From Performance Errors to Optimal Competence
Learnability of OT and HG with Simulated Annealing

TAMÁS BIRÓ

University of Amsterdam, Netherlands
t.s.biro@uva.nl

COMPETENCE as linguistic optimization

Generative linguistics as an optimization problem: how to map underlying form \( U \) onto surface form \( SF(U) \)?

\[ SF(U) = \text{arg max}_w \; H(w) \; \text{s.t.} \; U \]

Target function ("Harmony") \( H(w) \) derived from elementary functions \( C_i(w) \) ("constraints" – a monomer):

1. Hard constraints: \( H(w) = C_i(w) \; \& \; C_j(w) \; \ldots \; \& \; C_k(w) \) \( \rightarrow \) \( P \& P \)
2. Weighted sum: \( H(w) = g_1 \cdot C_1(w) + g_2 \cdot C_2(w) \ldots = \ldots = g_l \cdot C_l(w) \) \( \rightarrow \) \( H \)
3. Exponential weights: \( H(w) = -C_i(w) \cdot q_i - C_j(w) \cdot q_j - \ldots - C_k(w) \cdot q_k \) \( \rightarrow \) HG

4. OT tableau row: \( H(w) = C_i(w) \cdot C_j(w) \ldots C_l(w) \) \( \rightarrow \) OT

Rank \( r_i \) for each constraint \( C_i \). We focus on OT and HG.

Grammar = set of rules. Net constraints by rank:

- OT hierarchy: \( C_i \geq C_j \) iff \( r_i > r_j \)
- HG weights: \( g_i \) = "winner" produced loser. Two approaches (–)

Can you look into the brain of the teacher?

Online learning from teacher's performance.

Any need to learn until convergence on the grammar? Until the learner finds the teacher's set of ranks?

Graded online LEARNING needs production (that is, performance)

Repeated error-driven updates of the constant ranks \( r_i \) until convergence:

- Initially: Fix random target grammar, fi.x underlying form, initial random grammar for learner
- Error-driven: "winner" produced by target grammar vs. "loser" produced by learner’s current grammar.
- Update rule: update the rank \( r_i \) of every constraint \( C_i \), depending on whether \( C_i \) prefers the winner or the loser. Two approaches: (1) \( r_i \rightarrow r_i + 1 \), while ranks \( r_i \) are already random between \( i \) and \( N \). (2) Example: \( r_i \rightarrow r_i + 1 \) if winner-prefering, decrease rank by \( r_i \) if loser-prefering constraint.

- Magnússon (2009): increase rank of all winnner-prefering constraints by \( \epsilon \), decrease rank of highest ranked loser-prefering constraint by \( \epsilon \), where \( \epsilon \) is the number of winner-prefering constraints.


e = \text{number of winning constraints} \times \text{a step by which temperature decreases} \times \text{the current temperature}.

String Grammar ("toy grammar") imitating typical OT phonology:

- Constraints: \( G = \{ [i, j, l, p = 0, 1, \ldots] \} \).
- Neighborhood structure on this candidate set: \( x_y : \) differences in one unit, \( x_{y+1} = x \).

Basic steps: change exactly one character \( \Delta \) (and \( P \) or., \( Q \) similarly). Each neighbor with equal probability.

Example: neighbors of \( 0123 \) are exactly: \( 1123, 3123, 0023, 0223, 0113, 0133, 0122 \) and \( 0120 \).

- Constraints (for all \( i, j, l, p = 0, 1, \ldots \)):
  - No \( j \) (number of character is a string): \( \forall (w) = \sum_{r=0}^{\infty} \frac{w_r}{10^{j-r}} \), \( n = w_k \).
  - No discrimination \( \text{Rank}(w) = \sum_{r=0}^{\infty} \frac{w_r}{10^{r-p}} \).
  - Size discrimination \( \text{Rank}(w) = \sum_{r=0}^{\infty} r \cdot \frac{w_r}{10^{r-p}} \).
  - Size discrimination \( \text{Rank}(w) = \sum_{r=0}^{\infty} r^2 \cdot \frac{w_r}{10^{r-p}} \).
  - Factoring \( \text{Rank}(w) = \sum_{r=1}^{\infty} w_{r-1} \cdot \frac{w_r}{10^{r-p}} \).

Distance of performances: \( JSD \).

