Such an arbitrary probability distribution means that the speaker has to learn 120 different probabilities (or 119, since the last one can be calculated by subtracting from 100%!)
- We might want to try more-restrictive models first...
  - maybe instead the speaker learns a number for each constraint, and derives the probability distribution from that
  - or the speaker learns a number for each pair of constraints
  - or...

9 Back to variable constraint ranking

- Recall that for Labov's New York City (th) case, we could say this:

/θɪk/	*θ	IDENT(cont)
☞ *a* [θɪk]	*	
☞ *b* [t̪ɪk]		*

Jagged line: constraint ranking *varies*.

Dotted line: ranking is *unknown*.

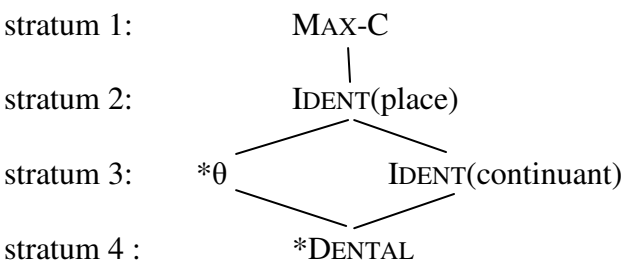
- More of the ranking:

/θɪk/	MAX-C	IDENT(place)	*θ	IDENT(cont)	*DENTAL
☞ *a* [θɪk]			*		*
☞ *b* [t̪ɪk]				*	*
c [ɪk]	*!				
d [sɪk]		*!			

- This is like saying:
  - something >0 % : MAX-C >> IDENT(place) >> *θ >> IDENT(continuant) >> *DENTAL
    - and/or something >0 % : IDENT(place) >> MAX-C >> *θ >> IDENT(cont) >> *DENTAL
  - something >0 % : MAX-C >> IDENT(place) >> IDENT(continuant) >> *θ >> *DENTAL
    - and/or something >0 % : IDENT(place) >> MAX-C >> IDENT(cont) >> *θ >> *DENTAL
  - 0% : every other ranking

10 Anttila's proposal

- A speaker's grammar is a ranking of constraints into *strata*:



for convenience, I'll assume that MAX-C >> IDENT(place), although we can't know

- If a stratum has more than one constraint in it, every time the speaker makes a tableau she/he must arrange those constraints into a linear order
  - each order of constraints within a stratum is equally probable
- In our example, this means we have the following probability distribution over rankings:
  - 50% : MAX-C >> IDENT(place) >> *θ >> IDENT(continuant) >> *DENTAL
  - 50% : MAX-C >> IDENT(place) >> IDENT(continuant) >> *θ >> *DENTAL
  - 0% : every other ranking
- o Let's discuss the restrictions that this theory places on probability distributions: what are some things that can't happen?

11 Finnish example

- As you read, Anttila proposes the following constraint ranking for Finnish genitives:

(50) The grammar for Finnish, final version

SET 1	SET 2	SET 3	SET 4	SET 5
*X.X	*Ĺ	*H/I	*H/O	*H/A
	*H	*Í	*Ó	*Á
		*L.L	*L/A	*L/O ≫ *L/I
			*H.H	*A ≫ *O ≫ *I
			*H́	*L
			*X.X	

(Anttila 1997, p. 21)

- o Let's rewrite this in the format we used above and consider some parts of the probability distribution over rankings
- The predicted outcomes depend on the set of constraints
  - This is true in any type of model
  - But it's even more true here because the theory puts such strong restrictions on the probability distribution over rankings
  - Unlike in a regression model, we can't adjust coefficients: a constraint is either in a certain stratum or not in that stratum
- o Let's discuss the reading question: what are some changes to the constraint set that would be a-priori reasonable? How do we think they might change the outcomes?

12 software: OTSoft

- It is quite tedious to compute the probability of each candidate by hand, especially with a big grammar like the Finnish one.
- Fortunately, there is software that will help us—it does many other things too.
- You can download OTSoft for free from Bruce Hayes's webpage (see course webpage for direct link; Hayes & al. 2011).

  - Only for Windows, unfortunately, but you can do most of the same things in Praat too (see course webpage for link; Boersma & Weenink 2012)
- Switch to projector: I'll demonstrate how to use OTSoft to do simple OT ("Constraint demotion" button). (I'll post the Excel file on the course webpage so you can try it yourself)
  - If you download OTSoft you'll see it has a pretty good instruction manual.

13 How to do Anttilan partial ordering in OTSoft

- I don't know of a way to reliably *learn* an Anttilan grammar with OTSoft, but if we know what the grammar is there's a way to run it and get candidate percentages.
- Switch to projector:
  - I downloaded the file Anttila_data.txt from Paul Boersma (www.fon.hum.uva.nl/paul/gla)
  - I reformatted it into an OTSoft file (posted on course webpage; there might be some other, minor changes)
  - We can apply OTSoft to the file but choose a different option, Gradual Learning Algorithm (which we'll learn about next week).
  - We can use the "Initial rankings" menu to tell the program what the rankings should be. We assign bigger numbers to higher strata, with the numbers 20 points apart (next week we'll see what this means)
  - Set "Number of times to go through forms" to zero—this tells the program not to try to learn anything; just use the rankings we gave it
  - The results file tells us how often each candidate won in the testing phase!
- If we have extra time, we can try modifying some of these constraints or moving them to a different stratum and see what happens.

References

Anttila, Arto. 1997. Deriving variation from grammar. In Frans Hinskens, Roeland van Hout and Leo Wetzels (eds.), *Variation, Change and Phonological Theory*, Amsterdam, John Benjamins. pp. 35-68.

Boersma, Paul & David Weenink. 2012. *Praat: doing phonetics by computer*. Software package, www.praat.org

Canty, Angel, and Brian Ripley (2012). boot: Bootstrap R (S-Plus) functions. R package version 1.3-4.

Davison, A. C. & Hinkley, D. V. (1997) i. Cambridge University Press, Cambridge.

Hayes, Bruce, Bruce Tesar, and Kie Zuraw (2011) "OTSoft 2.3.1," software package, http://www.linguistics.ucla.edu/people/hayes/otsoft/.

Niyogi, Partha. 2006 *The computational nature of language learning and evolution*. Cambridge, MA: MIT Press.