Learning Phonotactics in a Differentiable Framework of Subregular Languages

Huteng Dai[1] and Richard Futrell[2]

[1] Rutgers University, New Brunswick
[2] University of California, Irvine

Take-home messages

- A general differentiable framework for modeling and inducing phonotactics;
- We can compare and even combine different subregular classes;
- We haven't found "one true model” for phonotactics. For example, some models perform better in learning nonlocal phonotactics in Navajo and Quechua, but not in others.

Roadmap

1. Introduction
2. Framework
3. Evaluation and result

Language and Grammar

"Language": a set of strings.

For example, given an inventory {s, o, ʃ},

Language A	Language B
Legal: soʃ, sosoʃ, …	Legal: ʃoʃoʃ, ʃos, …
Illegal: *sʃo, *sosʃ …	Illegal: *soʃ, *sosoʃ, …

"Class": a set of languages.

Language and Grammar

"Grammar": a set of constraints.

Strictly Local (SL)

"No adjacent sʃ sequence"

Grammar A: *sʃ

Legal: soʃ, sosoʃ, …

Illegal: *sʃo, *sosʃ …

Strictly Piecewise (SP)

"No s followed by an ʃ"

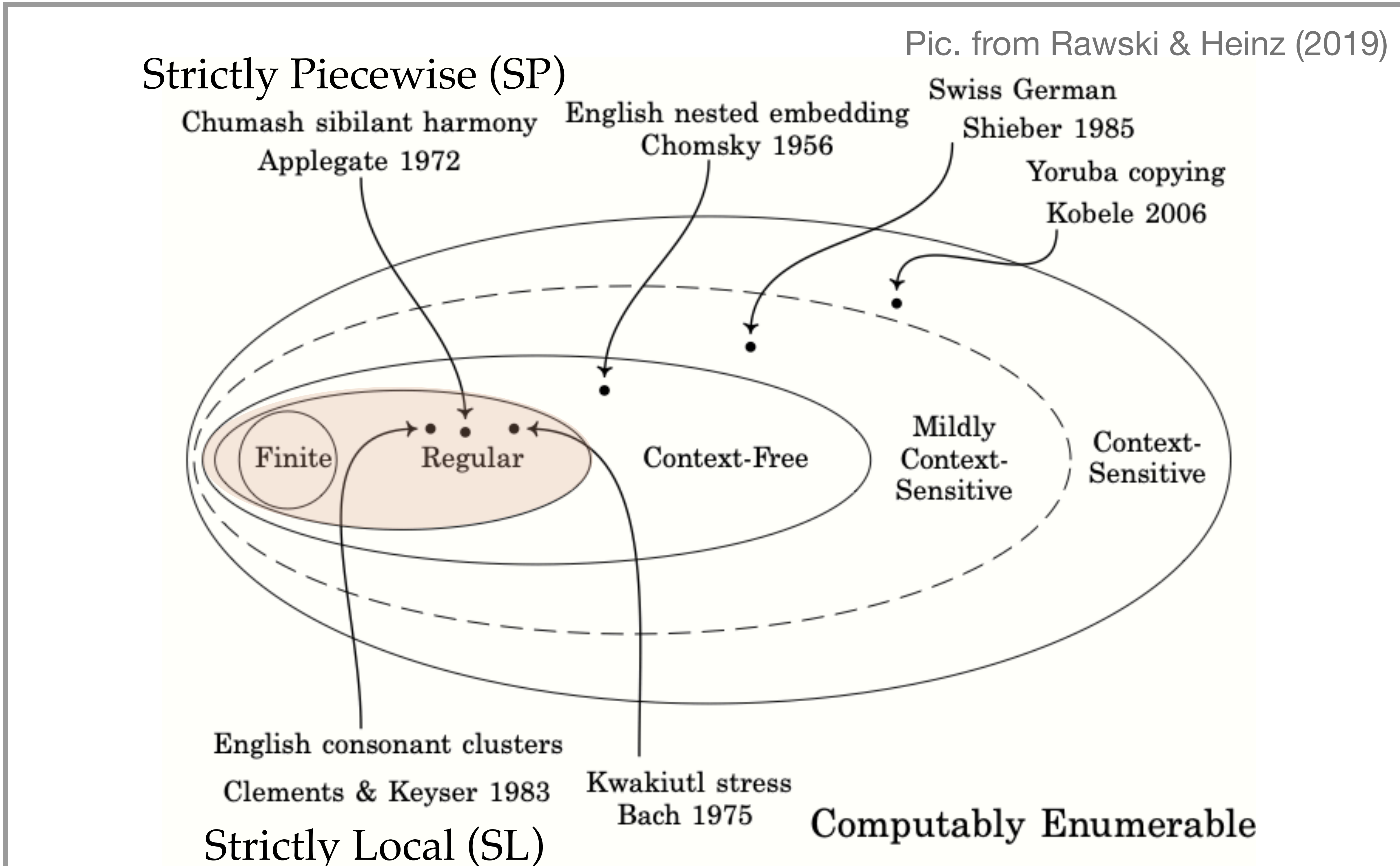
Grammar B: *s…ʃ

Legal: ʃoʃoʃ, ʃos, …

Illegal: *soʃ, *sosoʃ, …

Heinz (2010); Heinz & Rogers (2010)

Subregular hierarchy and phonological patterns



Finite-state Automata (FSAs)

Subregular grammars can all be represented as Finite-state Automata. For example,

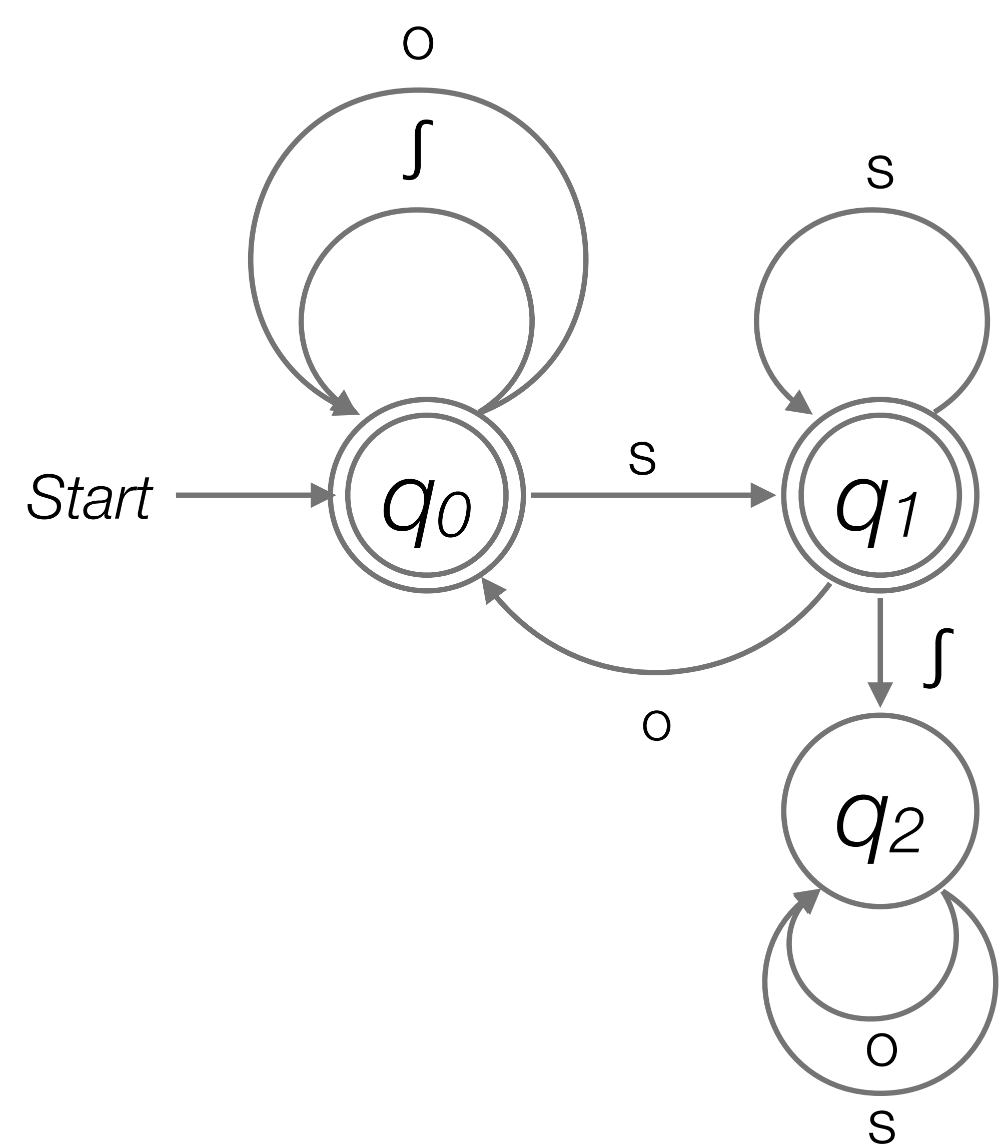
Strictly Local (SL)

"No adjacent sʃ sequence"

