Class 4, 4/11/13: Trashing the GLA; Maxent Grammars

1. Assignments etc.

- Please hand in your GLA exercise.
- Read:
  - Goldwater, Sharon and Mark Johnson (2003) Learning OT constraint rankings using a Maximum Entropy model. *Proceedings of the Workshop on Variation within Optimality Theory*, Stockholm University, 2003.
  - on course web site
  - please write a less-then-one-page summary, e.g. in bullet points, for Tuesday 4/16.

2. Where are we in the course?

- Exploring the formal side:
  - Grammar frameworks
  - Learning algorithms
- … meant to enable constraint-based analysis of variation systems

WHAT DO WE WANT A FREQUENCY-HANDLING FRAMEWORK TO DO FOR US?

3. Desiderata

- People seem to **frequency-match** (class 2), so the framework should make this possible.
  - The frequency-matching should be fairly **accurate** — like people.
  - Practical advantage: if the frequency matching is *perfect*, you know for sure that additional constraints would not be helpful. Some algorithms[1] sometimes get pretty close to perfect.
- The framework should be **restrictive** (Kie, last time). Example: just assigning a probability to every possible discrete ranking:
  - says less about what languages can be
  - produces a colossal search space, raising questions of realism (cf. Gaja Jarosz's searching through all *n*! rankings (http://pantheon.yale.edu/~gjs42/research.html)
- The framework should come with a **reliable learning algorithm**, since people tend to learn variation accurately.

4. The concept of the *objective function*

- It is in contrast to **process-based** learning — e.g., the GLA:
  - "The learned grammar is *whatever is produced by following this procedure*."

[1] I.e., maxent with a good search procedure, as in the Maxent Grammar Tool.

- An objective function is some number you can calculate from a grammar and the learning data, expressing some property of grammars deemed valuable.
- Learning with an objective function:
  - The grammar you seek to learn is the one that maximizes the function.
  - It then becomes a secondary, primarily practical question how to find it.
- Eloquent defenses of the objective function concept:
  - Sharon Goldwater's Ph.D. dissertation[2]
  - An actual practice demo by Jarosz, below.

5. Gaja Jarosz's argument for an objective function

- This very nice paper:
  - 2009 Gaja Jarosz. Restrictiveness in Phonological Grammar and Lexicon Learning. In *Proceedings of the 43rd Annual Meeting of the Chicago Lingusitic Society*. http://pantheon.yale.edu/~gjs42/files/2009_CLS.pdf

  argues that there are cases where sensible ranking biases will not get you where you need to be. An objective function works better.

PROBLEMS WITH THE GLA

6. Two cases

- one published, from Pater (2008)
- one concocted by me, based on my experience in using GLA

JOE PATER'S NOTORIOUS GARDEN PATH

7. Source

- Pater, Joe. 2008. Gradual learning and convergence. *Linguistic Inquiry* 39/2. 334-345. '
  - http://people.umass.edu/pater/pater-gradual-converge.pdf

[2] Nonparametric Bayesian Models of Lexical Acquisition. Sharon Goldwater. Ph.D. Dissertation, Brown University, 2006. http://homepages.inf.ed.ac.uk/sgwater/papers/thesis_1spc.pdf

8. Tableau

- Here is Pater's clever scheme:

			CON1	CON2	CON3	CON4	CON5	CON6
Form1	Winner1	1		1				
	Loser1		1		1			
Form2	Winner2	1			1			
	Loser2			1		1		
Form3	Winner3	1				1		
	Loser3				1		1	
Form4	Winner4	1					1	
	Loser4					1		1
Form5	Winner5	1						1
	Loser5						1	

- This has no variation at all, and is easily learned by a long-standing simple no-free-variation ranking algorithm (Tesar and Smolensky 1993). Correct (and unique) ranking is:

  CON1 >> CON2 >> CON3 >> CON4 >> CON5 >> CON6

9. The GLA freaks out

- … and fails utterly. A sample run on OTSoft:

```
1. Ranking Values Found

Con4     29,451.851
Con5     29,451.316
Con3     29,451.209
Con2     29,447.919
Con6     29,447.813
Con1     14,747.356

2. Matchup to Input Frequencies


   /Form1/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner1     1.000   0.877     200163      87676
   Loser1      0.000   0.123                 12324

   /Form2/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner2     1.000   0.604     199879      60440
   Loser2      0.000   0.396                 39560

   /Form3/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner3     1.000   0.579     199899      57910
   Loser3      0.000   0.421                 42090

   /Form4/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner4     1.000   0.589     200018      58863
   Loser4      0.000   0.411                 41137
```

```
   /Form5/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner5     1.000   0.891     200041      89106
   Loser5      0.000   0.109                 10894
```

10. There *is* a GLA grammar that works

Just take the ranking above and separate everything by 20 or so.

