Simple induction of (deterministic) probabilistic finite-state automata for phonotactics by stochastic gradient descent

Huteng Dai
Rutgers University
huteng.dai@rutgers.edu

Richard Futrell
University of California, Irvine
rfutrell@uci.edu

Abstract

We introduce a simple and highly general phonotactic learner which induces a probabilistic finite-state automaton from word-form data. We describe the learner and show how to parameterize it to induce unrestricted regular languages, as well as how to restrict it to certain subregular classes such as Strictly k-Local and Strictly k-Piecewise languages. We evaluate the learner on its ability to learn phonotactic constraints in toy examples and in datasets of Quechua and Navajo. We find that an unrestricted learner is the most accurate overall when modeling attested forms not seen in training; however, only the learner restricted to the Strictly Piecewise language class successfully captures certain nonlocal phonotactic constraints. Our learner serves as a baseline for more sophisticated methods.

1 Introduction

Natural language phonotactics is argued to fall in the class of regular languages, or even in a smaller class of subregular languages (Rogers et al., 2013). This observation has motivated the study of probabilistic finite-state automata (PFAs) that generate these languages as models of phonotactics. Here we implement a simple method for the induction of PFAs for phonotactics from data, which can induce general regular languages in addition to languages in certain more restricted subclasses, for example, Strictly k-Local and Strictly k-Piecewise languages (Heinz, 2018; Heinz and Rogers, 2010). We evaluate our learner on corpus data from Quechua and Navajo, with a particular emphasis on the ability to learn nonlocal constraints.

We make both theoretical and empirical contributions. Theoretically, we present the differentiable linear-algebraic formulation of PFAs which enables learning of the structure of the automaton by gradient descent. In our framework, it is possible to induce an unrestricted automaton with a given number of states, or an automaton with hard-coded constraints representing various subregular languages. This work fills a gap in the formal linguistics literature, where learners have been developed within certain subregular classes (Shibata and Heinz, 2019; Heinz, 2010; Heinz and Rogers, 2010; Futrell et al., 2017), whereas our learner can in principle induce any (sub)regular language. In addition, we demonstrate how Strictly Local and Strictly Piecewise constraints can be encoded within our framework, and show how information-theoretic regularization can be applied to produce deterministic automata.

Empirically, our main result is to show that our approach gives reasonable and linguistically accurate results. We find that inducing an unrestricted PFA produces the best fit to held-out attested forms, while inducing an automaton for a Strictly 2-Piecewise language yields a model that successfully captures nonlocal constraints. We also analyze the nondeterminism of induced automata, and the extent to which induced automata overfit to their training data.

2 Model specification

2.1 Probabilistic Finite-state Automata

A **probabilistic finite-state automaton** (PFA) for generating sequences consists of a finite set of states Q, an inventory of symbols Σ, an **emission distribution** with probability mass function $p(x|q)$ which gives the probability of generating a symbol $x \in \Sigma$ given state $q \in Q$, and a **transition distribution** with probability mass function $p(q'|q, x)$ which gives the probability of transitioning into new state q' from state q after emission of symbol x.

We parameterize a PFA using a family of right-stochastic matrices. The **emission matrix E**, of

Proceedings of the Seventeenth SIGMORPHON Workshop on Computational Research

in Phonetics, Phonology, and Morphology,pages 167–176

shape $|Q| \times |\Sigma|$, gives the probability of emitting a symbol x given a state. Each row in the matrix represents a state, and each column represents an output symbol. Given a *distribution* on states represented as a stochastic vector $\mathbf{q}$, the probability mass function over symbols is:

$$p(\cdot|\mathbf{q}) = \mathbf{q}^\top \mathbf{E}. \tag{1}$$

Each symbol $x \in \Sigma$ is associated with a right-stochastic **transition matrix** $\mathbf{T}_x$ of shape $|Q| \times |Q|$, so that the probability distribution on following states given that the symbol x was emitted from the distribution on states $\mathbf{q}$ is