Grammar A: *sʃ

Legal: soʃ, sosoʃ, …

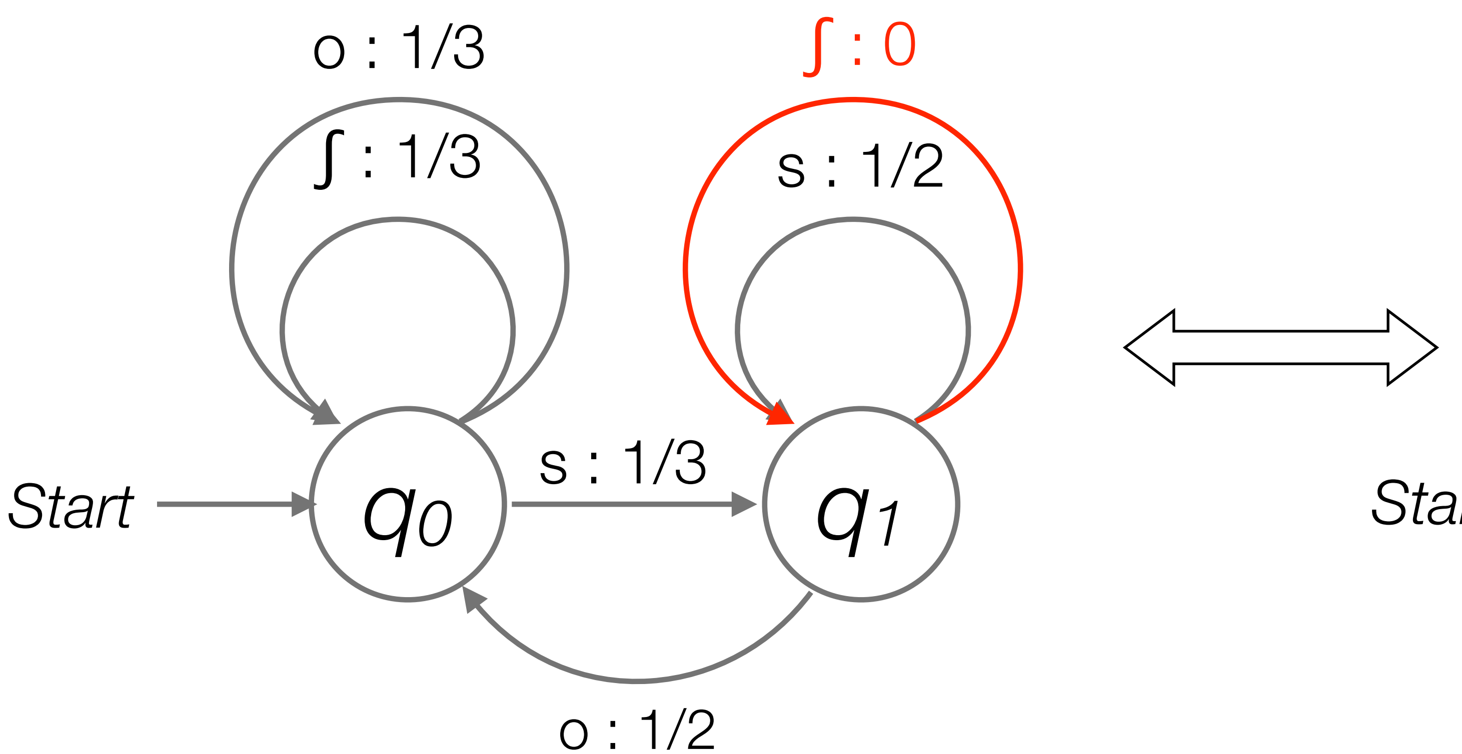
Illegal: *sʃo, *sosʃ …

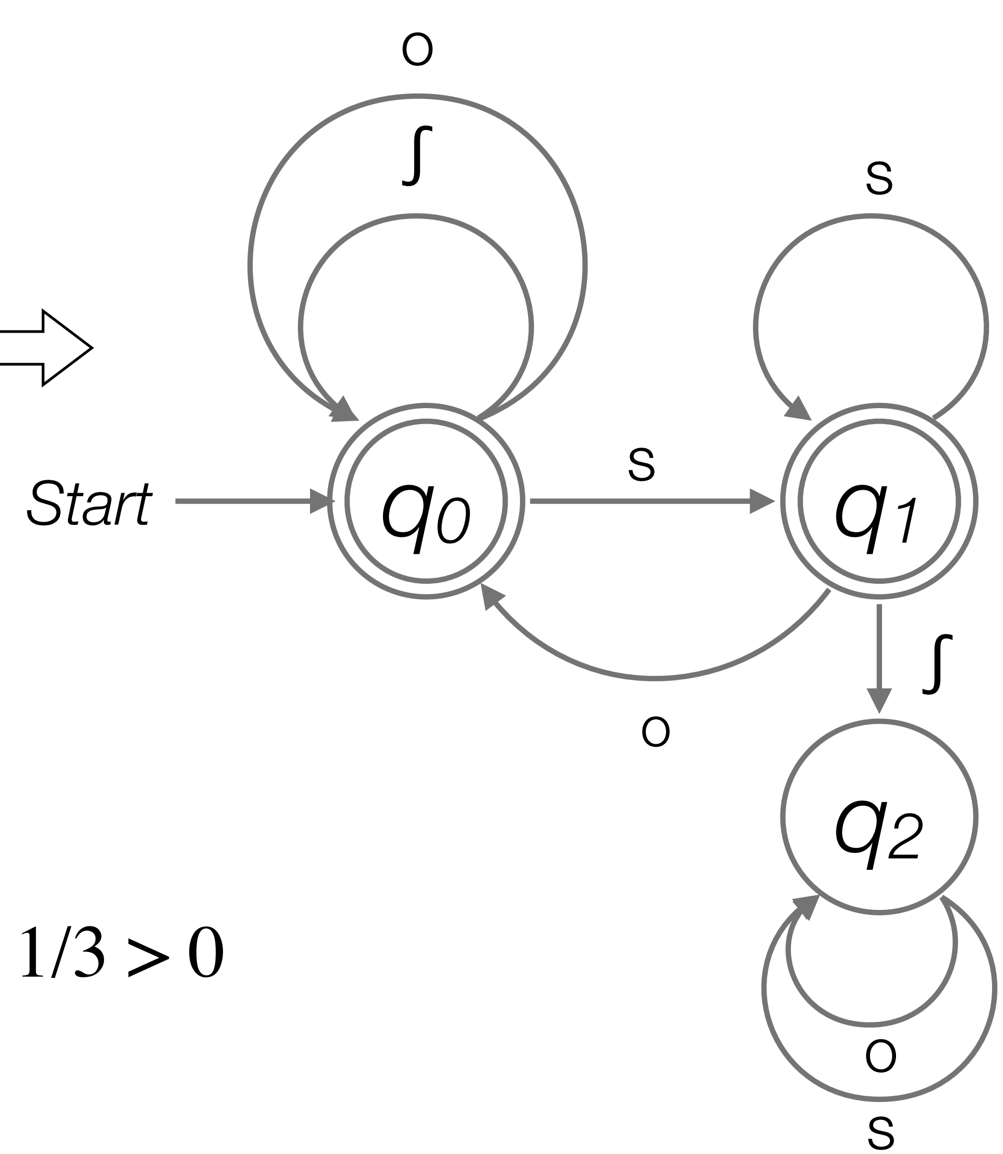


Probabilistic Finite-state Automata (PFAs)

By adding probability to each transition, we can now correlate wellformedness with word likelihood.

Vidal et al (2005)





Likelihood(*sosʃ) = 1/3 * 1/2 * 1/3 * 0 = 0

Likelihood(sosoʃ) = 1/3 * 1/2 * 1/3 * 1/2 * 1/3 > 0

→ Likelihood(sosoʃ) > Likelihood(*sosʃ)

Phonotactic learning

Goal: find the target grammar that **predicts** the *unseen* data.

- Probabilistic model: good at handling noisy corpus data;
- The subregular hierarchy provides a restricted hypothesis space.

Hypothesis space

Previous works: learning algorithms for individual subregular classes, e.g.

- (Tier-based) Strictly Local (‘local *n*-grams’)

(Hayes & Wilson 2008; Jardine & Heinz 2016 Gouskova & Gallagher 2020; Lambert 2021)

- Strictly Piecewise (‘nonlocal *n*-grams’)

(Heinz 2010; Shibata & Heinz 2019; Dai 2021)

Is it possible to compare and combine these classes?

Contributions

* We provide a unified framework for learning finite-state constraints based on PFAs.

* The hypothesis space can be constrained to any (sub)regular class.

* We compare the ability of learners in different classes to capture phonotactic patterns.

Roadmap

1. Introduction
2. Framework
3. Evaluation and result

The framework

Initializing PFA matrices

↓

Computing word likelihood

↓

Inducing the model by gradient methods

Matrices and PFAs

We can parameterize PFAs by matrices.

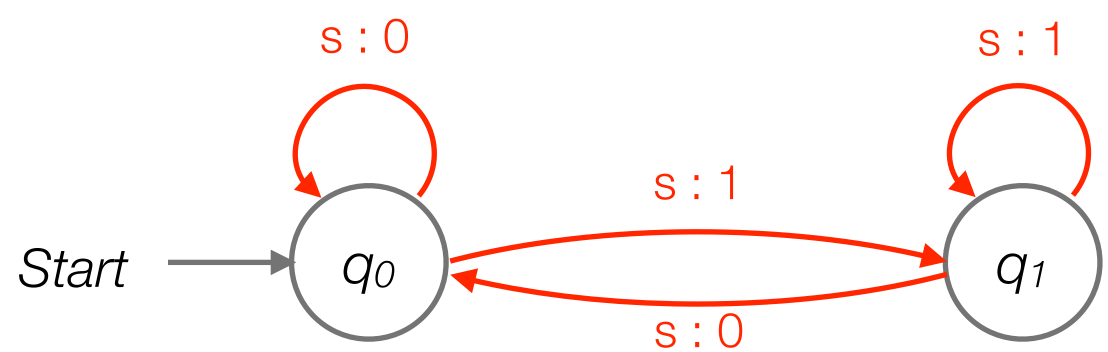
Transition matrix (T_x):

“How the states change given a symbol *x*”

Emission matrix (E):

Transition matrices in PFAs

How do we encode transitions as a matrix?