11. Pater's trick

Almost every winner-preferrer competes with two loser-preferrers, one of which has to be ranked high, the other low.
GLA lacks the foresight to know which one is which …

12. Magri's repair

- Reference:
  - Magri, Giorgio. 2012. Convergence of error-driven ranking algorithms. *Phonology*, 29.2: 213-269.
- He tweaks the GLA, dividing up the amount of promotion available when there are multiple winners.
- This solves the Pater problem. In my OTSoft implementation:[3]

```
1. Ranking Values Found

Con1     107.333
Con2     94.934
Con3     83.571
Con4     72.181
Con5     61.070
Con6     50.128

2. Matchup to Input Frequencies

   /Form1/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner1     1.000   1.000     200313      99997
   Loser1      0.000   0.000                     3

   /Form2/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner2     1.000   1.000     200028      99995
   Loser2      0.000   0.000                     5

   /Form3/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner3     1.000   1.000     200412      99997
   Loser3      0.000   0.000                     3

   /Form4/   Input Fr. Gen Fr.  Input #     Gen. #
   Winner4     1.000   1.000     199699      99995
   Loser4      0.000   0.000                     5

   /Form5/   Input Fr. Gen Fr.  Input #     Gen. #
```

[3] From the GLA menu, under Learning Schedule, select Magri Update Rule.

```
Winner5      1.000   1.000     199548      99996
Loser5       0.000   0.000                     4
```

- Magri shows his algorithm is a very sensible move: in the standard GLA, the correct solution is *not even in the search space* (set of possibilities considered), and in his algorithm of course it is.
- Yet even Magri's improvement is "dead reckoning"; no objective function.

TROUBLE FOR GLA II: ENDLESS DESCENTS INTO HELL

13. The "descent into hell" phenomenon

- Practical use of the GLA often shows certain constraints descending into hell — super-low ranking values, with no bottom in sight as learning continues.
- In principle, this might not be a problem — if the descending constraints are not needed.
- But here is, I think, a realistic case where things go wrong.

14. My nightmare scenario (hopefully realistic)

Inputs	*Candidates*	*Freq.*	PREFER A	PREFER B	PREFER C	PREFER D
Frequent-C	A with C	60		*	*	
	B with C	40	*			
Frequent-no C	A no C	50		*		
	B no C	50	*			
Rare input	C	2				*
	D	8			*	

- Fundamental choice is between A and B candidates, with near-tie of PREFER A and PREFER B.
  - The inputs for this competition happen to be frequent.
- Assume projection-from-the-lexicon, as in many studies of the Law of Frequency Matching.
- Through *sheer bad luck*, a minor constraint PREFER C happens to mismatch the lexicon — penalizing A candidates, modestly, in an area where they are preferred — hence 60/40 where C is present, 50/50 where it is not.
- PREFER C also helps choose the output frequency for a rare class of inputs, competing solely with PREFER D.

15. A typical learned grammar

```
1. Ranking Values Found

Prefer A     100.211
Prefer B     99.789
Prefer D     -2,769.433
Prefer C     -4,611.967
```

2. Matchup to Input Frequencies

/Frequent-C/	Input Fr.	Gen Fr.	Input #	Gen. #
A / C	0.600	0.557	143093	55719
B / C	0.400	0.443	95088	44281

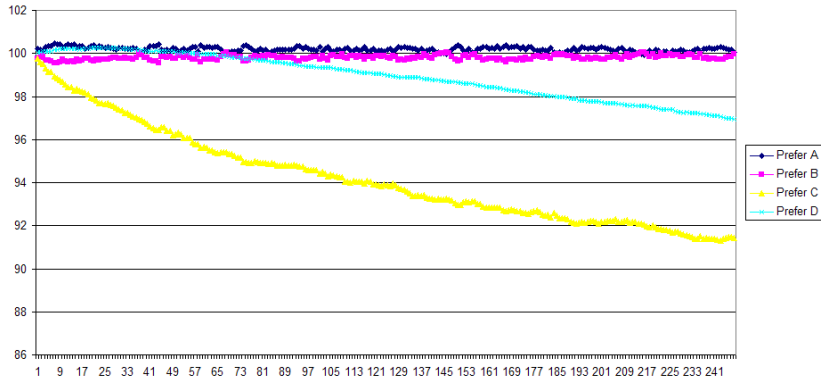
/Frequent-no C/	Input Fr.	Gen Fr.	Input #	Gen. #
A / no C	0.500	0.557	119229	55719
B / no C	0.500	0.443	118881	44281