$$p(\cdot|\mathbf{q}, x) = \mathbf{q}^\top \mathbf{T}_x. \tag{2}$$

Generation of a particular sequence $\mathbf{x} \in \Sigma^*$ works by starting in a distinguished **initial state** q_0, generating a symbol x, transitioning into the next state q', and so on recursively until reaching a distinguished **final state** q_f. Given a PFA parameterized by matrices $\mathbf{E}$ and $\mathbf{T}$, the probability of a sequence $x_{t=1}^N$ marginalizing over all trajectories through states can be calculated according to the Forward algorithm (Baum et al., 1970; Vidal et al., 2005a, §3) as follows:

$$p(x_{t=1}^N|\mathbf{E}, \mathbf{T}) = f(x_{t=1}^N|\delta_{q_0}),$$

where δ_q is a one-hot coordinate vector on state q and

$$f(\emptyset|\mathbf{q}) = \delta_{q_f}^\top \mathbf{q}$$
$$f(x_{t=1}^n|\mathbf{q}) = p(x_1|\mathbf{q}) \cdot f(x_{t=2}^n|\mathbf{q}^\top \mathbf{T}_{x_1}).$$

The important aspect of this formulation is that the probability of a sequence is a differentiable function of the matrices $\mathbf{E}$ and $\mathbf{T}$ that define the PFA. Because the probability function is differentiable, we can induce a PFA from a set of training sequences by using gradient descent to search for matrices that maximize the probability of the training sequences.

2.2 Learning by gradient descent

We describe a simple and highly general method for inducing a PFA from data by stochastic gradient descent. Although more specialized learning algorithms and heuristics exist for special cases (see for example Vidal et al., 2005b, §3), ours has the advantage of generality. Our goal is to see how effective this simple approach can be in practice.

Given a data distribution X with support over Σ^*, we wish to learn a PFA by finding parameter matrices $\mathbf{E}$ and $\mathbf{T}$ to minimize an objective function of the form

$$J(\mathbf{E}, \mathbf{T}) = \langle -\log p(x|\mathbf{E}, \mathbf{T}) \rangle_{x \sim X} + C(\mathbf{E}, \mathbf{T}), \tag{3}$$

where $\langle \cdot \rangle_{x \sim X}$ indicates an average over values x drawn from the data distribution X, and $-\log p(x|\mathbf{E}, \mathbf{T})$ is the **negative log likelihood** (NLL) of a sample x under the model; the average negative log likelihood is equivalent to the **cross entropy** of the data distribution X and the model. By minimizing cross-entropy, we maximize likelihood and thus fit to the data. The term $C(\mathbf{E}, \mathbf{T})$ represents additional complexity constraints on the $\mathbf{E}$ and $\mathbf{T}$ matrices, discussed in Section 2.4. When C is interpreted as a log prior probability on automata, then minimizing Eq. 3 is equivalent to fitting the model by maximum a posteriori.

Given the formulation in Eq. 3, because the objective function is differentiable, we can search for the optimal matrices $\mathbf{E}$ and $\mathbf{T}$ by performing (stochastic) descent on the gradients of the objective. That is, for a parameter matrix $\mathbf{X}$, we can search for a minimum by performing updates of the form

$$\mathbf{X}' = \mathbf{X} - \eta \nabla J(\mathbf{X}), \tag{4}$$

where the scalar η is the **learning rate**. In stochastic gradient descent, each update is performed using a random finite sample from the data distribution, called a **minibatch**, to approximate the average over the data distribution in Eq. 3.

However, we cannot apply these updates directly to the matrices $\mathbf{E}$ and $\mathbf{T}$ because they must be right-stochastic, meaning that the entries in each row must be positive and sum to 1. There is no guarantee that the output of Eq. 4 would satisfy these constraints. This issue was addressed by Dai (2021) by clipping the values of the matrix $\mathbf{E}$ to be between 0 and 1. A more general solution is that, instead of doing optimization on the $\mathbf{E}$ and $\mathbf{T}$ matrices directly, we instead do optimization over underlying real-valued matrices $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{T}}$ such that

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

in other words we derive the matrices $\mathbf{E}$ and $\mathbf{T}$ by applying the **softmax** function to underlying matrices $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{T}}$, whose entries are called logits. Gradient descent is then done on the objective as

a function of the logit matrices $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{T}}$. This approach to parameterizing probability distributions is standard in machine learning. Applied to induce a PFA with states Q and symbol inventory Σ, our formulation yields a total of $|Q| \times (|Q| \times |\Sigma| - 1)$ meaningful trainable parameters.