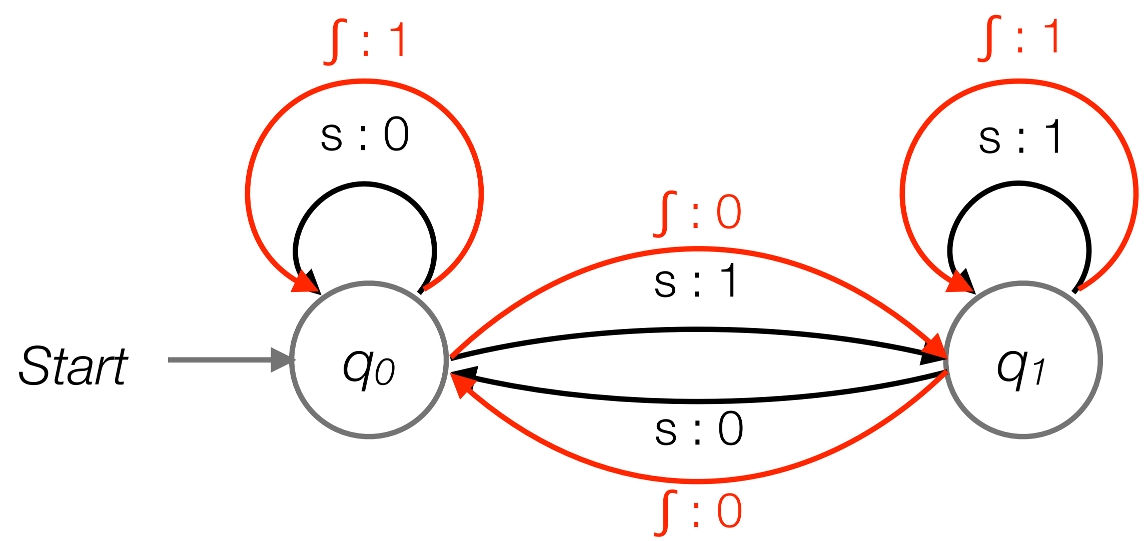
$$
T_s = \begin{array}{c} \\ q_0 \\ q_1 \end{array} \!\!\begin{array}{c} \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}
$$

On symbol s in state q_0,
transition into state q_0 is disallowed

On symbol s in state q_1,
transition into state q_1 is allowed

Transition matrices in PFAs

How do we encode transitions as a matrix?



$$T_s = \begin{array}{c c} & \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{array}{c} q_0 \\ q_1 \end{array} & \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}$$

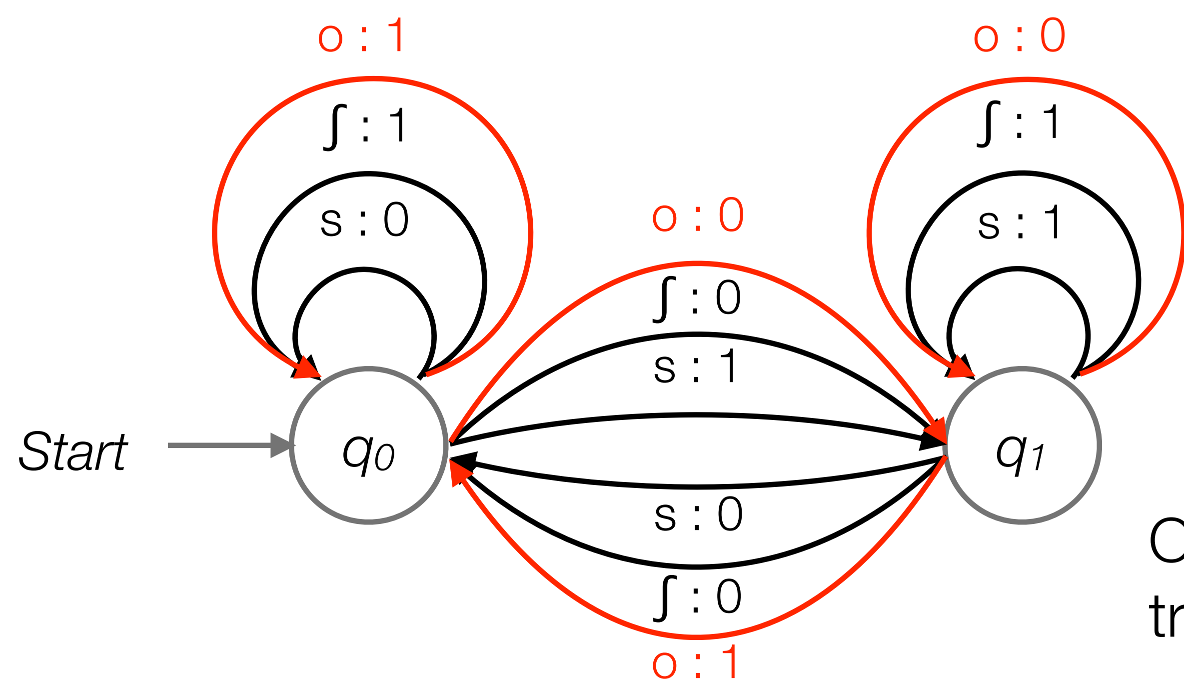
$$T_ʃ = \begin{array}{c c} & \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{array}{c} q_0 \\ q_1 \end{array} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array}$$

On symbol ʃ in state q_0,
transition into state q_0 is allowed

On symbol ʃ in state q_1,
transition into state q_0 is disallowed

Transition matrices in PFAs

How do we encode transitions as a matrix?

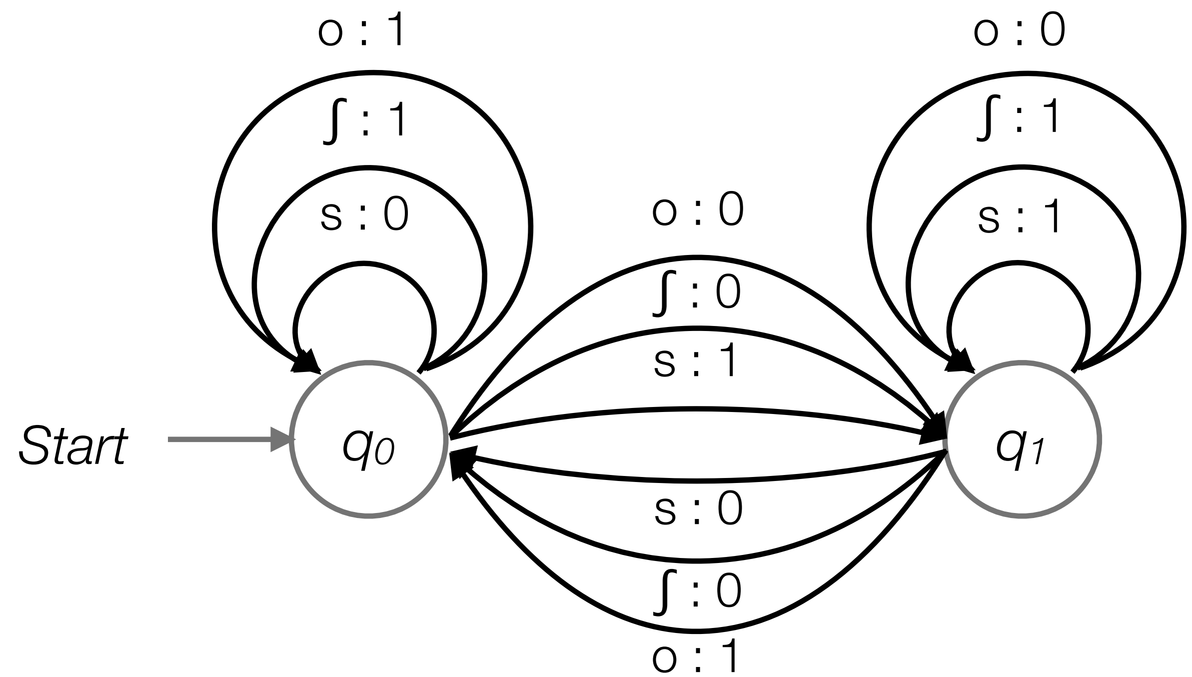


On symbol *o* in state q_0, transition into state q_0 is allowed

$$T_s = \begin{array}{c c} & \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{array}{c} q_0 \\ q_1 \end{array} & \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array} \qquad T_ʃ = \begin{array}{c c} & \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{array}{c} q_0 \\ q_1 \end{array} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \qquad T_o = \begin{array}{c c} & \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{array}{c} q_0 \\ q_1 \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \end{array}$$