/Rare input/	Input Fr.	Gen Fr.	Input #	Gen. #
D	0.800	1.000	19021	***100000**
C	0.200	0.000	4688	

- Good aspect of the grammar
  - The overall class of frequent inputs is well-matched, compromising the 50/50 and 60/40 frequencies at about 55/55.
  - i.e., PREFER A and PREFER B are in a good, data-matching relationship.
- Bad aspect of the grammar
  - PREFER C is far too high relative to PREFER D — the free variation for the third input is wiped out.

16. What causes this to happen?

- Exposure to the frequent inputs produce endless little adjustments of PREFER A and PREFER B — jiggling, to keep the output frequency near 55/45.
- PREFER C cannot improve the fit of the Frequent class of inputs, for it prefers the less frequent outcome — it is a passive participant in the A/B competition.
- As such, it gets demoted more often than promoted.
- The C/D competition is on a rare form — not enough "C for D" errors for PREFER D to catch up with PREFER C on its downward course. ("Wait up! Wait up!")
  - A "slow motion" graph of ranking value changes (plasticity set at just .01 throughout) shows this happening.



17. This is a failure of the learning algorithm, not the framework

- There *is* a Stochastic OT grammar that works perfectly well.

- I made up the ranking values myself, consulting the chart at http://www.linguistics.ucla.edu/people/hayes/GLA

```
1. Ranking Values Imposed

Prefer A     100.360
Prefer B     100.000
Prefer D     12.380
Prefer C     10.000
```

  - i.e. pairwise differences of .36 and 2.38, per chart

```
2. Matchup to Input Frequencies

 /Frequent-C/    Input Fr. Gen Fr.  Gen. #
   A / C          0.600   0.551    55115
   B / C          0.400   0.449    44885

 /Frequent-no C/ Input Fr. Gen Fr. Gen. #
   A / no C       0.500   0.551    55115
   B / no C       0.500   0.449    44885

 /Rare input/    Input Fr. Gen Fr. Gen. #
   D              0.800   0.799    79938
   C              0.200   0.201    20062
```

18. Magri's repair doesn't help

```
1. Ranking Values Found

Prefer D        3.470
Prefer A     -835.090
Prefer C     -836.508
Prefer B     -836.623
```

- His system demotes more widely, sending *three* constraints to hell.
- The data match seems to be worse; i.e. it
  - Fails on the rare input
  - doesn't get the frequent inputs either

```
2. Matchup to Input Frequencies

   /Frequent-C/ Input Fr. Gen Fr.  Input #     Gen. #
   A / C          0.600   0.555     285622      55456
   B / C          0.400   0.445     190595      44544

  /Frequent-no C/ Input Fr. Gen Fr.  Input #     Gen. #
   A / no C       0.500   0.705     237816      70520
   B / no C       0.500   0.295     238180      29480

   /Rare input/ Input Fr. Gen Fr.  Input #     Gen. #
   D              0.800   1.000      38102     100000
   C              0.200   0.000       9685
```

19. No problem in maxent (to be covered shortly)

```
1. Constraints and weights

   1.079 Prefer A
   0.879 Prefer B
   0.000 Prefer C
   1.386 Prefer D

Input       Candidates  Input frequencies  Input proportions  Predicted probabilities
Frequent-C    A / C           60                 0.600              0.550
              B / C           40                 0.400              0.450

Input          Candidates  Input frequencies  Input proportions  Predicted probabilities
Frequent-no C   A / no C          50                 0.500              0.550
                B / no C          50                 0.500              0.450


Input       Candidates  Input frequencies  Input proportions  Predicted probabilities
Rare input      C               2                 0.200              0.200
                D               8                 0.800              0.800
```

20. A possible lesson?

- The "descent into hell" cases is exactly what is likely to happen if you search, guided by hope and common sense, but not by an objective function.

