STOCHASTIC, REWARD-BASED LEARNING OF HIDDEN STRUCTURE IN PHONOLOGY

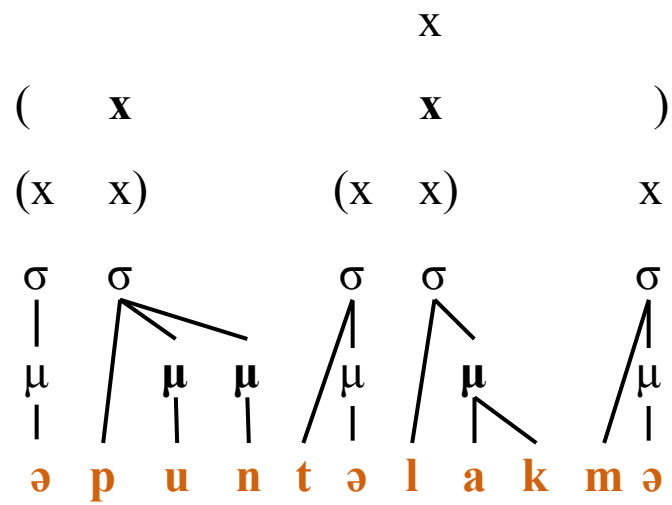
Gaja Jarosz
Yale University

OCP 2014
Leiden, Netherlands

Hidden Structure in Phonology

/ i p u m t u l a k m a /

→ or ↝



- There are several kinds of hidden structure in phonology
  - Structural ambiguity: syllables, moras, feet, autosegments, word boundaries
  - Underlying representations
  - Derivations
- Outstanding Learnability Problem. Solutions needed:
  - More realistic assumptions about what learners have access to
  - Modeling of how learning of hidden structure interacts with grammar learning

Learning Strategies for OT

- Extensive existing work on Error-Driven Learning (EDL) approaches
  - Categorical OT
    - Error-Driven Constraint Demotion & extensions (Tesar 1995; Tesar 1997; Tesar & Smolensky 1998; Merchant 2008; Akers 2012; Magri 2012; Biro 2012…)
  - Stochastic OT
    - Gradual Learning Algorithm & extensions (Boersma 1997; Boersma & Hayes 2001; Boersma 2003, Apoussidou 2007; Magri 2012; Jarosz 2012;…)
  - Harmonic Grammar and Maximum Entropy Grammars too
    - Stochastic Gradient Ascent or HG-GLA (Jaeger 2007; Boersma & Pater, to appear; Jarosz 2012; Magri 2012
  - All online constraint-based models work this way
- Production - driven
  - When there's an error, update grammar to make observed output more harmonic/likely than error
  - Learning only occurs when there's an error

Challenge: Ambiguity

- Phonological learning is full of ambiguity
  - Constraints:
    - *[tgip] Agree? SSP? *Complex? *Coda?...
  - Structural Ambiguity
    - *[(ˈaptɛ)ka] [(apˈtɛ)ka]? [(ˈaptɛ)ka]? [ap(ˈtɛka)]?...
  - Interdependent Components:
    - *[nug] Lexicon: /nuk/? Grammar: *Voice?
- The EDL has to blame something. But what?
- EDCD, GLA performance guarantees only in idealized setting without hidden structure (Tesar 1995, Magri 2012)
- Much work on enabling EDLs to deal with hidden structure
  - (Tesar & Smolensky 1998; Boersma 2003; Apoussidou 2007; Merchant 2008; Boersma & Pater to appear; Akers 2012; Jarosz 2013; Biro 2012…)

Reward Based Learning

- An under-explored alternative: Reward-Based Learning (RBL)
  - Limited existing work (Jarosz 2006a, 2006b, 2009, 2010, 2011)
  - No online learners
- Comprehension-driven
  - Analyze output and reward hidden variables that favor the output
    - Rewarding entails probability. Reward is meaningless without a notion of preference.
  - Learning occurs in response to all outputs
- Advantages
  - Well-understood machine learning algorithms work this way (EM, Gibbs, …)
  - Provable convergence with hidden structure
  - Provides a general approach to ambiguity (no special mechanisms required)
  - Ability to handle noise, variation comes for free
- This talk: application of this approach to OT

