Class 6: GLA II; lab

To do for Thursday

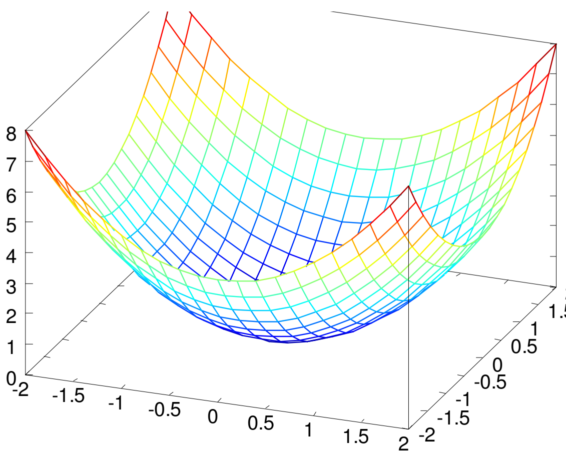
- Reading for Thursday is Coetzee 2009
  - Your reading question: take **one** tableau from your own data where the result is invariable.
  - If you have no data like that, use one of the invariable cases from Anttila's Finnish data.
  - Find weights for the constraints that produce the correct winning candidates.
  - It will probably help to implement the tableau in Excel so that you can easily change the weights until you find good ones.
  - Turn in a tableau like the one on Coetzee's p. 274 (on the right).
- Reading for Friday is section 4.7 of Martin 2007.
  - If you want to get a head start, your reading question will be: Why does adding the "Gaussian prior" to the expression that MaxEnt is maximizing help avoid overfitting?
- Decide whether you want to give a brief presentation on your data in the last class session. On Thursday in class we'll make up the schedule for the last day. Based on what you've turned in and our conversations, I think everyone should make a presentation!

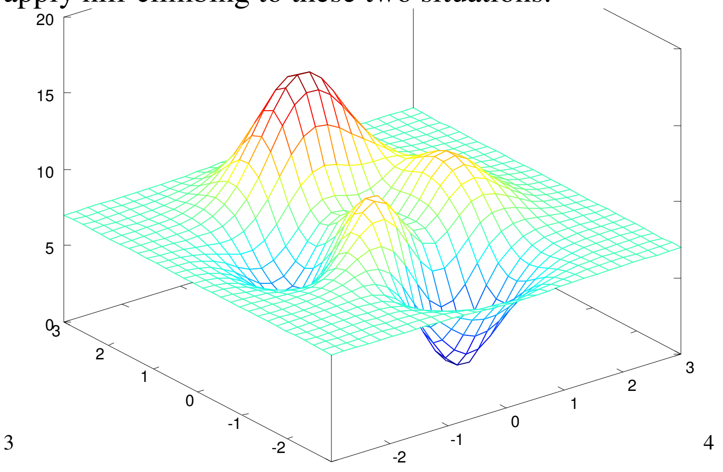
Overview: Some computational issues with the GLA; proposed solutions. Then a lab to practice using the GLA.

1 Convergence

- Suppose for simplicity that our job is to assign numbers (like ranking values) to just two constraints (horizontal axes in pictures below)
  - ...and minimize some error rate that we can quantify (vertical axis).
- The graphs below represent the error rate that we will find for each pair of ranking values
  - But, our learning algorithm doesn't have direct access to these pictures
- Typical "hill-climbing" approach:
  - Choose a pair of ranking values at random
  - Determine which direction is "downhill" from there on the vertical axis
    - A simple way to do this is to sample a ring of nearby points[1]
  - Take a step in the downhill direction[2]
  - When the error rate stops going down, stop.

○ Let's think about what will happen if we apply hill-climbing to these two situations.





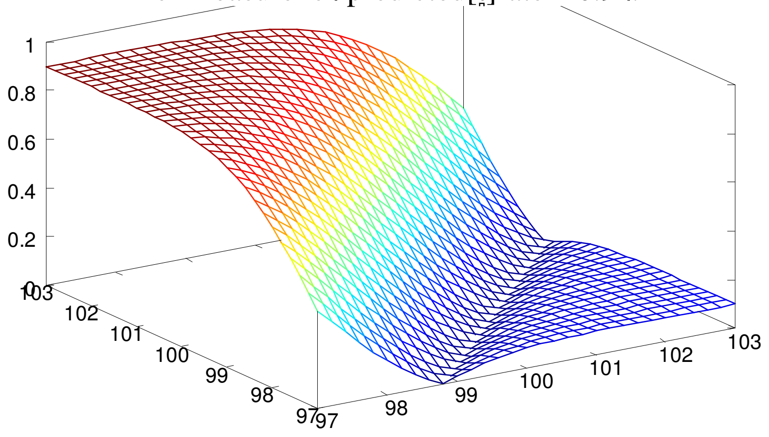
[1] In some cases, the algorithm knows enough about the error function to use calculus to find which direction is downhill.
[2] The size of the step can be constant or it can depend on how steep the error slope is.