INTRO. TO MAXENT GRAMMARS

21. Two approaches to probability in grammars

- Be "uncertain" about the actual content of the grammar (Kie)
  - Anttilan stratal grammars
  - Stochastic OT
  - (later: Noisy Harmonic Grammar)
- The content of the grammar is certain, but it outputs probability distributions over candidates.
  - This is the approach of maxent grammars

22. Where and how to run a maxent grammar

- I believe you can do it in Praat but I haven't tried it.
- In OTSoft: it's one of the options on the main interface.
- Maxent Grammar Tool: www.linguistics.ucla.edu/people/hayes/MaxentGrammarTool
  - Platform independent (written in Java); click and use
  - Programmed by Colin Wilson with interface by Ben George
  - Very fast and accurate
  - Uses OTSoft input format (you have to save it as tab-delimited text)
  - One drawback: you *always* have to use a Gaussian prior; mentioned later

EXCURSUS: HARMONIC GRAMMAR

23. Maxent grammar is a variety of *Harmonic Grammar*

- Explored by Paul Smolensky in the 1990's as a way of doing "analytically aware" connectionism.
- Obscured by orthodox OT for many years
  - OT also introduced the GEN + EVAL architecture, which is tremendously useful no matter whether your EVAL is as in OT or is
- Now making a comeback at UCLA, UMass, elsewhere

24. Varieties of harmonic grammar

- A **harmonic (OT) grammar** is one where
  - every constraint has a weight (real number, usually constrained to be nonnegative)[4]
  - for each candidate you compute the **dot product** of violations and weights
  - Meaning: for each candidate/constraint, multiply violations times weights, then sum of constraints — an intuitive "penalty score".

    Socrates: give dot product for these:

1.1	2	8.3	weights
2	1	1	violations

- The varieties are determined by how they calculate winners.
  - **Nonstochastic harmonic grammar**: candidate with lowest penalty wins
  - **Noisy harmonic grammar**: follow this proportional analogy:

    Noisy harmonic grammar : nonstochastic harmonic grammar
    Stochastic OT : classical OT

    i.e. jiggle the weights a bit at random, each "evaluation time". More on this later.
  - **Maxent**: use a formula to compute a probability distribution

25. Some references for harmonic grammar

- Legendre, Géraldine; Miyata, Yoshiro; & Smolensky, Paul. (1990). Harmonic Grammar: A formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. Report CU-CS-465-90. Computer Science Department, University of Colorado at Boulder.
- Smolensky, Paul, and Geraldine Legendre. 2006. *The Harmonic Mind*. Cambridge: MIT Press. (summarizing work of two decades, including early Harmonic Grammar)

[4] An alternative convention makes all the weights non-positive; it's all a matter of where you put the minus sign in the calculations.

- Pater, Joe (2009) Weighted constraints in generative linguistics. *Cognitive Science* 33: 999-1035.
- Potts, C., J. Pater, K. Jesney, R. Bhatt, and M. Becker (2009) Harmonic Grammar with Linear Programming: From linear systems to linguistic typology. *Phonology* 27:77-118.

26. Harmonic grammar's most salient prediction

- Constraint ganging:
  - Two weaker constraints can gang up to defeat a stronger constraint.

27. A real-life example of ganging: Japanese consonant voicing

- Source: Shigeto Kawahara (2006) A faithfulness ranking projected from a perceptibility scale: the case of [+voice] in Japanese. *Language* 82:536-574.
- Historically, Japanese didn't allow:
  - voiced obstruent geminates, like [bb], [dd]
  - two voiced obstruents in stem like *[baba] — "Lyman's Law".

No devoicing

webbu	‘web’
sunobbu	‘snob’
habburu	‘Hubble’
kiddo	‘kid’
reddo	‘red’
heddo	‘head’
suraggaa	‘slugger’
eggu	‘egg’
furaggu	‘flag’

bagii	‘buggy’	bogii	‘bogey’
bobu	‘Bob’	bagu	‘bug’
dagu	‘Doug’	daibu	‘dive’
daijamondo	‘diamond’	doguma	‘dogma’
giga	‘giga- (prefix)’	gaburieru	‘Gabriel’
gibu	‘give’	gaidansu	‘guidance’

Devoicing

gepperusu	‘Göbbels’
gutto	‘good’
betto	‘bed’
doretto	‘dreadlocks’
dettobooru	‘dead ball (baseball term)’
batto	‘bad’
deibitto	‘David’
dokku	‘dog’
bakku	‘bag’
dorakku	‘drug’
bikku	‘big’

28. Non-stochastic Harmonic Grammar analysis

- Goal: find weights for
  - IDENT(voice)
  - LYMAN'S LAW[5] = *[−sonorant,+voice] … [−sonorant,+voice]
  - *VOICED GEMINATE

  that will derive Kawahara's pattern.
- Inputs and candidates:
  - /bobu/ → ✓bobu, *bopu, *pobu, *popu
  - /webbu/ → ✓webbu, *weppu
  - /doggu/ → ✓dokku, *doggu, *dokku, *tokku

- I made a little spreadsheet, letting me experiment with weights.
  - I used the SUMPRODUCT() function (a.k.a. dot product) to compute harmony scores.
  - I messed around with the weights by hand until I had something that picked all winners.