Expectation Maximization Inspiration

- Expectation Maximization (Dempster et al 1977)
  - General-purpose machine learning algorithm for hidden structure
  - Provably convergent (with hidden structure)
  - General approach to ambiguity and hidden structure
- This talk: stochastic approximations of EM adapted to OT
  - Take EM's basic insight about how to deal with hidden structure, but
  - Avoid explicit calculation of likelihood: use sampling instead
  - Introduce new grammatical representation that makes this possible
  - Introduce both batch and online learning algorithms
- Outline
  - RBL strategy, novel representation, novel algorithms
  - Simulations: Metrical Structure & Underlying Representations

Expectation Maximization Inspiration

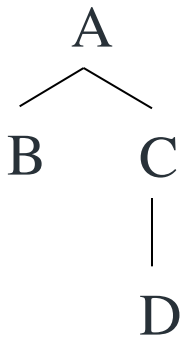
- EM for OT, intuitively (see also Jarosz 2006 et seq):
  - Iterate:
    - E-step: Select an output and analyze it
      - What relative rankings does this output favor?
    - M-step: Update the grammar
      - Reward those relative rankings
- Tiny Baby Example
  - Two constraints: IDENT[VOICE], *VOICE
  - Two words:
    - /pa/ → [pa] : IDENT ≫ *VOICE or *VOICE ≫ IDENT
    - /ba/ → [ba] : IDENT ≫ *VOICE
  - Over time learner settles on IDENT ≫ *VOICE

How to represent the grammar?

- Of course, there are more than two constraints!
- To make EM feasible, we need to represent an OT grammar in terms of independent parameters:
  - There are n! possible rankings – considering all of them separately is not computationally feasible
  - Stochastic OT won't work because parameters aren't independent and M-step is difficult
- We need to be able to
  - E-step: Analyze
    - Estimate preferred value for each parameter
  - M-step: Reward
    - Independently update parameters

Stochastic Partial Orders

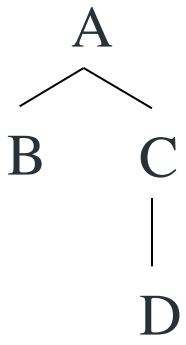
- A new grammatical representation makes analyze/reward possible
- Immediate Advantages
  - Independent updates of parameters are possible
  - Reward step is trivial
- Stochastic Partial Orders are an extension of Partial Orders
  - ABCD
  - ACBD
  - ACDB



	A	B	C	D
A		1	1	1
B	0		~	~
C	0	~		1
D	0	~	0	

Stochastic Partial Orders

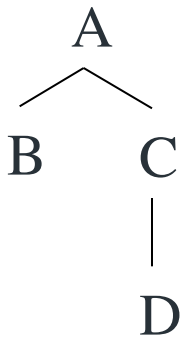
- Stochastic Partial Orders permit any value between 0 and 1
- Like POs, SPOs describe permissible rankings
  - ABCD
  - ACBD
  - ACDB
- SPOs also describe preferences about relative rankings:
  - Pr(ABCD) = .5
  - Pr(ACBD) = .25
  - Pr(ACDB) = .25



	A	B	C	D
A		1	1	1
B	0		1/2	3/4
C	0	1/2		1
D	0	1/4	0	

Sampling from SPOs

- To exploit these probabilities, we need a way to sample
- To sample a total ranking given a SPO:
  - Iterate until all mutual rankings are set:
    - Randomly select an unset mutual ranking:
      - Set it according to its pairwise ranking probability
      - Set any mutual rankings implied by transitivity
- Example:
  - Choose B vs D
    - 3/4 B » D
      - 1/2: C » B
      - 1/2: B » C
    - 1/4 D » B
      - ⇒ C » B