Performance on learning: \( \Delta \) is Jensen-Shannon divergence

- Convergence criteria: \( JSD \) between sample produced by target grammar and sample produced by learner’s current grammar \( C \) (average \( JSD \) of two samples produced by target grammar) (Sample size = 100).

\[ \text{JSD} = \frac{1}{2} \left( D(P || Q) + D(Q || P) \right) \]

where \( D(P || Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)} \) (relative entropy, Kullback-Leibler divergence), and \( M(x) = P(x) \cdot Q(x) \)

- Symmetric: \( JSD(P || Q) = JSD(Q || P) \).
- Finite and non-negative: \( 0 \leq JSD(P || Q) \leq 1 \).

- \( JSD(P || Q) = 0 \) if and only if \( P(x) = Q(x) \).
- \( JSD(P || Q) = 1 \) if and only if \( P(x) \neq Q(x) \).

All same language: \( JSD(P || Q) = 0 \). Not a single overlap: \( JSD(P || L) = 1 \).

OBSERVATIONS: very long tail (significance based on Wilcoxon rank-sum test)

- Performance errors make grammars more difficult to learn: \( \text{gramm} < 0.8 \pm 0.1 \) on.
- But reversed for HG and SA (other significant, or not significant tendency). Why?
- Magnússon’s update rule (M) quicker than Boersma’s (B) (extremely significant). Due to larger update steps?
- Grammar type (OT vs. \( q \) HG) only minor influence ("hardly any" and "small, but very significant")
- OT much easier to learn than 1.5-HG (significant difference for non-case HB) NB: also quicker to produce.

Does learning speed depend on initial grammar? Or order of learning data?

New experiment: Run two learners learning the same target of learning steps.

- With same initial grammar: strong correlation in their rate of learning steps.
- Learning data not the same: slightly decreased correlation
- With different initial grammars: correlation (almost) lost, large differences in learning time.

Converging on performance: \( \Delta \) is Jensen-Shannon divergence

Perusal of the above function \( JSD(P || Q) \) can provide insight into the probability of finding the global optima. Biró (2006) demonstrates how to apply simulated annealing in the unsupervised OT case -- in this SA/OT, irregular forms also emerge, which prevent even at slow speed.

References


Performance or production as an implementation

1. Competence: theoretical knowledge: grammatical forms (explained by) grammar
2. Mental computation in the brain: produced forms implementation of grammar
A self-evident, and yet too often ignored fact about (child) language acquisition is that the learner acquiring her linguistic competence is exposed to the teacher’s linguistic performance—hence, also to performance errors, fast speech forms, or other variations. The performance pattern, which may be more complex than “simple random noise”, could in theory render the learning problem extremely difficult, but a clever learning algorithm could also make use of the errors, thereby enriching the allegedly poor stimulus.

The computational approach employed in this paper has a threefold structure. Linguistic competence (both of the teacher, and of the learner) is modelled either by standard Optimality Theory (Prince and Smolensky 1993), or by a symbolic Harmonic Grammar with exponential weights (as discussed, for instance, in Biró 2009a). Performance patterns are produced either by always taking the most harmonic form, or by symbolic simulated annealing (Bíró 2006), an algorithm introducing performance errors as a function of the “speech rate”. Finally, online learning employs either Paul Boersma’s update rule (Boersma 1997), or Giorgio Magri’s (2009) one.

The grammar (“phenomenon”) studied is the abstract string grammar proposed by Biró (2007), arguably mimicking a simple but typical phonological grammar. As several constraint rankings or weight families correspond to the same language, the learner is not expected to converge to the teacher’s competence (grammar), but to his performance (distribution of forms). In particular, the learner’s distance from the teacher is measured by the Jensen-Shannon divergence between a sample of the teacher’s performance pattern and a sample of the learner’s performance pattern. The learner is said to have learnt the target language if this distance is smaller than the divergence of two random samples of the same size produced by the teacher.

The table on the poster summarizes the results of an initial experiment. Magri’s approach is significantly faster than Boersma’s. If performance errors are present, then learning OT is faster than learning HG. Yet, we do not want to draw far-reaching conclusions from this toy grammar. We focus more on methodological issues of this novel approach, such as the “stability” of the learning process, and its dependence on the initial conditions and the order of the learning data.

Work implemented on:

Tools for Optimality Theory
http://www.birot.hu/OTKit/
A free, Java based user interface and a Java library for teaching or linguists wishing to implement their grammars.