			IDENT(VOICE)	LYMAN'S LAW	*VOICED GEMINATE	
		Weights:	***4***	***3***	***2***	*Harmony*
bobu	☞ bobu	1		1		*3*
	*bopu		1			*4*
	*pobu		1			*4*
	*popu		2			*8*
webbu	☞ webbu	1			1	*2*
	*weppu					*4*
doggu	☞ dokku	1	1			*4*
	*doggu			1	1	*5*
	*toggu		1		1	*6*
	*tokku		2			*8*

- What's crucial about 4 3 2 is that neither Lyman's Law nor *Voiced Geminate alone is powerful enough to overcome Ident(voice); but together they are.
- For software (using the math known as Linear Programming) that reliably finds working weights for non-stochastic Harmonic Grammar, visit OTHelp at UMass: http://people.umass.edu/othelp/
- Real Japanese is stochastic; this is an idealized example.
- I suspect that most ganging occurs in stochastic conditions, a puzzle we should think about.[6]

[5] It's really Lyman's Constraint, but nobody calls it that…
[6] See Edward Flemming (ms.) Conflict resolution in phonetics and phonology, ms.

29. How pervasive is ganging?

- The debate over EVAL (OT/Harmonic Grammar) partly hinges on this.
- Potts et al. (*Phonology* 2010) give an elaborate case from Lango, which works very cleanly for them with (nonstochastic) Harmonic Grammar
- Hayes and Wilson (*LI* 2008) point out possible ganging effects in English phonotactics—sounds (e.g. rare [ð] is both a voiced fricative and a dental fricative)
- OT literature is replete with conjoined constraints.
    - For how ganging sometimes, but not always, makes conjoined constraints unnecessary, see Bruce Hayes, Colin Wilson, and Anne Shisko. 2012. Maxent grammars for the metrics of Shakespeare and Milton. *Language* 88: 691-731.

BACK TO MAXENT PER SE

30. References

- Goldwater, Sharon, and Mark Johnson. 2003. Learning OT constraint rankings using a maximum entropy model. Proceedings of the Stockholm Workshop on Variation within Optimality Theory, ed. by Jennifer Spenader; Anders Eriksson, and Osten Dahl, 111–120. Stockholm: Stockholm University Department of Linguistics.
- Wilson, Colin. 2006. Learning phonology with substantive bias: an experimental and computational investigation of velar palatalization. *Cognitive Science* 30:945–982
- Hayes, Bruce and Colin Wilson, (2008) A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry* 39: 379-440.

31. The maxent procedure

- Works input-by-input.
- First, compute the harmony of all candidates for a given input, as above.
- Compute from each one a "maxent value" = *e* to the minus harmony.
- Then total all maxent values and compute each candidate's share.
- This is the predicted probability of the candidate.

32. Back to the Japanese data

- Kawahara followed up his original study with an experiment.
    - Kawahara, Shigeto (2011) Japanese loanword devoicing revisited: A rating study. *Natural Language and Linguistic Theory*.
    - This is a rating study, which confirms the psychological reality of the lexical study.
- Since modeling ratings is tricky, let us simply model the lexical frequencies on which the ratings are (probably, mostly) based.

- Kawahara gives data that could (for pedagogical purposes only) be interpreted as the following percentages of devoicing:

  babba-type words: 57.4
  pabba-type words: 3.7
  baba-type words: assumed near zero

What sort of grammar could generate these numbers?