	A	B	C	D
A		1	1	1
B	0		1/2	3/4
C	0	1/2		1
D	0	1/4	0	

SPO EM Skeleton

- Iterate
  - E-step: analyze the data
    - For each pair of constraints (A, B) determine relative preference for A ≫ B vs. B ≫ A
  - M-step: update the grammar
    - For each pair of constraints (A, B) set the grammar parameters to the preference above

The M-step (Grammar Update)

- The principle advantage of SPOs is that the grammar update step is trivial
- Suppose the E-step revealed the data had these preferences:
  - A≫B, A≫C, B≫C: 100%
  - A≫D: 75%
  - B≫D: 50%
  - C≫D: 25%
- Finding the SPO that captures this is trivial:
  - For each pair of constraints A, B:
  - Enter the probabilities into a matrix!

	A	B	C	D
A		1	1	3/4
B	0		1	1/2
C	0	0		1/4
D	1/4	1/2	3/4	

The E-step (Analysis)

- How do we determine e.g. A≫B is 75%?
- Formally, the data's preference for A≫B is the conditional probability of A≫B, given the overt datum *d* and the current grammar *G*
- $\Pr(A >> B \mid d, G)$
- This is what needs to be estimated to do RBL

Estimating Pairwise Ranking Likelihood

- The proposed algorithms estimate this by exploiting Bayes' Law:
  - $$\Pr(A >> B \mid d, G) = \frac{\Pr(d \mid A >> B, G) \cdot \Pr(A >> B \mid G)}{\Pr(d \mid G)}$$
- The terms on the right:
  - $\Pr(A >> B \mid G)$
    - This is the grammar parameter! Simple look-up!
  - $\Pr(d \mid A >> B, G)$
    - Novel constrained sampling approach
  - $\Pr(d \mid G)$
    - Combination of the above
    - $$\Pr(d \mid G) = \Pr(d \mid A >> B, G) \cdot \Pr(A >> B \mid G) + \Pr(d \mid B >> A, G) \cdot \Pr(B >> A \mid G)$$

Estimating Pairwise Ranking Likelihood

- $\Pr(d \mid \mathrm{A} >> \mathrm{B}, \mathrm{G})$
  - How likely is *d* if A >> B but otherwise grammar is left as is?
- Novel constrained sampling approach
- For each pair of constraints:
  - Set A >> B and sample outputs for *d*
  - Proportion of matches is the estimate!
    - Structure is irrelevant – matches look only at overt form *d*

SPOEM: batch

- E-Step
  - Initialize |CON| x |CON| matrix M to 0s
  - For each data point *d*:
    - For each pair of constraints A and B:
      - Temporarily set A >> B
        - Sample *n* total rankings to estimate Pr(*d* | A>>B)
      - Temporarily set B >> A
        - Sample *n* total rankings to estimate Pr(*d* | B>>A)
    - Use Bayes' Law to calculate preferences and add them to M
- M-Step
  - Set $\Pr_{G_{new}}(A >> B) = \frac{M(A,B)}{M(A,B)+M(B,A)}$

SPOEM: online

- SPOEM Batch introduces the essential RBL ingredients
  - SPO representation makes grammar update trivial
  - SPO representation makes constrained sampling possible
  - SPOEM deals implicitly and generally with hidden structure
- Batch processing is not ideal for modeling acquisition
  - Children learn gradually as data comes in
  - Learning data is a random sample from the language
- Minor modification yields SPOEM Online…

SPOEM: online

- E-Step
  - Initialize |CON| x |CON| matrix M to 0s
  - Sample a data point *d*:
    - For each pair of constraints A and B:
      - Temporarily set A >> B
        - Sample *n* total rankings to estimate Pr(*d* | A>>B)
      - Temporarily set B >> A
        - Sample *n* total rankings to estimate Pr(*d* | B>>A)
    - Set M(A,B) = estimated Pr(A>>B| *d*, *G*)
- M-Step
  - Move $\Pr_{G_{new}}(A >> B)$ closer to $\frac{M(A,B)}{M(A,B)+M(B,A)}$