- Ideally, we want a situation as on the left:
  - There's exactly one point from which every direction is uphill.
  - A learning problem like this is called "convex"—computer scientists like convexity

- If a learner keeps going downhill and never stops (except maybe because we tell it to stop after 10,000 steps), it has *failed to converge*.
  - o What would the error function have to look like for that to happen?
- The learner could also fail to converge because it doesn't always go downhill
- If a learner converges but didn't find the lowest point, we say that it got stuck in a *local minimum* or *local optimum* instead of finding the *global optimum*.

By the way, here's the plot for our (th) 'thick' example.

Error measure is | predicted[t̪]rate – 0.9 |:



[5]

2 Gradual Learning Algorithm?

- For the original GLA, there was no proof of convergence or, if it converges, of finding the global optimum.
- Pater 2008 pointed out a type of case that is problematic (the "credit problem").

Suppose 6 constraints: NOCODA, ONSET, *VOICEDOBSTRUENT, *ɛ#, MAX-C, DEP-C

o Determine the constraint ranking for this language and fill in the tableaux

/da/						
☞ *a* da						
b a						
/lob/						
c lob						
☞ *d* lo						
/tɛf/						
☞ *e* tɛf						
f tɛ						
/kɛ/						
g kɛʔ						
☞ *h* kɛ						

[3] Octave commands: [x,y]=meshgrid([-2:.2:2]); Z=x.^2+y.^2; mesh(x,y,Z) . Based on www.mathworks.com/help/techdoc/visualize/f0-18164.html
[4] [X,Y,Z]=peaks(30); mesh(X,Y,Z+7) . Based on www.mathworks.com/help/techdoc/ref/surfc.html
[5] [x,y]=meshgrid([97:0.2:103]); Z=abs(0.9-normcdf(x-y,0,sqrt(2))); mesh(x,y,Z).

- Let's think about what the GLA will do for each of these tableaux when it makes a mistake during learning

⇒ Now let's try it in OTSoft (I have an input file prepared)

3 The Revised GLA: Magri to appear

- Suppose current plasticity is 1.
- Suppose the learner makes some errors like these.
    - I've included the amount by which regular GLA will change each ranking value

	/inputA/	constraint1	constraint2	constraint3	constraint4
desired (adult) winner	candA1	* (-1)			
grammar-in-progress's current winner	candA2		* (+1)		

	/inputB/	constraint1	constraint2	constraint3	constraint4
desired (adult) winner	candB1	* (-1)			
grammar-in-progress's current winner	candB2		* (+1)	* (+1)	

	/inputC/	constraint1	constraint2	constraint3	constraint4
desired (adult) winner	candC1	* (-1)			
grammar-in-progress's current winner	candC2		* (+1)	* (+1)	* (+1)

- Magri's revised GLA says instead that the winner-preferring constraints have to *share the credit*:

	/inputA/	constraint1	constraint2	constraint3	constraint4
desired (adult) winner	candA1	* (-1)			
grammar-in-progress's current winner	candA2		* (+1)		

	/inputB/	constraint1	constraint2	constraint3	constraint4
desired (adult) winner	candB1	* (-1)			
grammar-in-progress's current winner	candB2		* (+0.5)	* (+0.5)	

	/inputC/	constraint1	constraint2	constraint3	constraint4
desired (adult) winner	candC1	* (-1)			
grammar-in-progress's current winner	candC2		* (+0.33)	* (+0.33)	* (+0.33)

- What if there is more than one constraint that prefers the losing candidate?
    - We still demote each loser-preferring constraint by 1
    - But we promote each winner-preferring constraint once for every loser-preferring constraint
    - In other words, if *l* is the number of loser-preferring constraints and *w* is the number of winner-preferring constraints,
        - Demote each loser-preferrer by 1
        - Promote each winner-preferrer by *l*/*w*

- See Magri's paper for the proof that this algorithm does converge (not easy reading!)

- o OTSoft can't implement Magri's algorithm (yet), but let's think through how this would help with Pater's case.

See also this web page of Paul Boersma's: www.fon.hum.uva.nl/paul/gla/, for a bibliography and discussion of when and how often GLA and other error-driven algorithms will fail.