33. A Maxent grammar analysis

I submitted this file to the Maxent Grammar Tool:

			Ident(voice)	Lyman	*bb
			Ident(voice)	Lyman	*bb
babba	babba	436		1	1
	bappa	574	2		
	pabba	0	1		1
	pappa	0	3		
pabba	pabba	963			1
	pappa	37	2		
		100			
baba	baba	0		1	
	bapa	0	1		
	paba	0	1		
	papa	0	2		

- Weights found:

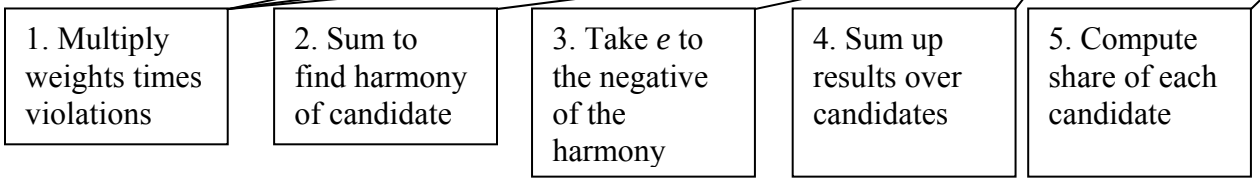
  17.7 IDENT(voice)
  3.5 LYMAN'S LAW
  32.2 *VOICED GEMINATE OBSTRUENT

34. Computing the probabilities from the weights

- We're going to implement the formalism given above in (31).
- The expression "E-12" means "ten to the minus twelfth power", a small number

			Id(vce)	LL	*bb
/babba/	babba	436	0	1	1
	bappa	574	2	0	0
	pabba	0	1	0	1
	pappa	0	3	0	0
/pabba/	pabba	963	0	0	1
	pappa	37	2	0	0
/baba/	baba	1000	0	1	0
	bapa	0	1	0	0
	paba	0	1	0	0
	papa	0	2	0	0
			Weights:		
			17.7	**3.53**	**32.2**

	Input: babba		Id(vce)	LL	*bb	Harmony	e to minus H	Total per input	Share
/babba/	babba	436	0	3.53	32.2	35.73	3.03849E-16	7.26494E-16	0.418
	bappa	574	35.4	0	0	35.4	4.22645E-16		0.582
	pabba	0	17.7	0	32.2	49.9	2.1316E-22		0.000
	pappa	0	53.1	0	0	53.1	8.68886E-24		0.000
/pabba/	pabba	963	0	0	32.2	32.2	1.03685E-14	1.07912E-14	0.961
	pappa	37	35.4	0	0	35.4	4.22645E-16		0.039
/baba/	baba	1000	0	3.53	0	3.53	0.029304916	0.029304957	1.000
	bapa	0	17.7	0	0	17.7	2.05583E-08		0.000
	paba	0	17.7	0	0	17.7	2.05583E-08		0.000
	papa	0	35.4	0	0	35.4	4.22645E-16		0.000



35. Upshot

- The ganging is obtained without adding an extra constraint. (two parameters get three frequencies).
- The actual calculations mimicked the input frequency to 6 decimal places.

Input	Candidate	Observed Freq.	Observed Proportion	Predicted proportion
babba	babba	436	0.431683	0.431685
	bappa	574	0.568317	0.568315
	pabba	0	0	2.89E-07
	pappa	0	0	1.11E-08
pabba	pabba	963	0.963	0.962999
	pappa	37	0.037	3.70E-02
baba	baba	1000	1	0.999999
	bapa	0	0	6.70E-07
	paba	0	0	6.70E-07
	papa	0	0	1.31E-14

36. Why is maxent (= "maximum entropy") so called?

- [Because mathematical people tend to be clueless when inventing terminology.]
- The spirit of the thing: grammar should assign equal probability — sheer randomness — for all cases where data do not tell it otherwise.
  - This randomness means that the grammar is neutral in its commitments where data are not available.
  - Example: rerun the Japanese problem with all zero data frequencies; you get all zero weights and these probabilities:

Input:	Candidate:	Observed:	Predicted:
babba	babba	0	0.25
	bappa	0	0.25
	pabba	0	0.25
	pappa	0	0.25
pabba	pabba	0	0.5
	pappa	0	0.5
baba	baba	0	0.25
	bapa	0	0.25
	paba	0	0.25
	papa	0	0.25

37. A caution about maxent: harmonically-bounded candidates can win!

- See the tiny frequency for /baba/ → [papa] above.
- They never get the highest frequency.
- For a possible case in the real world see
  - Bruce Hayes and Claire Moore-Cantwell. (2011) Gerard Manley Hopkins' sprung rhythm: corpus study and stochastic grammar. *Phonology* 28:235-282.
- One should be cautious in maxent analysis in including the harmonically bounded candidates as well.