SPOEM: online (simplified)

- Iterate
  - Sample a data point
    - For each pair of constraints
      - Determine which pairwise ranking is favored by the output
  - Update grammar slightly toward these preferences

Simulations: Structural Ambiguity

- Tesar and Smolensky (2000) metrical structure system:
  - 12 metrical structure constraints
  - 124 languages
  - Structural ambiguity:
    - Overt data is the stress pattern [L1-L-L2-H2-L]
    - Learner must discover foot structure and ranking
- Enables direct comparison to existing models
- Previous results with EDLs plus Parsing Mechanisms
  - 45-60%: RIP/EDCD (Tesar & Smolensky 2000, Boersma & Pater to appear)
  - 58-59%: RIP/GLA (Boersma & Pater to appear, Jarosz 2013)
  - 84%: RRIP/GLA (Jarosz 2013)
  - **94%: EIP/GLA** (Jarosz 2013)

Simulations: Set Up

- All results are averaged over 10 runs for each language
- Varying sample size from 10 to 100
- Initial grammar for all simulations was set to 0.5 for all parameters
  - All constraints ranked equally
- Maximum iterations
  - SPOEM batch: 1000
  - SPOEM online: 10000
- Accuracy - Percent of languages learned

Simulations: Results

	10	25	50	100
SPOEM batch	95.54 (0.56)	95.65 (0.56)	95.73 (1.19)	95.65 (0.94)
SPOEM online	95.32 (1.19)	95.08 (1.11)	95.32 (0.99)	95.65 (0.87)

- Results consistently: 95.08% - 95.73%
- All results are comparable (n.s.)
  - Online performs as well as batch (with very limited resources)
  - Performance stable even with very small sampling size
- Significantly better than existing models
  - Substantially higher than RIP/EDCD, RIP/GLA, RRIP/GLA
  - Significantly better than EIP/OT-GLA ($p < 0.05$; Welch two-sample t-test)

General Approach to Ambiguity

- These results show SPOEM is a viable alternative to existing EDLs when it comes to structural ambiguity
  - Success rates are higher than EDLs
  - Learning takes fewer iterations (not shown)
- SPOEM fully exploits its probabilistic grammar:
  - Each data point's grammatical preferences can be estimated
  - SPO representation records these preferences concisely
  - Processing of each data point influences learning
- And, this approach is fully general…

Extending SPOEM to URs

- Use same approach to learn Grammars & Lexicons!
- Example: lexical/grammatical stress & length
  - ˈpaː-ki pi-ˈkuː pe-ˈkiː piː-ˈta…
  - Task:
    - Grammar: Stress aligned to left? Right? Weight sensitive?
    - Lexicon: is [paː] underlying long? Or predictably long? Stressed?
- SPOEM + UR
  - Grammar learned just as before
  - Lexicon involves binary features: ±LONG, ±STRESS
  - Set features to + and – one-by-one, and sample as before!

Simulations: UR Learning

- Test Set: Tesar's (2006) PAKA language:
  - 6 Constraints (MAINLEFT, MAINRIGHT, *V:, WTS, ID-LENGTH, ID-STRESS)
  - 2 UR features (±LONG, ±STRESS)
  - 24 languages in the system
  - Learner's task
    - Learn ranking and UR feature settings
- Results
  - 100% accuracy
  - SPOEM+UR learns all 24 languages on all runs
  - Combination of ranking and lexicon produces all forms correctly in each language

Summary

- Novel SPO representation makes RBL for OT possible
- SPOEM succeeds on multiple types of hidden structure
  - Best performance on T&S (2000) stress system
  - Learns both URs and Grammars for all of Tesar's (2000) PAKA languages
- RBLs provide a general approach to hidden structure
  - No specific mechanisms to deal with structure needed
  - Here: OT, but approach easily extendible (e.g. P&P)
- This approach combines
  - Principles of statistical inference from machine learning
  - Phonological Theory
  - Gradual, online learning that makes acquisition modeling possible