On symbol *o* in state q_1, transition into state q_0 is disallowed

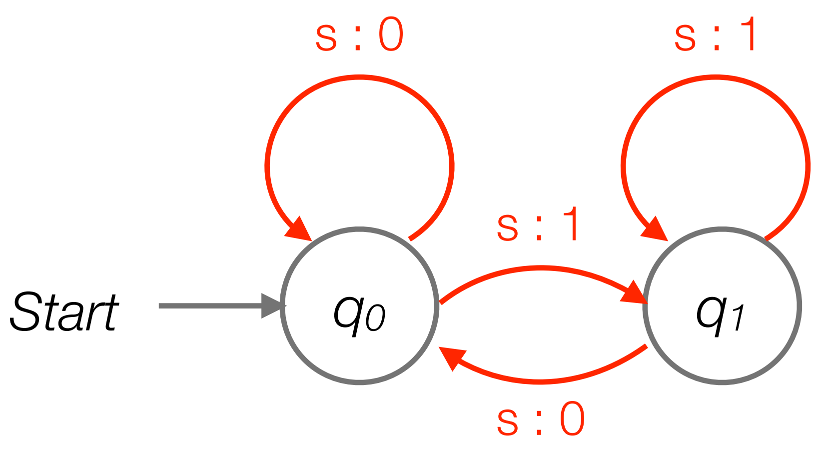
Summary: transition matrix

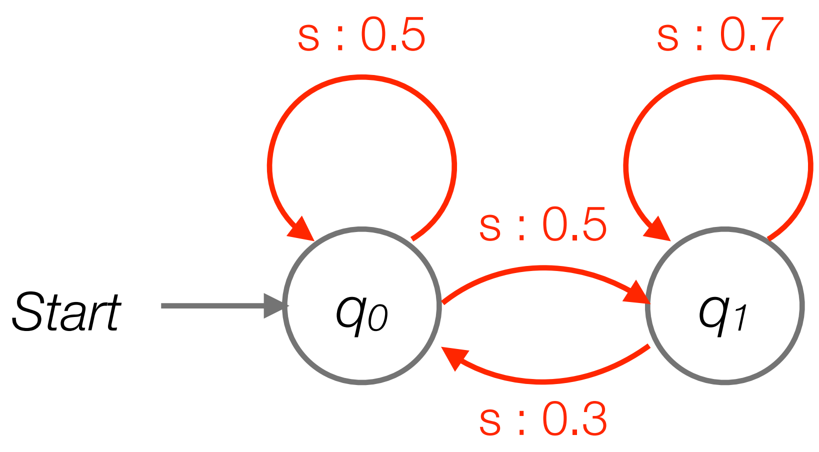


$$T_s = \begin{array}{c c} & \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{array}{c} q_0 \\ q_1 \end{array} & \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array} \qquad T_ʃ = \begin{array}{c c} & \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{array}{c} q_0 \\ q_1 \end{array} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \qquad T_o = \begin{array}{c c} & \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{array}{c} q_0 \\ q_1 \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \end{array}$$

"Always move into q_1" "Stay in the same state" "Always move into q_0"

Deterministic vs nondeterministic PFAs





$$T_s = \begin{array}{c} \\ q_0 \\ q_1 \end{array} \begin{array}{c} \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}$$

Deterministic: binary;
one symbol only enters one state

$$T_s = \begin{array}{c} \\ q_0 \\ q_1 \end{array} \begin{array}{c} \begin{array}{cc} q_0 & q_1 \end{array} \\ \begin{bmatrix} 0.5 & 0.5 \\ 0.3 & 0.7 \end{bmatrix} \end{array}$$

Nondeterministic: numeric;
Each symbol can enter multiple states

Encoding subregular constraints in transition matrix

Subregular constraints corresponds to **Deterministic** PFAs which have a highly restrictive and consistent shape.

These constraints are translated into linear-algebraic constraints on transition matrices.

We can hard-code their **transition matrices** T and learn only their emission matrix E.

Matrices and PFAs

We can parameterize PFAs by matrices.

Transition matrix (T_x):

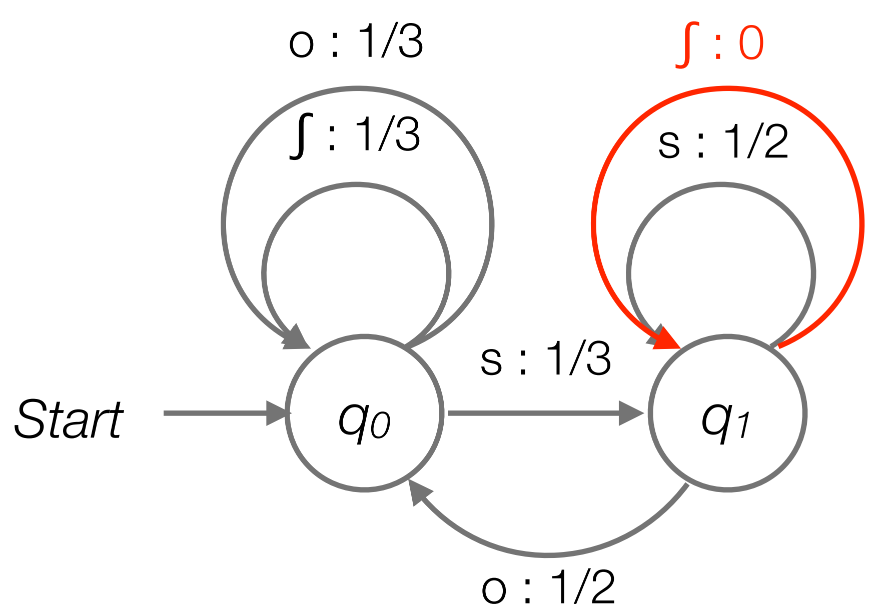
"How the states change given a symbol *x*"

Emission matrix (E):

"How likely the automaton observes certain symbol from each state"

Emission matrices in PFAs

Emission matrices represent emission probabilities:



$$E = \begin{array}{c} \\ q_0 \\ q_1 \end{array} \begin{array}{c} \begin{array}{ccc} \text{S} & \text{O} & \text{ʃ} \end{array} \\ \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \end{bmatrix} \end{array}$$

ʃ can be emitted from q_0 with 1/3 probability

ʃ cannot be emitted from q_0

Computing word likelihood

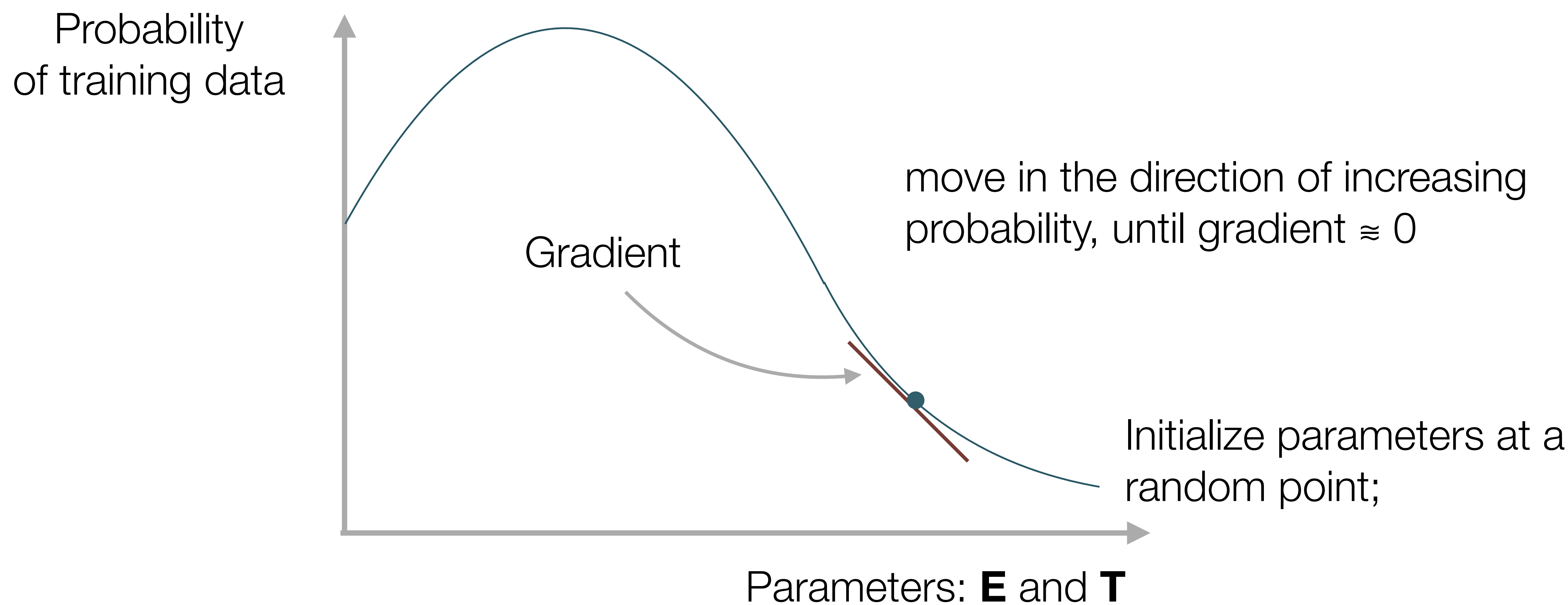
- We can obtain the probability of a word by multiplying matrices **E** and **T** in a certain order.

See details in Dai & Futrell (2021) SIGMORPHON 2021 Paper

- Differentiable: we can learn PFAs from a set of training sequences by gradient methods.