THE LEARNABILITY SITUATION FOR MAXENT

38. Maxent learning is based on an objective function

- This is: *maximize the predicted probability of the data under the grammar*.
  - I.e., find the grammar (weight set) for which the predicted probability of the observed data is maximized.
  - This means the predicted probability of the *un*observed data is minimized — which is good!

39. Maxent works well in a variety of respects

- There always is an answer.
- Other than issues involving learning biases — it is unique.
- There are fast algorithms for finding it.

40. How to calculate the predicted probability of the data

- We're actually going to calculate the *log* of the probability.
  - i.e. natural log, base *e* (about 2.71828)
  - Log probability is sometimes jocularly referred to as **plog**
- Here is a tiny enough grammar (two constraints) for us to visualize plog

			Id(voice)	*VtV
		Freq\Weights:	2.197	1.792
ata	ata	60	0	1
	ada	40	1	0
da	da	90	0	0
	ta	10	1	0

- Note the characteristic assignment of minority probability to a harmonically bounded candidate.
- I made a spreadsheet that, for any pair of constraint weights, finds the plog of the data.
- Example: the best weights (learned with the Maxent Grammar Tool) happened to be
  - ID(voice) = 2.197
  - *VtV = 1.792

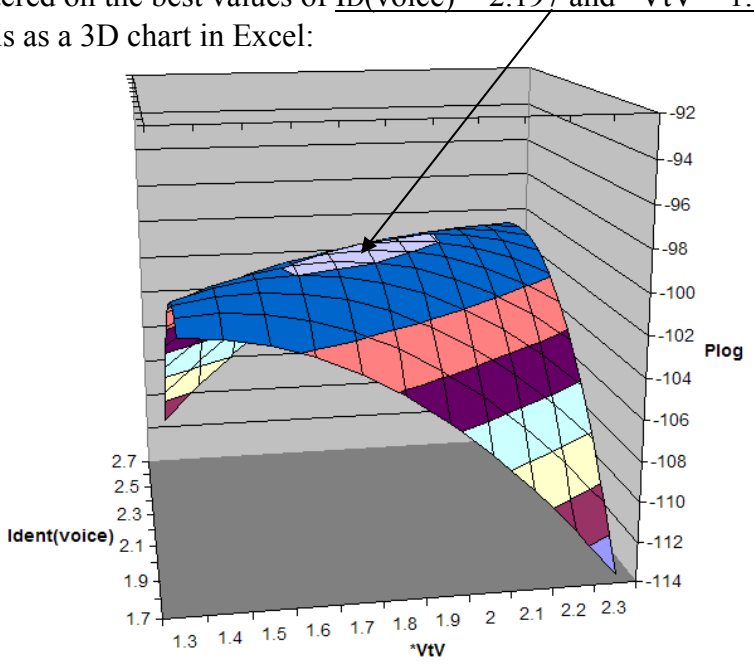
  Let's go through the basic procedure.
- Harmony values (Dot products):
  - ata = 1.792
  - ada = 2.197
  - da = 0
  - ta = 2.197 Check for yourself that these are correct.
- Take *e* to the minus harmony. The Excel expression for this is "EXP(-A1)" for cell A1.
  - ata = $e^{-1.792}$ = 0.166627
  - ada = 0.11136
  - da = 1

  - ta = 0.11136
- Find total for each input:
  - /ata/ = sum for [ata] and [ada] = 0.166627 + 0.11136 = 0.277763
  - /da/ = 1.111136
- Find the share of each candidate for its input:
  - ata = share of [ata]/combined [ata] and [ada] = 0.166627 / 0.277763 = 0.6
  - ada = 0.4
  - da = 0.9
  - ta = 0.1
- Experience reassurance, because these match the training data perfectly.
- Turn these probabilities into logs
  - ata = ln(0.6) = −0.511
  - ada = −0.916
  - da = −0.105
  - ta = −2.303
- Now, remember that to combine probabilities (probability of A and B = P(A) * P(B)), and that multiplication is actually addition when you do logarithms.
- Remember that we have 60 copies of [ata], 40 of [ada], 90 of [da], 10 of [ta].
- So: (60 * −0.511) + (40 * −0.916) + (90 * −0.105) + (10 * −2.303) = **-99.80947**
- This is the plog of the data.
- If this were expressed as a number it would be $4.5009 * 10^{-44}$

41. The predicted probability of the data forms a magnificent dome

- We just calculated the plog using the weights of the very best grammar (told to us by the Maxent Grammar Tool)
- But we want to learn a bit how the Maxent Grammar Tool found these best weights.
- So, let's explore: try a whole bunch of weights in the same region.