Thank You

Select References

- Apoussidou, Diana. 2007. The learnability of metrical phonology. PhD Dissertation, University of Amsterdam.
- Boersma, Paul. 2003. Review of Tesar & Smolensky (2000): Learnability in Optimality Theory. *Phonology* 20(3).. 436-446.
- Boersma, Paul and Pater, Joe. to appear. Convergence Properties of a Gradual Learning Algorithm for Harmonic Grammar. In John McCarthy and Joe Pater (editor.), *Harmonic Grammar and Harmonic Serialism*. London: Equinox Press.
- Jarosz, Gaja. 2006. Rich Lexicons and Restrictive Grammars - Maximum Likelihood Learning in Optimality Theory. PhD Dissertation, the Johns Hopkins University, Baltimore, MD.
- Jarosz, Gaja. 2013. Learning with Hidden Structure in Optimality Theory and Harmonic Grammar: Beyond Robust Interpretive Parsing. *Phonology* 30(1).. Cambridge University Press. 27-71.
- Merchant, Nazarré. 2008. Discovering Underlying Forms: Contrast Pairs and Ranking. PhD Dissertation, Rutgers University, New Brunswick, NJ.
- Tesar, Bruce and Smolensky, Paul. 1998. Learnability in Optimality Theory. *Linguistic Inquiry* 29 (2).. 229-268.

Further Directions

- SPO & SPOEM
  - SPOs connection to Partially Order Grammars (Anttila 2002) and free variation
    - No learning algorithm for POGs
  - General approach can straighforwardly extend to derivations
    - SPOEM can be applied to learning in Harmonic Serialism as-is
    - SPOEM can be applied to learning opaque derivations
- RBLs more generally
  - Given appropriate representations, RBLs are possible for other generative theories
    - for example Principles & Parameters

Why test constructed languages?

- Generality
  - Standard approach is to require success given arbitrary constraints
- Understanding
  - Systematically manipulate properties to understand the model
- Learnability
  - Learnability depends on ability to navigate an entire hypothesis space (defined by arbitrary constraints) successfully
  - Success of a model on a real language says nothing about its learnability properties
- Practicality
  - Creation of test data for a large typology is a significant endeavor
  - Requires theoretical analysis of all the data (e.g. set of constraints)
  - This is an important next step…

Extending SPOEM to URs

- E-Step
  - Initialize matrix M_1 of (grammar) expected counts
  - Initialize matrix M_2 of (lexicon) expected counts
  - For each data point d_i:
    - For each pair of constraints C_a and C_b:
      - Temporarily set $C_a >> C_b$
        - Sample *n* UR, ranking pairs to estimate $Pr(d_i \mid C_aC_b)$
      - Temporarily set $C_b >> C_a$
        - Sample *n* UR, ranking pairs to estimate $Pr(d_i \mid C_bC_a)$
      - Add expected counts to M_1
    - For each UR feature:
      - Temporarily set it to +
        - Sample *n* UR, ranking pairs to estimate $Pr(d_i \mid +)$
      - Temporarily set it to -
        - Sample *n* UR, ranking pairs to estimate $Pr(d_i \mid -)$
      - Add expected counts to M_2
- M-Step
  - Update G by normalizing M_1
  - Update L by normalizing M_2

Simulations: UR Learning - Settings

- All simulations averaged over 10 runs for each language
- Initialization:
  - Initial grammar: all parameters set to .5
  - Initial lexicon: all UR features set to .5
- Sample size:
  - $n = 50$ samples for conditional likelihood estimates
- Iterations
  - 1000
- Languages counted as learned if:
  - All data points were produced with 100% accuracy when sampled 100 times
  - Total error < 0.01