Part II: lab

4 Experiment to see what the menu option "Present data to GLA in exact proportions" does

- Download any OTSoft-formatted input file from the class webpage (Dutch, Navajo, Finnish...whatever you like), or use your own data
- Run the GLA on it in OTSoft
  - run the learner with menu option `Options > Present data to GLA in exact proportions` selected
  - look at the results
  - run it with that option not selected; look at the results
- Can you see what the learner is doing differently?
  - Hint: look at the "input #" column in the results

5 Track ranking values over time

- For this one, use your own data
- Run the GLA on it, using whatever parameter settings you like, except...
  - Choose the option `Options > Print file with history of ranking values`
- Inside the folder that OTSoft has created (its name should include the name of the input file), there is now a file called `YourInputFileName`**`History`**`.xls`
- Open it in Excel and take a look
- Make a graph of the ranking values over time in Excel
- This part will take some time:
  - Choose about 25 timesteps from your `***`**`History`**`.xls` file.
  - Choose 3 or 4 tableaux from the input file that you're interested in
  - Make an Excel file that looks like this (except with all 26 rows):

timestep	rate of adjacent harmony	rate of distal harmony	rate of adj harmony in SHALA	rate of distal harmony in KISHALA
0				
1				
2				
...				
24				
25				

  - For each timestep, run the GLA in OTSoft...
    - Use `Initial rankings > Edit file for fully customized initial ranking values` to set the ranking values to those of the current timestep
    - For timestep 0, use your actual initial rankings values (e.g., all 100s)
    - Set `Number of times to go through forms` to 0 (no learning)

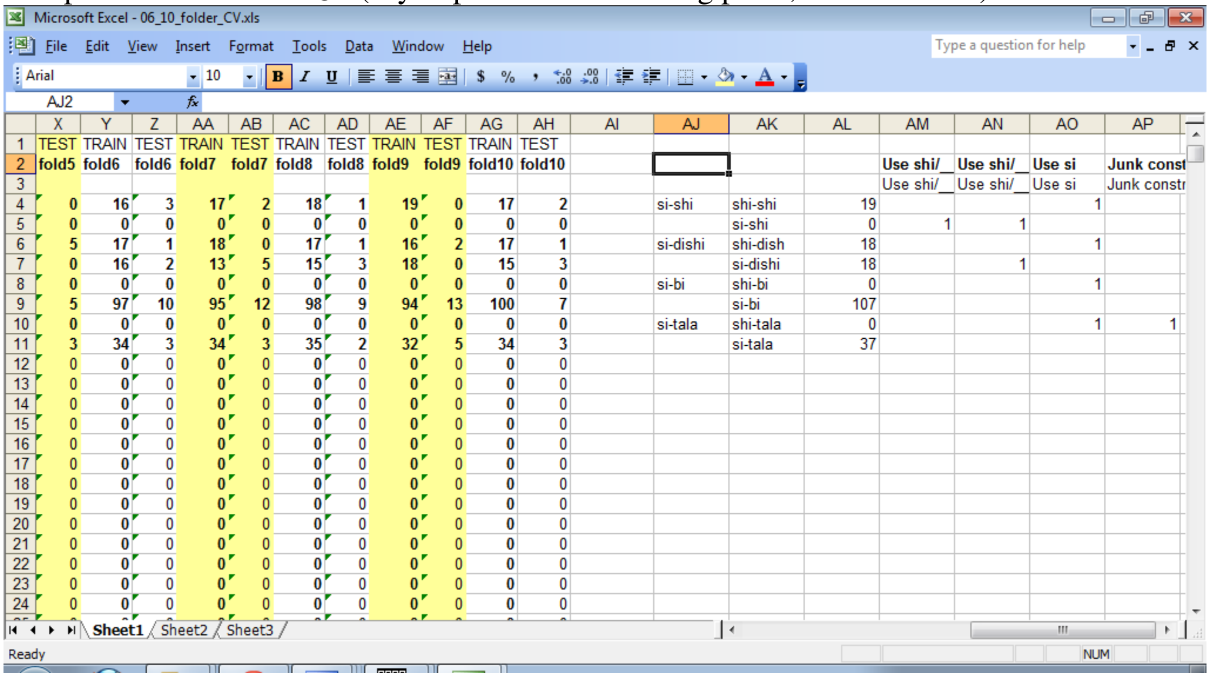
- Use your results file to fill in, for each timestep, how often the candidate you're interested in won:

timestep	rate of adjacent harmony	rate of distal harmony	rate of adj harmony in SHALA	rate of distal harmony in KISHALA
0	0.501	0.253	0.453	0.135
...				

- Make a graph in Excel of your learner's outputs change over time.
- You probably will want to save these files, for possible later use in a presentation.