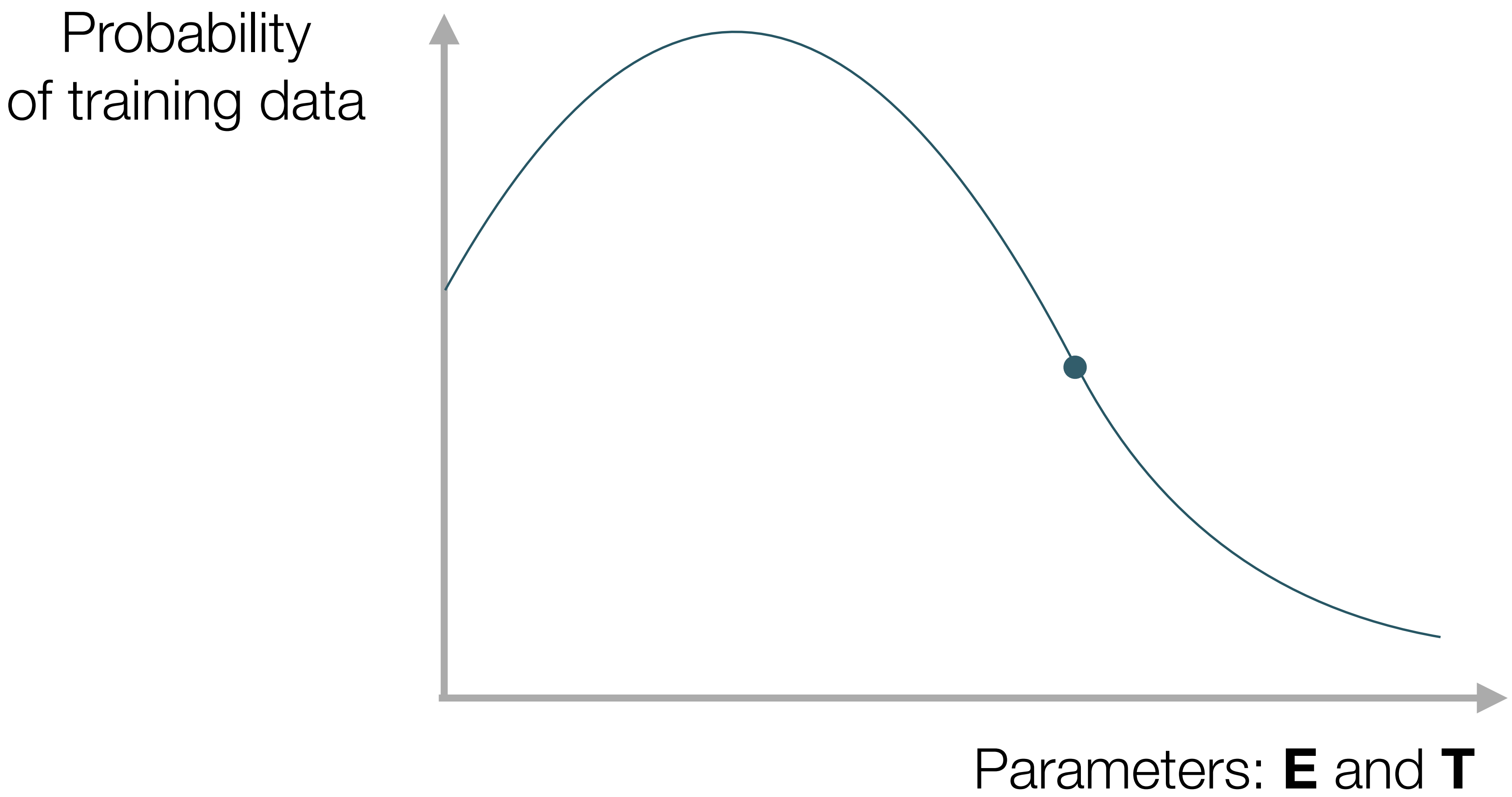
Learning by gradient methods

Find matrices **E** and **T** to maximize the probability of training data:



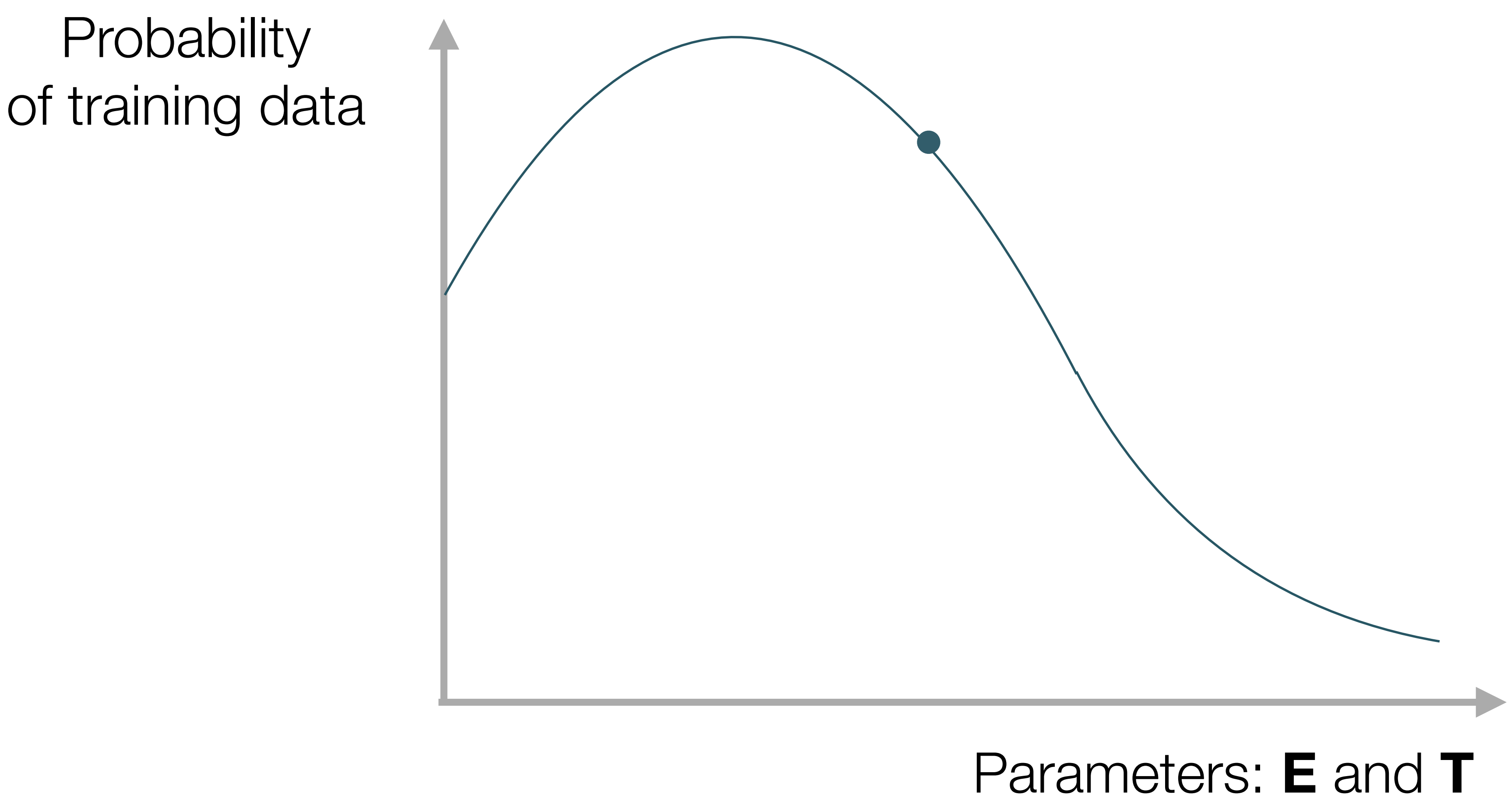
Learning by gradient methods

Every update is called a “training step”.



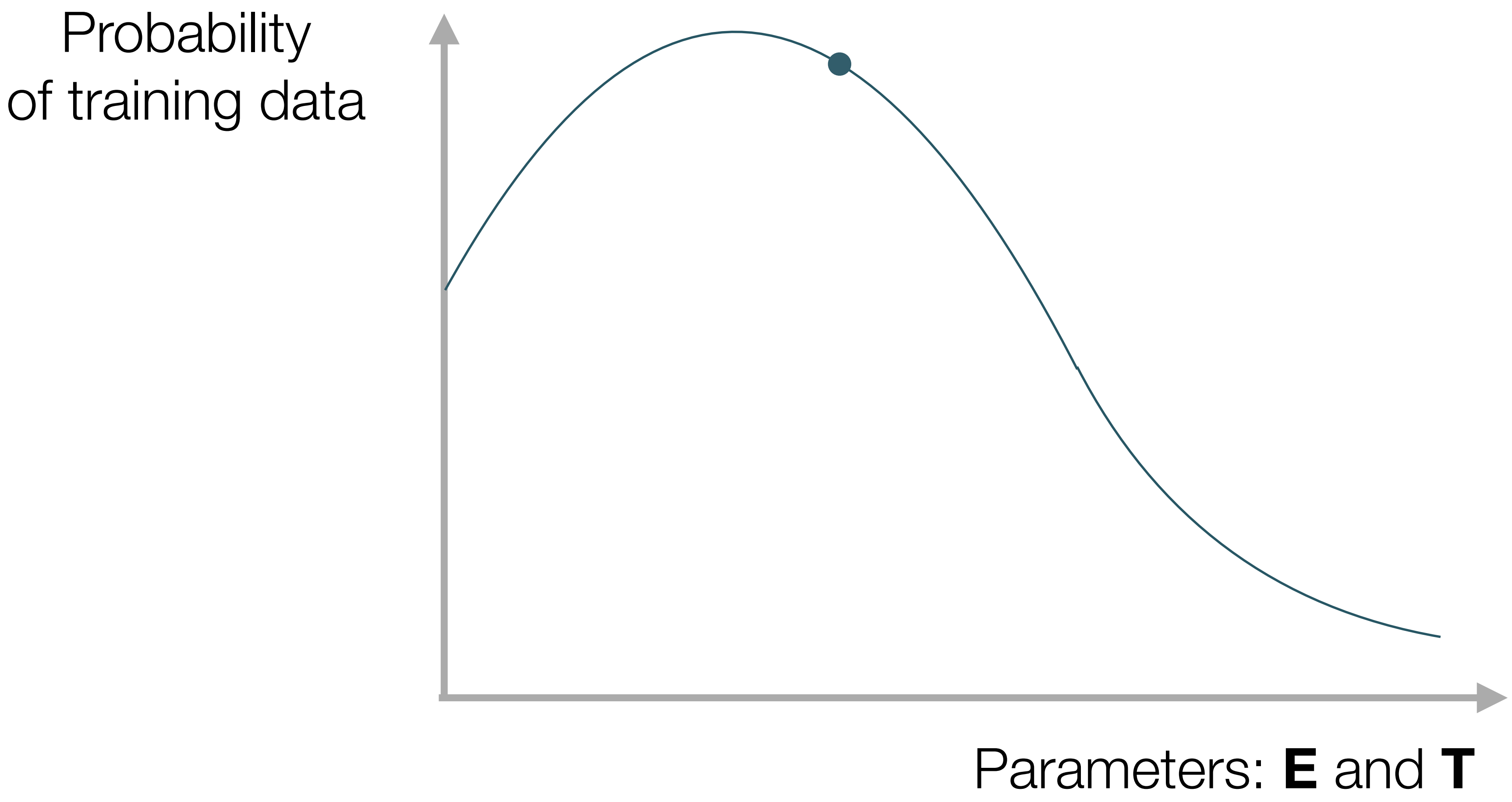
Learning by gradient methods

Every update is called a “training step”.