True EM for SPOs

- Of course we don't know the true rankings!
  - E-step calculates the expected rankings based on the data and the current grammar
  - Then, the M-step does what we just saw based on those
- True EM for SPOs:
  - Expectation
    - For each pair of constraints C_a, C_b:
      - Calculate $E(\#C_aC_b)$ given current grammar and the data
  - Maximization
    - For each pair of constraints C_a, C_b:
      - Set $\Pr(C_aC_b) = \dfrac{E(\#C_aC_b)}{E(\#C_aC_b) + E(\#C_bC_a)}$

Expected Number of Pairwise Rankings

- What is E(#C_aC_b)?
  - It's really *given the data* D *and current grammar* G_i
  - For a single data point d_i with frequency f_i, it's simply:
    - $E(\#C_aC_b \mid d_i, G_j) = \Pr(C_aC_b \mid d_i, G_j) \cdot f_i$

Expected Number of Pairwise Rankings

- What is E(#C_aC_b)?
  - It's really *given the data* D *and current grammar* G_i
  - For a single data point d_i with frequency f_i, it's simply:
    - $E(\#C_aC_b \mid d_i, G_j) = \Pr(C_aC_b \mid d_i, G_j) \cdot f_i$
  - For all the data, we simply sum:
    - $E(\#C_aC_b \mid D, G_j) = \sum_{d_i \in D} E(\#C_aC_b \mid d_i, G_j)$

Expected Number of Pairwise Rankings

- What is E(#C_aC_b)?
  - It's really *given the data* D *and current grammar* G_i
  - For a single data point d_i with frequency f_i, it's simply:
    - $E(\#C_aC_b \mid d_i, G_j) = \Pr(C_aC_b \mid d_i, G_j) \cdot f_i$
  - For all the data, we simply sum:
    - $E(\#C_aC_b \mid D, G_j) = \sum_{d_i \in D} E(\#C_aC_b \mid d_i, G_j)$
  - So the overall update equations for true EM are:

$$\Pr_{G_{j+1}}(C_aC_b) = \frac{E(\#C_aC_b \mid D, G_j)}{E(\#C_aC_b \mid D, G_j) + E(\#C_bC_a \mid D, G_j)}$$

$$\Pr_{G_{j+1}}(C_bC_a) = 1 - \Pr_{G_{j+1}}(C_aC_b)$$

Estimating E(#C_aC_b)

- Posterior Probability of ranking is key:
  - $E(\#C_aC_b \mid d_i, G_j) = \Pr(C_aC_b \mid d_i, G_j) \cdot f_i$
- By Bayes' Law:
  - $\Pr(C_aC_b \mid d_i, G_j) = \dfrac{\Pr(d_i \mid C_aC_b, G_j) \cdot \Pr(C_aC_b \mid G_j)}{\Pr(d_i \mid G_j)}$
- Decompose into conditional likelihood and prior

Estimating E(#C_aC_b)

- Posterior Probability of ranking is key:
  - $E(\#C_aC_b \mid d_i, G_j) = \Pr(C_aC_b \mid d_i, G_j) \cdot f_i$
- By Bayes' Law:
  - $\Pr(C_aC_b \mid d_i, G_j) = \dfrac{\Pr(d_i \mid C_aC_b, G_j)\,\Pr(C_aC_b \mid G_j)}{\Pr(d_i \mid G_j)}$
- Prior is just the SPO parameter!

Estimating E(#C_aC_b)

- Posterior Probability of ranking is key:
  - $E(\#C_aC_b \mid d_i, G_j) = \Pr(C_aC_b \mid d_i, G_j) \cdot f_i$
- By Bayes' Law:
  - $\Pr(C_aC_b \mid d_i, G_j) = \dfrac{\Pr(d_i \mid C_aC_b, G_j) \cdot \Pr(C_aC_b \mid G_j)}{\Pr(d_i \mid G_j)}$

$$= \Pr(d_i \mid C_aC_b, G_j) \cdot \Pr(C_aC_b \mid G_j) + \Pr(d_i \mid C_bC_a, G_j) \cdot \Pr(C_bC_a \mid G_j)$$