We note that this procedure is not guaranteed to find an automaton that globally minimizes the objective when optimizing $\mathbf{T}$ (see Vidal et al., 2005b, §3). But in practice, stochastic gradient descent in high-dimensional spaces can avoid local minima, functioning as a kind of annealing (Bottou, 1991, §4); using these simple optimization techniques on non-convex objectives is now standard practice in machine learning.

2.3 Sequence representation and word boundaries

In order to model phonotactics, a PFA must be sensitive to the boundaries of words, because there are often constraints that apply only at word beginnings or endings (Hayes and Wilson, 2008; Chomsky and Halle, 1968). In order to account for this, we include in the symbol inventory Σ a special word boundary delimiter $\#$, which occurs as the final symbol of each word, and which only occurs in that position. Furthermore, we constrain all matrices $\mathbf{T}$ to transition deterministically back into the initial state following the symbol $\#$, effectively reusing the initial state q_0 as the final state q_f.

By constructing the automata in this way, we ensure that their long-run behavior is well-behaved. If an automaton of this form is allowed to keep generating past the symbol $\#$, it will generate successive concatenated independent and identically distributed samples from its distribution over words, with boundary symbols $\#$ delineating them. This construction makes it possible to calculate stationary distributions over states and complexity measures related to them.

2.4 Regularization

The objective in Eq. 3 includes a regularization term C representing complexity constraints. Any differentiable complexity measure could be used here. This regularization term can be viewed from a Bayesian perspective as defining a prior over automata, and providing an inductive bias. We propose to use this term to constrain the PFA induction process to produce deterministic automata.

Most formal work on probabilistic finite-state automata for phonology has focused on *deterministic* PFAs because of their nice theoretical properties (Heinz, 2010). A deterministic PFA is distinguished by having fully deterministic transition matrices $\mathbf{T}$. This condition can be expressed information-theoretically. Assuming $0 \log 0 = 0$, letting the entropy of a stochastic vector $\mathbf{p}$ be:

$$H[\mathbf{p}] = -\sum_i p_i \log p_i,$$

a PFA is deterministic when it satisfies the condition $H[\mathbf{q}^\top \mathbf{T}_x] = 0$ for all symbols x and state distributions $\mathbf{q}$.

We can use this expression to monitor the degree of nondeterminism of a PFA during optimization, or to add a determinism constraint to the objective in Section 2.2. The average **nondeterminism** N of a PFA is given by

$$N(\mathbf{E}, \mathbf{T}) = \sum_{ij} \hat{q}_i E_{ij} H[\delta_{q_i}^\top \mathbf{T}_j],$$

where $\hat{\mathbf{q}}$ is the **stationary distribution** over states, representing the long-run average occupancy distribution over states. The stationary distribution $\hat{\mathbf{q}}$ is calculated by finding the left eigenvector of the matrix $\mathbf{S}$ satisfying

$$\hat{\mathbf{q}}^\top \mathbf{S} = \hat{\mathbf{q}},$$

where $\mathbf{S}$ is a right stochastic matrix giving the probability that a PFA transitions from state i to state j marginalizing over symbols emitted:

$$S_{ij} = \sum_{x \in \Sigma} p(x|q_i) p(q_j|q_i, x).$$

For the Strictly Local and Strictly Piecewise automata, $N = 0$ by construction. For an automaton parameterized by $\mathbf{T} = \text{softmax}(\tilde{\mathbf{T}})$, it is not possible to attain $N = 0$, but nonetheless N can be made arbitrarily small. There are alternative parameterizations where $N = 0$ is achievable, for example using the sparsemax function instead of softmax (Martins and Astudillo, 2016; Peters et al., 2019).

In order to constrain automata to be deterministic, we set the regularization term in Eq. 3 to be

$$C = \alpha N,$$

where α is a non-negative scalar determining the strength of the trade-off of cross entropy and nondeterminism in the optimization. With $\alpha = 0$ there is no constraint on the nondeterminism of the automaton, and minimizing the objective in Eq. 3 reduces to maximum likelihood estimation.