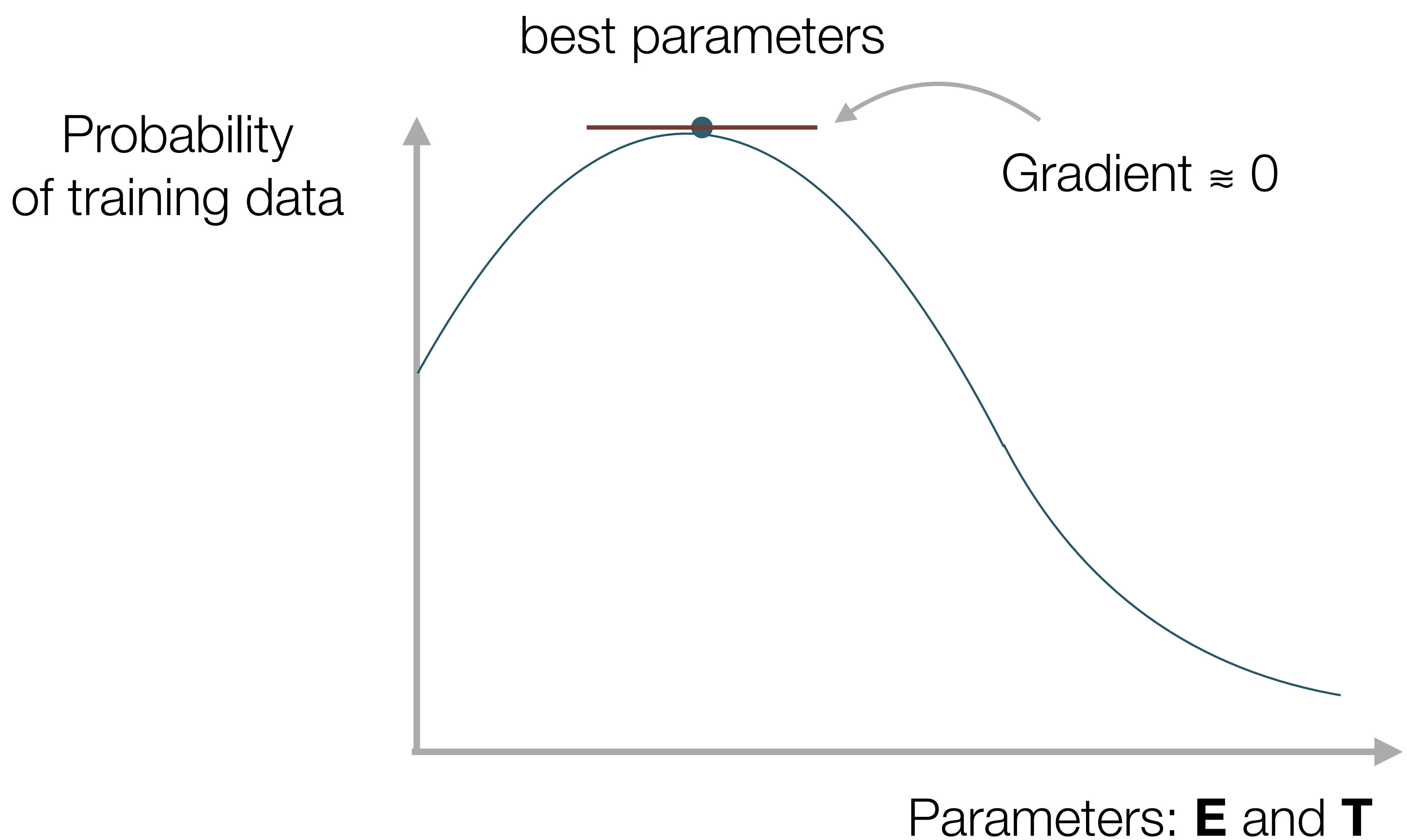
Learning by gradient methods

Every update is called a “training step”.



Learning by gradient methods

Every update is called a “training step”.



The learner only update **E** for subregular constraints.

Roadmap

1. Introduction
2. Framework
3. Evaluation and result

Datasets (Gouskova & Gallagher, 2020)

- Training data
    - Natural phonological words;
    - Navajo: 6279; Quechua: 10804;
    - 80% to training set, 20% to held-out set (unseen training data);
- Testing data
    - Nonce words, labelled as legal vs illegal based on nonlocal constraints;
    - Navajo: 5000; Quechua: 24352.

Navajo and Quechua

Navajo: the co-occurrence of alveolar and palatal strident is illegal;

s o s	*s o ʃ
s o r o s	*s o r o ʃ

Quechua: no stop can be followed by an ejective or aspirated stop;

t' o r o k	*t o r o k'
t^h o r o k	*t o r o k^h
t o r o k	*t^h o r o k

Evaluation metrics

- **Legal-illegal difference** in predicting nonce forms:

  → Ability of predicting nonlocal phonotactics;

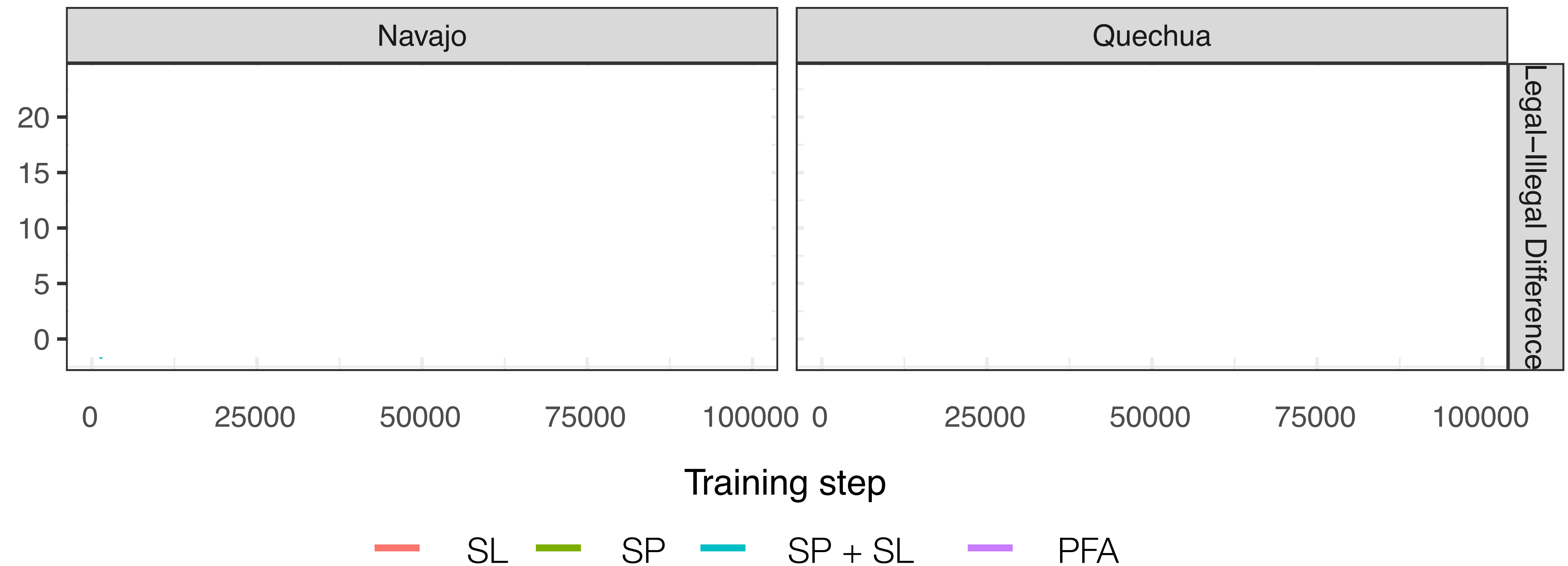
- **Log Likelihood** (LL) of held-out forms:

  → Ability of predicting other phonotactic patterns in any unseen training data;