6 Model comparison with cross-validation

- This one will also take some time
- If you're a programmer, you'll see that you can easily write a program for yourself to automate this.
- But for now, so that everyone can try it, I've created an Excel file to help you do 10-fold cross-validation.
- Download and open the Excel cross-validation spreadsheet from the class webpage.
- Make 20 copies of the OTSoft input file that contains your data.
- Add to the file names something like "train1", "test1", "train2", "test2"... "train10", "test10"
- Open the original OTSoft input file and paste its contents into the cross-validation spreadsheet in **cell AJ2** (if you paste into the wrong place, it won't work):



	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP
1	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST								
2	fold5	fold6	fold6	fold7	fold7	fold8	fold8	fold9	fold9	fold10	fold10					Use shi/__	Use shi/__	Use si	Junk const
3																Use shi/__	Use shi/__	Use si	Junk constr
4	0	16	3	17	2	18	1	19	0	17	2		si-shi	shi-shi	19			1	
5	0	0	0	0	0	0	0	0	0	0	0			si-shi	0	1	1		
6	5	17	1	18	0	17	1	16	2	17	1		si-dishi	shi-dish	18			1	
7	0	16	2	13	5	15	3	18	0	15	3			si-dishi	18		1		
8	0	0	0	0	0	0	0	0	0	0	0		si-bi	shi-bi	0			1	
9	5	97	10	95	12	98	9	94	13	100	7			si-bi	107				
10	0	0	0	0	0	0	0	0	0	0	0		si-tala	shi-tala	0			1	1
11	3	34	3	34	3	35	2	32	5	34	3			si-tala	37				
12	0	0	0	0	0	0	0	0	0	0	0								
13	0	0	0	0	0	0	0	0	0	0	0								
14	0	0	0	0	0	0	0	0	0	0	0								
15	0	0	0	0	0	0	0	0	0	0	0								
16	0	0	0	0	0	0	0	0	0	0	0								
17	0	0	0	0	0	0	0	0	0	0	0								
18	0	0	0	0	0	0	0	0	0	0	0								
19	0	0	0	0	0	0	0	0	0	0	0								
20	0	0	0	0	0	0	0	0	0	0	0								
21	0	0	0	0	0	0	0	0	0	0	0								
22	0	0	0	0	0	0	0	0	0	0	0								
23	0	0	0	0	0	0	0	0	0	0	0								
24	0	0	0	0	0	0	0	0	0	0	0								

- Your data have been divided into 10 random slices of approximately equal size.
- In your "train1" file, change the frequencies to the numbers from the "TRAIN fold1" column (column 0), and similarly for the other 19 files.
- Now you are ready to compare different GLA parameter settings for your data

- Run the GLA on your 10 "train" files, using the same parameter settings.
  - This will give your 10 results files with final ranking values
  - For each "test" file, run the GLA with the initial ranking values from the corresponding "train" file results, but set "Number of times to go through forms" to 0
  - In a spreadsheet or some other file, write down the accuracy for each "test" file

- Now run the GLA again on the **same** 10 "train" files, but change something in the parameter settings
  - You could change the initial ranking values, or change the plasticity schedule, or change the constraints...
  - It's better to change only one thing—if you want to try another change, do this procedure instead

The file where you keep your results might look like this:

	average error per candidate			
fold	initial values = 100, initial plasticity = 2 final plasticity = 0.02 5000 learning trials	initial values = 50 for faith, 100 for mark initial plasticity = 2 final plasticity = 0.02 5000 learning trials	initial values = 100, initial plasticity = 1 final plasticity = .01 5000 learning trials	initial values = 100, initial plasticity = 2 final plasticity = 0.02 10000 learning trials
1	0.23121	0.339097	0.10259	0.313654
2	0.273329	0.34808	0.172439	0.265392
3	0.203475	0.372748	0.135748	0.315946
4	0.208444	0.338935	0.161818	0.285827
5	0.268028	0.348117	0.194969	0.276292
6	0.242403	0.331804	0.186735	0.310904
7	0.296243	0.316736	0.160914	0.290142
8	0.2636	0.335112	0.109037	0.311412
9	0.283604	0.377543	0.152997	0.340538
10	0.287223	0.365562	0.107381	0.253428
mean	**0.255756**	**0.347373**	**0.148463**	**0.296354**

I was going to have us do this on the Finnish data, with different sets of constraints, but now I think it makes more sense for you to work on your own data.

Save all your output files!! They will be useful for your presentation.

Next time: A quite different type of quantitative OT: constraint weighting—different because it can't be translated into a probability distribution over classic OT rankings!

References

Magri, Giorgio. to appear. HG has no computational advantages over OT: towards a new toolkit for computational OT. *Linguistic Inquiry.*
Pater, Joe. 2008. Gradual Learning and Convergence. *Linguistic Inquiry*.