2.5 Implementing restricted automata

We define Strictly Local and Strictly Piecewise automata as automata that generate the respective languages. We implement Strictly Local and Strictly Piecewise automata by hard-coding the transition matrices $\mathbf{T}$. For these automata, we only do optimization over the emission matrices $\mathbf{E}$.

Strictly Local In a Strictly k-Local (k-SL) language, each symbol is conditioned only on *immediately preceding* $k-1$ symbol(s) (Heinz, 2018; Rogers and Pullum, 2011). We implement a 2-SL automaton by associating each state $q \in Q$ with a unique element x in the symbol inventory Σ. Upon emitting symbol x, the automaton deterministically transitions into the corresponding state, denoted q_x. Thus the transition matrices have the form

$$\mathbf{T}_x = \begin{array}{c} \begin{matrix} \cdots q_{\neq x} \cdots & q_x & \cdots q_{\neq x} \cdots \end{matrix} \\ \begin{bmatrix} \vdots & \vdots & \vdots \\ \cdots 0 \cdots & 1 & \cdots 0 \cdots \\ \vdots & \vdots & \vdots \end{bmatrix} \end{array}.$$

This construction can be straightforwardly extended to k-SL, yielding $|\Sigma|^{k-1} \times (|\Sigma| - 1)$ trainable parameters for a k-SL automaton.

Strictly Piecewise A Strictly k-Piecewise k-SP) language, each symbol depends on the presence of any preceding $k-1$ symbols at arbitrary distance (Heinz, 2007, 2018; Shibata and Heinz, 2019). For example, in a 2-SP language, in a string *abc*, *c* would be conditional on the presence of *a* and the presence of *b*, without regard to distance nor the relative order of *a* and *b*.

The implementation of an SP automaton is slightly more complex than the SL automaton, as the number of states required in a naïve implementation is exponential in the symbol inventory size, resulting in intractably large matrices. We circumvent this complexity by parameterizing a 2-SP automaton as a product of simpler automata. We associate each symbol $x \in \Sigma$ with a sub-automaton A_x which has two states q_0^x and q_1^x, with state q_0^x indicating that the symbol x has not been seen, and q_1^x indicating that it has been seen. Each sub-automaton A_x has an emission matrix $\mathbf{E}^{(x)}$ of size $2 \times |\Sigma|$ corresponding to the two states q_0^x and q_1^x; the emission matrix for all states q_0^x is constrained to be the uniform distribution over symbols. The transition matrices $\mathbf{T}^{(x)}$ are

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

Then the probability of the t'th symbol in a sequence x_t given a context of previous symbols $x_{i=1}^{t-1}$ is the geometric mixture of the probability of x_t under each sub-automaton, also called the co-emission probability

$$p(x_t | x_{i=1}^{t-1}) \propto \prod_{y=1}^{|\Sigma|} p_{A_y}(x_t | x_{i=1}^{t-1}).$$

Because each sub-automaton A_y is deterministic, its state after seeing the context $x_{i=1}^{t-1}$ is known, and the conditional probability $p_{A_y}(x_t | x_{i=1}^{t-1})$ can be computed using Eq. 1. For calculating the probability of a sequence, we assume an initial state of having seen the boundary symbol $\#$; that is, the sub-automaton $A_\#$ starts in state $q_1^\#$.

Using this parameterization, we can do optimization over the collection of emission matrices $\{\mathbf{E}^{(x)}\}_{x \in \Sigma}$. This construction yields $|\Sigma| \times (|\Sigma| - 1)$ trainable parameters for the 2-SP automaton, the same number of parameters as the 2-SL automaton.

SP + SL It is also possible to create and train an automaton with the ability to condition on both 2-SL and 2-SP factors by taking the product of 2-SL and 2-SP automata, as proposed by Heinz and Rogers (2013). We refer to the language generated by such an automaton as 2-SL + 2-SP. We experiment with such product machines below.