Models in comparison

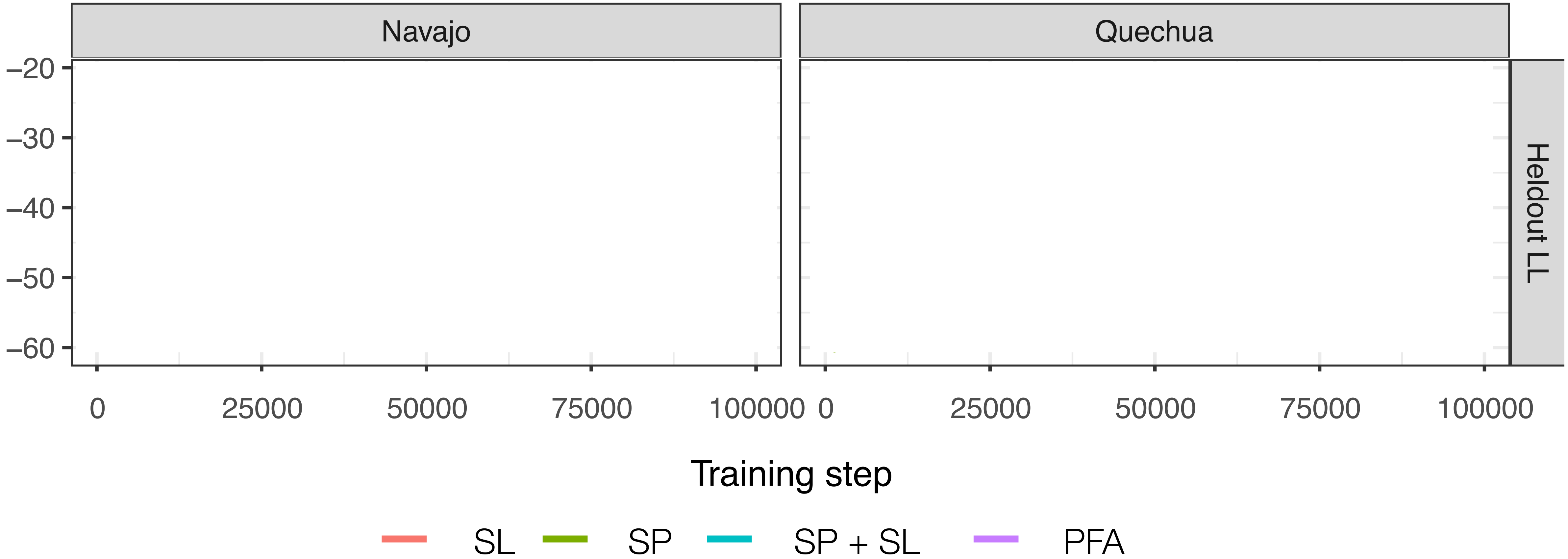
1. Nondeterministic PFA: most expressive
2. Deterministic PFAs
    - SP grammar
    - SL grammar
    - SP + SL grammar

Primary results: legal-illegal difference



Work in progress: compare this result with Gouskova & Gallagher (2020)

Primary results: held-out LL



Discussion

How should we interpret this result? Two possibilities:

1. There exists a undiscovered grammar in Chomsky Hierarchy that is restrictive enough but can also capture all the patterns;

2. It's also possible that the phonotactics patterns are so diverse that it's impossible to have "one true grammar, which might be another reason for a general framework as we proposed.

Conclusions

It's possible to compare the induction of various (sub)regular languages in a unified framework:

- Inducing unrestricted Probabilistic Finite-state Automata (PFAs) produces the best fit to naturalistic held-out forms;
- However, a restricted subregular model (Strictly Piecewise) is superior in capturing nonlocal constraints as evidenced in nonce data.

Acknowledgement

- Thanks to Adam Jardine, Adam McCollum, Jeff Heinz, three anonymous reviewers, and the audience at Rutgers PhonX group for helpful comments. Special thanks to Maria Gouskova, Gillian Gallagher, Seoyoung Kim for their help on the dataset.

- This work was supported by an NVIDIA GPU Grant to Richard Futrell.

- All our code is available at http://github.com/hutengdai/pfa-learner

Reference

Dai, H. (2021). Learning nonlocal phonotactics in Strictly Piecewise phonotactic model. *Proceedings of the Society for Computation in Linguistics*, *4*(1), 401-402.

Gouskova, M., & Gallagher, G. (2020). Inducing nonlocal constraints from baseline phonotactics. *Natural Language & Linguistic Theory*, *38*(1), 77-116.

Hayes, B., & Wilson, C. (2008). A maximum entropy model of phonotactics and phonotactic learning. *Linguistic inquiry*, *39*(3), 379-440.Jardine, A., & Heinz, J. (2016). Learning tier-based strictly 2-local languages. *Transactions of the Association for Computational Linguistics*, *4*, 87-98.

Heinz, J., & Rogers, J. (2010). Estimating strictly piecewise distributions. In Proceedings of the 48th annual meeting of the association for computational linguistics (pp. 886-896).

Heinz, J. (2010). Learning long-distance phonotactics. Linguistic Inquiry, 41(4), 623-661.

Jardine, A., & Heinz, J. (2016). Learning tier-based strictly 2-local languages. *Transactions of the Association for Computational Linguistics*, *4*, 87-98.

Lambert, D. (2021). Grammar Interpretations and Learning TSL Online. In *International Conference on Grammatical Inference* (pp. 81-91). PMLR.

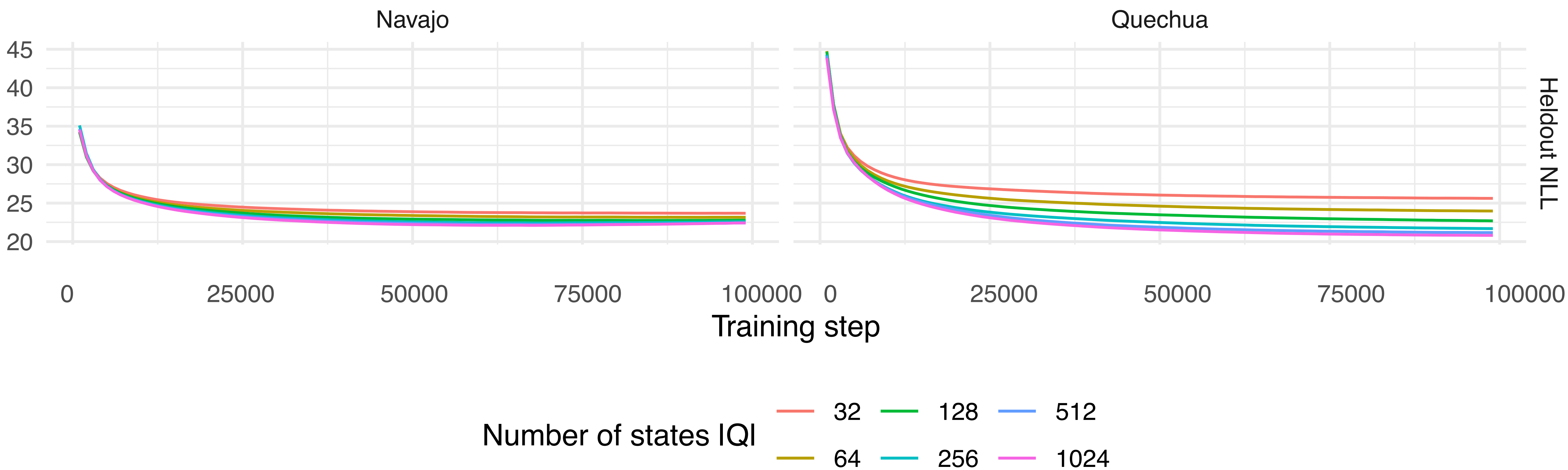
Rawski, J., & Heinz, J. (2019). No free lunch in linguistics or machine learning: Response to pater. *Language*, *95*(1), e125-e135.

Shibata, C., & Heinz, J. (2019). Maximum likelihood estimation of factored regular deterministic stochastic languages. In *Proceedings of the 16th Meeting on the Mathematics of Language* (pp. 102-113).

Vidal, E., Thollard, F., De La Higuera, C., Casacuberta, F., & Carrasco, R. C. (2005). Probabilistic finite-state machines-part I. *IEEE transactions on pattern analysis and machine intelligence*, *27*(7), 1013-1025.

Vidal, E., Thollard, F., De La Higuera, C., Casacuberta, F., & Carrasco, R. C. (2005). Probabilistic finite-state machines-part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(7), 1026-1039.

Unrestricted PFA Induction Results



Probabilistic Finite-State Automata (PFAs)

PFAs are parameterized by matrices **E** and $\mathbf{T}_x$.

Emission probability:
Conditional on state *q*, the probabilistic distribution on symbols

Transition probability:
Conditional on state *q* and symbol *x*, the probabilistic distribution on next states

$$p(\,\cdot \mid \mathbf{q}) = \mathbf{q}^\top \mathbf{E}$$

$$p(\,\cdot \mid \mathbf{q}, x) = \mathbf{q}^\top \mathbf{T}_x$$

State distribution: e.g.
[q_0: 0.45, q_1: 0.50, q_2: 0.05]

With $\mathbf{T}_x$, we no longer need to specify state transitions for PFAs in learning.

Regularization against nondeterminism

In Deterministic PFA (DPFA), state transition distribution is deterministic.

We penalize nondeterministic automata by using **average nondeterminism** as a regularization term:

$$N(\mathbf{E}, \mathbf{T}) = \mathrm{H}[q' \mid x, q]$$

Conditional entropy of next state given previous state and current symbol.

= 0 for perfectly deterministic PFAs.