- Likelihood is just product of prior and cond. likelihood

Estimating E(#C_aC_b)

- Posterior Probability of ranking is key:
  - $E(\#C_aC_b \mid d_i, G_j) = \Pr(C_aC_b \mid d_i, G_j) \cdot f_i$
- By Bayes' Law:
  - $\Pr(C_aC_b \mid d_i, G_j) = \dfrac{\Pr(d_i \mid C_aC_b, G_j) \cdot \Pr(C_aC_b \mid G_j)}{\Pr(d_i \mid G_j)}$

$$= \Pr(d_i \mid C_aC_b, G_j) \cdot \Pr(C_aC_b \mid G_j) + \Pr(d_i \mid C_bC_a, G_j) \cdot \Pr(C_bC_a \mid G_j)$$

- Conditional likelihood can be estimated by constrained sampling:
  - Set $C_a >> C_b$ and sample outputs for d_i!
  - Proportion of matches is the estimate!

Iteratively Setting Pairwise Ranks

- This sampling is not exact
  - The sample it generates doesn't exactly match the pairwise probs
  - It's much better than multiplying the pairwise probs
  - In practice, the sampling error is very small:
    - For a sample of 1000, the average sampling error for a pairwise ranking probability is around 1%
  - And it seems to work well in simulations

SPO parameters aren't independent

- Explicitly calculating the probability of a total ranking is tricky…
- We might be tempted to multiply the pairwise ranking probs:
  - Pr(ABCD) = 1*1*1*1/2*3/4*1 = 3/8 (should be .5)
  - Pr(ACBD) = 1*1*1*1/2*1*3/4 = 3/8 (should be .25)
  - Pr(ACDB) = 1*1*1*1*1/2*1/4 = 1/8 (should be .25)
- The parameters aren't independent…
- Luckily, we don't need to explicitly calculate the probabilities
  - All we need is to be able to sample randomly from this grammar

	A	B	C	D
A		1	1	1
B	0		1/2	3/4
C	0	1/2		1
D	0	1/4	0	

Simulations: Settings

- All results are averaged over 10 runs for each language
- Initial grammar for all simulations was set to 0.50 for all parameters
  - All constraints ranked equally
- Learning Rates
  - sPOEM batch: full EM update
  - sPOEM online: 0.25 toward the full update
- Maximum iterations
  - sPOEM batch: 1000
  - sPOEM online: 10000
- Languages were counted as learned if either:
  - All data points were produced with 100% accuracy when sampled 100 times
  - Total error < 0.5 (out of 62)

Simulations - Results

	10	25	50	100
sPOEM batch	95.54 (0.56)	95.65 (0.56)	95.73 (1.19)	95.65 (0.94)
sPOEM batch 2x			**97.5 (0.71)**	**97.34 (0.77)**
sPOEM online	95.32 (1.19)	95.08 (1.11)	95.32 (0.99)	95.65 (0.87)

- All sPOEM batch significantly better than EIP/OT-GLA ($p < 0.05$; Welch two-tailed two-sample t-test)
- sPOEM online better than EIP/OT-GLA
- sPOEM batch sample size makes no difference ($p > .1$)

Simulations: UR Learning - Results

- sPOEM+UR learns all 24 languages on all runs
  - Combination of ranking and lexicon produces all forms correctly
- Learns the target URs for each morpheme
  - r1 = /pa/ r2 = /pa:/ r3 = /pá/ r4 = /pá:/ s1 = /ka/ s2 = /ká/ s3 = /ká:/
- Learns a SPO grammar with all required crucial rankings
- It favors balanced SPOs. Example:
  - Crucial rankings shown on the right
  - The algorithm learns the SPO on the left
    - Pr(Id-Stress » WTS) varies btwn .22 and .50
    - Pr(MainR » *V:) varies btwn .41 and .71