2.6 Related work

PFA induction from data is a well-studied task which has been the subject of multiple competitions over the years (see Verwer et al., 2012, for a review). The most common approaches are variants of Baum-Welch and heuristic state-merging algorithms (see for example de la Higuera, 2010). Gibbs samplers and spectral methods have also been proposed (Gao and Johnson, 2008; Bailly, 2011; Shibata and Yoshinaka, 2012). Induction of restricted PDFAs, especially for SL and SP languages, is explored in Heinz and Rogers (2013, 2010)

Our work differs from previous approaches in its simplicity. Inspired by Shibata and Heinz (2019), we optimize the training objective directly via gradient descent, without approximations or heuristics other than the use of minibatches. The same algorithm is applied to learn both transition and emission structure, for learning of both general PFAs and restricted PDFAs. One of our contributions

is to show that this very simple approach gives reasonable results for learning phonotactics.

3 Inducing toy languages

First, we test the ability of the model to recover automata for simple examples of subregular languages. We do so for the two subregular classes 2-SL and 2-SP described in Section 2.5. For each of these language classes, we implement a reference PFA which generates strings from a simple example language in that class, then generate $10,000$ sample sequences from the reference PFA. We then use these samples as training data, and study whether our learners can recover the relevant constraints from the data.

3.1 Evaluation

We evaluate the ability to induce appropriate automata in two ways. First, since we are studying very simple languages and automata, it is possible to directly inspect the $\mathbf{E}$ and $\mathbf{T}$ matrices and check that they implement the correct automaton by observing the transition and emission probabilities.

Second, we study the probabilities assigned to carefully selected strings which exemplify the constraints that define the languages. For each language, we define an **illegal test string** which violates the constraints of the language, and a minimally-different **legal test string**. Given an automaton, we can measure the **legal–illegal difference**: the log probability of the legal test string minus the log probability of the illegal test string. A larger legal–illegal difference indicates that the model is assigning a higher probability to the legal form compared to the illegal one and therefore is successfully learning the constraints represented by the testing data.

3.2 Languages

All languages are defined over the symbol inventory $\{a, b, c\}$ plus the boundary symbol #.

As an exemplar of 2-SL languages, we use the language characterized by the forbidden factor **ab*. A deterministic PFA for the language is given in Figure 1 (top). The language contains all strings that do not have an *a* followed immediately by a *b*. Our legal test string for this language is *bacccb#* and the illegal test string is *babccc#*.

As an exemplar of 2-SP languages, we use the language characterized by a forbidden factor **a*...*b*. This language contains all strings that do not have an *a* followed by a *b* at any distance. The reference automaton is given in Figure 1 (bottom). The legal test string is *baccca#* and the illegal test string is *bacccb#*.

3.3 Training parameters

The logit matrices $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{T}}$ are initialized with random draws from a standard Normal distribution (Derrida, 1981). We perform stochastic gradient descent using the Adam algorithm, which adaptively sets the learning rate (Kingma and Ba, 2015). We perform $10,000$ update steps with starting learning rate $\eta = 0.001$ and minibatch size 5.

3.4 Results

Unrestricted PFA induction succeeds in recovering the reference automata for both toy languages. Learners restricted to the appropriate classes, as well as the automaton combining SL and SP factors, also succeed in inducing the appropriate automata, while learners restricted to the 'wrong' class fail.

Figure 1 shows the legal–illegal differences for test strings over the course of training. We can see that, when the learner is unrestricted or when the learner is in the appropriate class, it eventually picks up on the relevant constraint, with the legal–illegal difference increasing apparently without bound over training. Unrestricted learners take longer to reach this point, but they reach it reliably. On the other hand, looking at the legal–illegal differences for learners in the wrong class, we see that they asymptote to a small number and stop improving.

These results demonstrate that our simple method for PFA induction does succeed in inducing certain simple structures relevant for modeling phonotactics in a small, controlled setting. Next, we turn to induction of phonotactics from corpus data.

4 Corpus experiments

We evaluate our learner by training it on dictionary forms from Quechua and Navajo and then studying its ability to predict attested forms that were held out in training in addition to artificially constructed nonce forms which probe the ability of the model to represent nonlocal constraints.