Induction by Gradient Descent: Softmax

Update underlying weight matrices $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{T}}$, which are transformed into probabilistic matrices **E** and **T** by Softmax:

$$E_{ij} = \frac{\exp \tilde{E}_{ij}}{\sum_k \exp \tilde{E}_{ik}}, \ T_{ij} = \frac{\exp \tilde{T}_{ij}}{\sum_k \exp \tilde{T}_{ik}}$$

Running the program

```
python new_pfa.py --model_class sp_sl --lang navajo --activation softmax --print_every 1000 --lr 0.01
Training set size = 5023
Dev set size = 1256
Segment inventory size =  47
Model class = sp_sl
0
nondeterminism_penalty,memory_mi_penalty,init_temperature,activation,batch_size,lr,model_class,epoch,train_nll,dev_nll,
0.0,0.0,1,softmax,5,0.01,sp_sl,0,41.609500885009766,41.73338317871094,0.0,nan,39.65355246848905,39.63214659916812,
1000
0.0,0.0,1,softmax,5,0.01,sp_sl,1000,17.554428100585938,18.200313568115234,0.0,nan,34.43270141530961,34.7243515103281,
2000
0.0,0.0,1,softmax,5,0.01,sp_sl,2000,16.844873428344727,17.5587215423584,0.0,nan,35.587077134740774,36.448850729093024,
3000
0.0,0.0,1,softmax,5,0.01,sp_sl,3000,16.608205795288086,17.3580379486084,0.0,nan,36.85474407665988,37.8886514242108 7,
4000
0.0,0.0,1,softmax,5,0.01,sp_sl,4000,16.509923934936523,17.308490753173828,0.0,nan,38.25863582154379,39.46197283052953,
5000
0.0,0.0,1,softmax,5,0.01,sp_sl,5000,16.455812454223633,17.3162784576416,0.0,nan,39.804874700366454,41.104307318710475,
```

Induction by Gradient Descent: Objective

Find matrices **E** and **T** to minimize the training objective:

$$J(\tilde{\mathbf{E}}, \tilde{\mathbf{T}}) = \Big\langle -\log p(x \,|\, \mathbf{E}, \mathbf{T}) \Big\rangle_{x \sim X} + N(\mathbf{E}, \mathbf{T})$$

Average negative log likelihood of data

Regularization

Encoding subregular constraints

Implement 2-SP as the product of factor machines $A^{(x)}$, one per segment *x.*

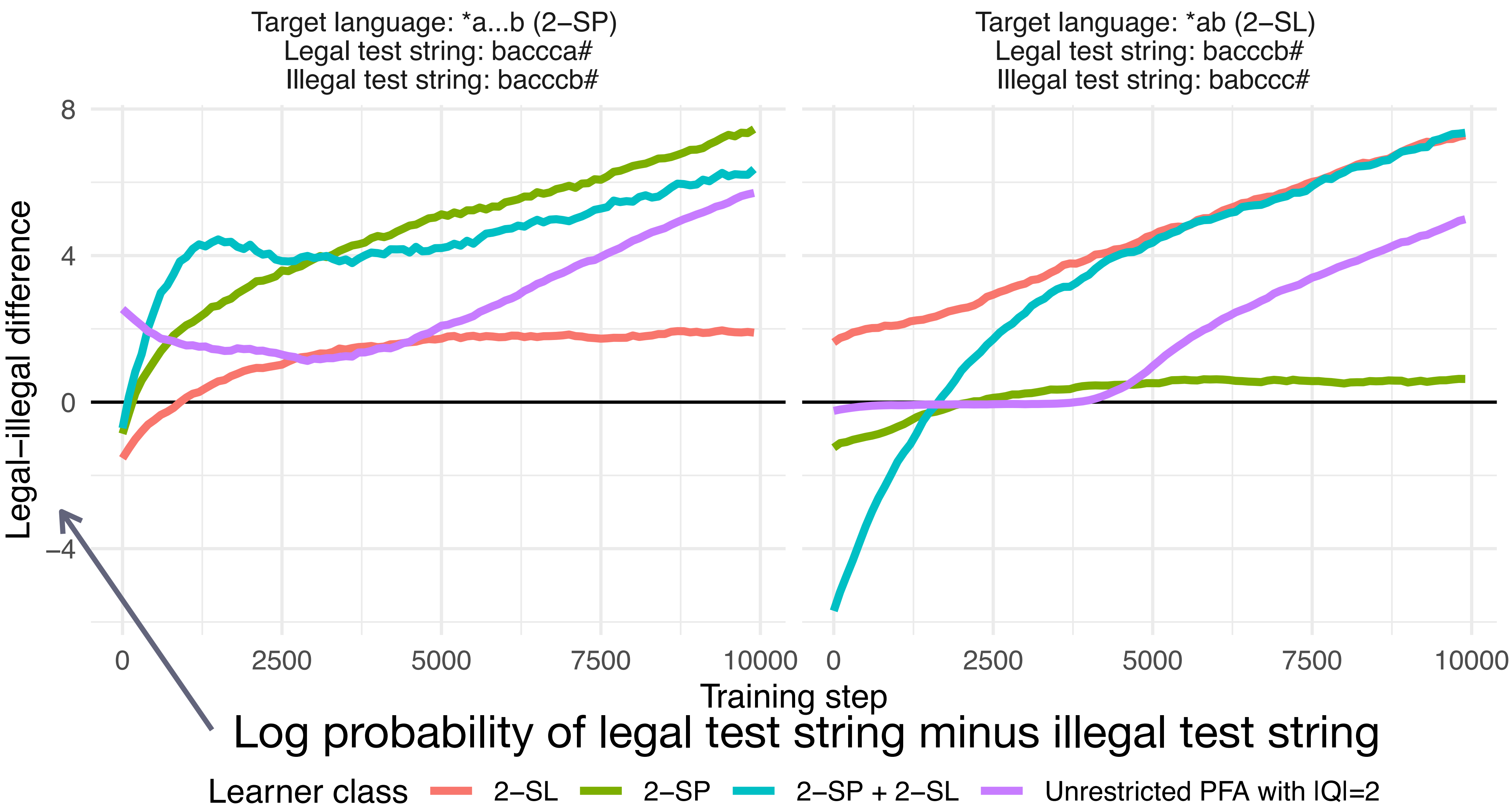
e.g. transition matrices for machine $A^{(x)}$ in 2-SP

$$\mathbf{T}_x^{(x)} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \mathbf{T}_{y \neq x}^{(x)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

"If you see *x*, move into the state corresponding to symbol *x*"

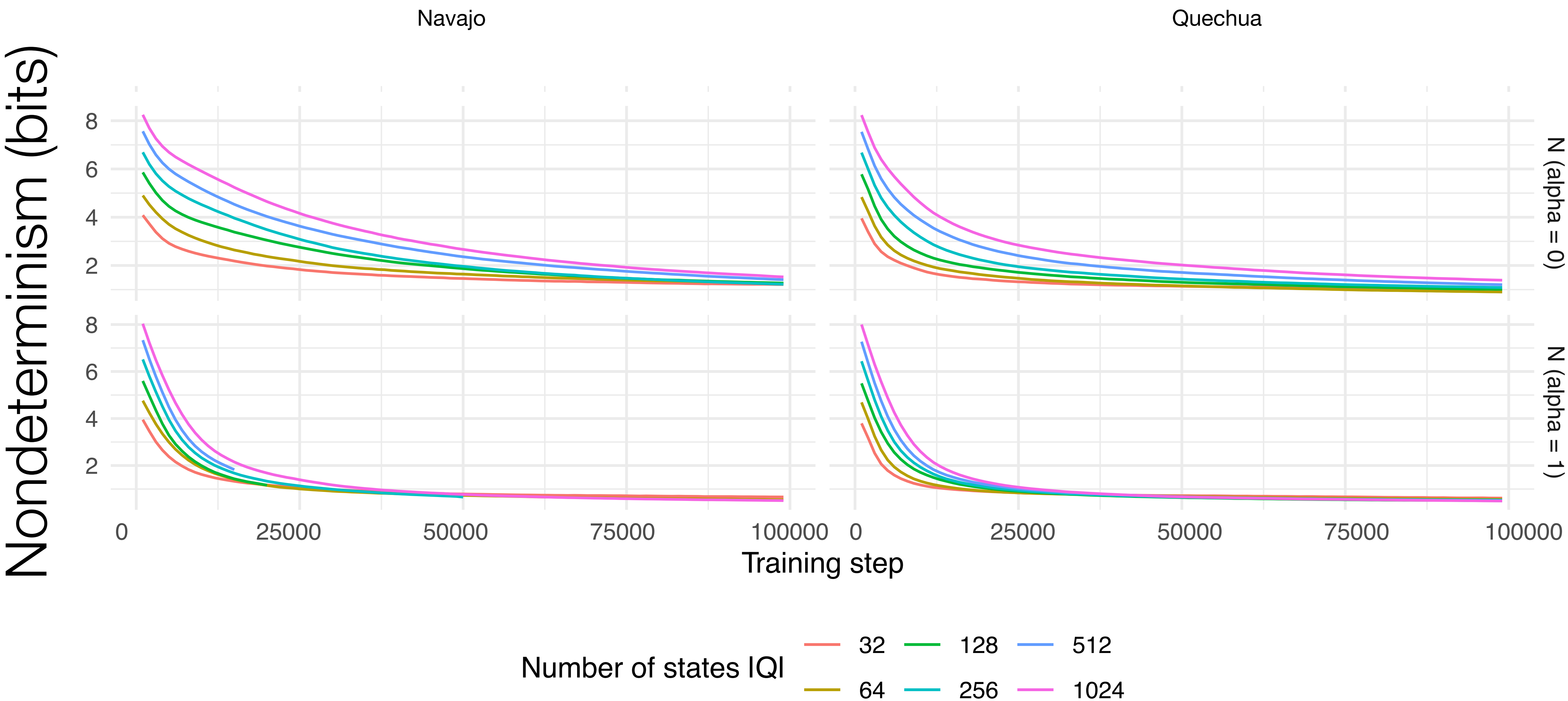
"If you do not see *x*, stay in the same state"

Evaluation: Toy languages



Log probability of legal test string minus illegal test string

The emergence of determinism



Encoding subregular constraints

Implement 2-SP as the product of factor machines $A^{(x)}$, one per segment *x.*

e.g. transition matrices for machine $A^{(x)}$ in 2-SP

$$\mathbf{T}_{x}^{(x)} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \mathbf{T}_{y \neq x}^{(x)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

"If you see *x*, move into the state corresponding to symbol *x*"

"If you do not see *x*, stay in the same state"