- Using Excel, I calculated the predicted probability of the data for a whole bunch of points, centered on the best values of ID(voice) = 2.197 and *VtV = 1.792.
- I plotted this as a 3D chart in Excel:



- This is just a bit of the full dome-shaped structure, whose summit is at the best value.
- There is *only one maximum*, the global maximum at Id(voice) = 2.197 and *VtV = 1.792.
  - technical usage: "the search space is convex"
- When the search space is convex, computer scientists experience joy, and so should we.
  - You'll always find the answer if you just keep climbing uphill.
  - And the answer is unique.
  - This is the informal basis of the *convergence proof* for maxent (i.e., we can always maximize the objective function because of the convexity of the search space).

42. Finding the best weights: followup details

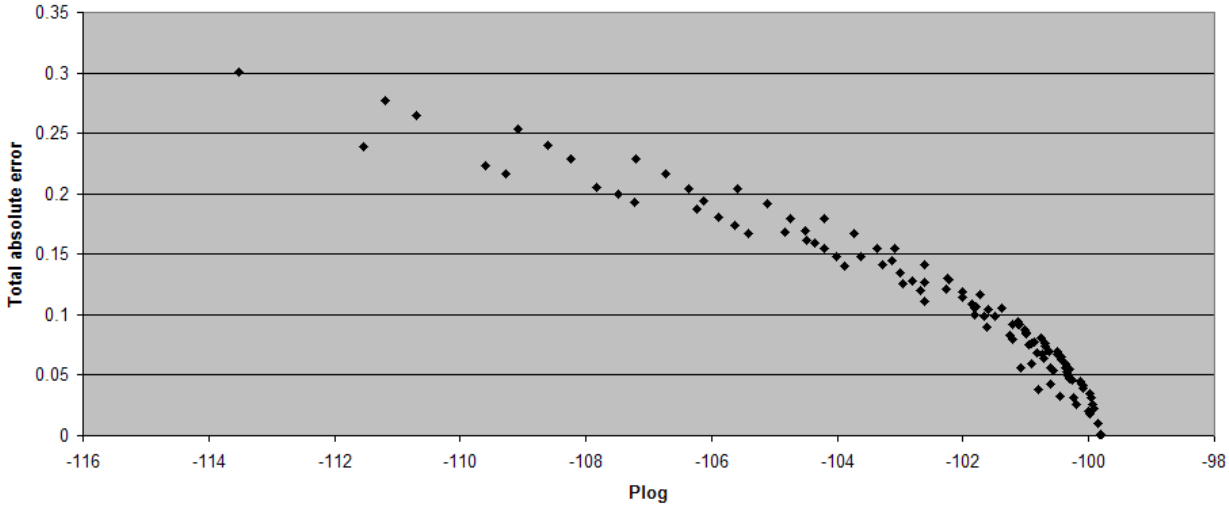
- Usually, one searches for the best weights by starting in some arbitrary point (e.g. (0, 0) for two constraints) and moving *uphill* short distances, over and over.
- OTSoft uses a slow simple search method; MaxentGrammarTool uses a sophisticated fast one.
  - Better search methods make effective guesses about how far to move at each iteration.
- Math not covered here tells you what direction is uphill.
  - Specifically: partial derivatives of plog against each constraint weight
- Socrates: this suggests a way of know when your at the top of the hill — what is it?
- The process generalizes to any number of constraints (but only two can be visualized as a dome).

43. What it's like outside paradise

- In many other learning problems there are *local maxima* — upward bulges in the terrain that do not culminate in the highest point overall.
- Computer scientist curse when local maxima are encountered — moving uphill could just land you on a lower peak, failing to find the global optimum.

44. A last bit: should we trust the objective function of maximizing plog?

- A bit of intuition: for the data that created the 3D chart of (41), I also calculated the *raw summed error* of the various little grammars.
  - i.e. absolute value of predicted vs. observed, for each of the two data points, then summed
- Plotting this against plog, we see informal confirmation that high-plog grammars tend to be accurate:



45. Summing up

- Maxent is a computational paradise for constraint-based linguists, with
  - a clear objective function
  - provable convergence
  - excellent algorithms to search for the weights
- It is affiliated with a theoretical framework that has two properties that are controversial and should be monitored with care.
  - constraint ganging (all harmonic grammar)
  - positive probabilities (never maximum) for harmonically-bounded candidates