J. Pater, JJM/JP Grant Meeting, 3/25/09

Learning in Praat: the basics

1. File structure

Learning in Praat requires two files: a grammar file, and a distribution file.

The grammar file contains the tableaux, specifies initial weights, learning rate (plasticity), and the grammar model (evaluation mode). The distribution file specifies the distribution over the learning data that will be provided to the learner.

To do hidden structure learning, you need to use a special grammar file type ("OTGrammar 2")

With Praat's grammar file structure it's easy to make mistakes in putting in violation marks: this is why OT-Help has Praat file output (which gives a "collection" file containing both the grammar and distribution file).

Another common mistake one can make with this grammar file structure is to add candidates, tableaux or pairs of learning data without changing the number at the top of the relevant list. Praat returns a somewhat oblique error message, so this error can also be hard to spot.

For the distribution file, your numbers specify the distribution that Praat will use in randomly sampling learning data: giving all of your data a weight of 1 is the same as giving them a weight of 100.

2. Learning

2.1 Changing the grammar

Before learning, you may want to adjust some aspect of you grammar file. To do so, select it, and then choose one of the "modify" options:

Modify ranking: good for setting all the weights to the same initial position. To change individual weights, it's usually easier to do this by looking at the grammar in the edit window, clicking on the constraint name, and choosing edit (unless you know the number of the constraint).

Modify behaviour: this is where you can change the type of grammar, under "set decision strategy":

Optimality Theory: OT with tied constraints (this is how EDCD works too)
Harmonic Grammar: Plain HG with positive and negative weights
Linear OT: HG with any negative weight changed to zero
Exponential HG: HG with exp(w), where w is the weight
MaximumEntropy: Probabilistic HG, as in Goldwater and Johnson , Jaeger, Wilson
Positive HG: HG with any weight < 1 changed to 1
ExponentialMaximumEntropy: Probabilistic HG, with exp(w), where w is the weight

2.2 Learning
To get the learning menu, you need to select a grammar and a distribution file. You then want to click learn...

(Note that "get positive weights" is the Potts *et al.* Linear Programming application)

I usually reset most of the standard settings (except "symmetric all", which automatically switches between OT-GLA and HG-GLA as appropriate - see Boersma and Pater 2008):

Evaluation noise: should be 0 for Maximum Entropy learning, 0 for categorical OT or HG, 0.2 for Noisy Exponential HG
Initial Plasticity: some number smaller than 1.0 (usually 0.01, or 0.001 for Exp-HG)
Replications per plasticity: depends on the problem – may need to be as high as 100,000 or 1,000,000 to get fully accurate learning of variable patterns, or very low to see stages or incomplete learning
Plasticity decrement: Plasticity is multiplied by this number in each "new plasticity" (learning stage). I usually just set it to 1.
Number of plasticities: also 1.
Rel. Plasticity Spreading: noise on plasticity - I set it to 0
Number of chews: number of times the learner processes each datum - 1

2,3 Seeing the result
To see the final grammar, select the grammar file, and press edit.

To see what it does on repeated evaluations:

1. Choose the "evaluate" option from the edit menu inside the edit window (the keyboard shortcut can be fun if you have noise)

2. Choose both the grammar and the pair distribution, and choose: "Get fraction correct"

3. Choose the grammar, To output distributions,choose the distributions, convert to table, and press edit (undocumented and a little bit of pain - luckily there is scripting...).

Also, the grammar file can be saved and opened in excel. When you are working with Maximum Entropy or non-noisy HG, you can then calculate the results yourself in a spreadsheet based on the weights (remember to use *exp*(w) when appropriate!)

3. Scripting

The best thing about learning in Praat is that it's fully scriptable. Anything you do can be added to a script by using "Paste History". You can also have variables, jumps and loops which are handy for various things. I don't have any programming background at all, but by using Paste History and looking at a couple of PB's scripts, I've been able to do quite a lot (*e.g.* automatically running the same learning procedure multiple times, measuring a learner's progress over time, making constraints decay or drift up over time, agent-based learning).