4.1 Training parameters

All training parameters are as in Section 3.3, except that we train for $100,000$ steps, and control the

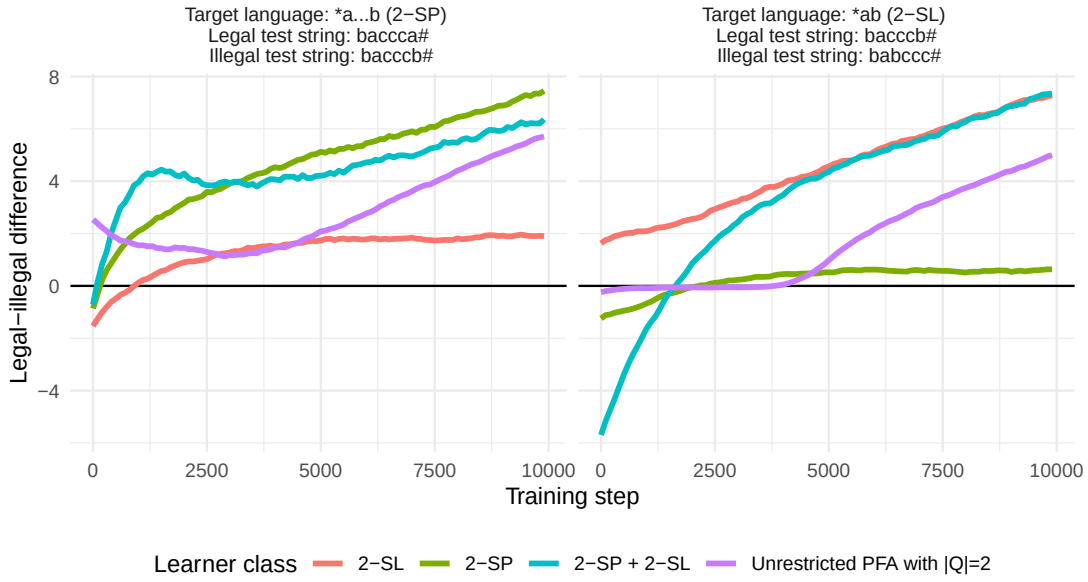

Figure 1: Difference in log probabilities for legal and illegal forms over the course of PFA induction for toy languages. A large positive value indicates that the relevant constraint has been learned.

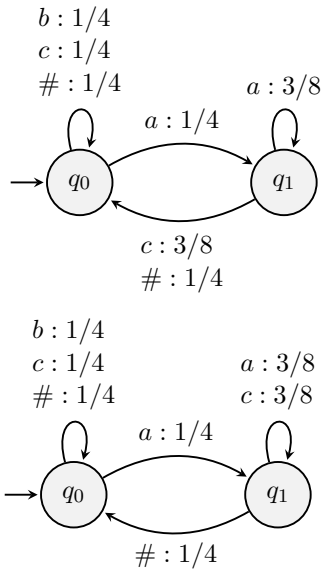

Figure 2: Reference automata for the 2-SL language characterized by the constraint **ab* (top) and the 2-SP language characterized by the constraint **a…b* (bottom). Arcs are annotated with symbols emitted and their corresponding emission probabilities.

succession of minibatches to be the same across models within the same language.

4.2 Dataset

The proposed learner is applied to the datasets of Navajo and Quechua (Gouskova and Gallagher, 2020), in which nonlocal phonotactics are attested.

In Navajo, the co-occurrence of alveolar and palatal strident is illegal. The learning data of Navajo includes 6,279 Navajo phonological words; we divide this data into a training set of 5,023 forms and a held-out set of 1,256 forms. The nonce testing data of Navajo consists of 5,000 generated nonce words, which were labelled as illegal ($N = 3,271$) and legal ($N = 1,729$) based on whether the nonlocal phonotactics are satisfied.

In Quechua, any stop cannot be followed by an ejective or aspirated stop at any distance. The learning data of Quechua includes 10,804 phonological words, which we separate into 8,643 training forms and 2,160 held-out forms. The testing data of Quechua (Gouskova and Gallagher, 2020) consists of 24,352 nonce forms which were manually classified as legal ($N = 18,502$) and illegal ($N = 5,810$, including stop-aspirate and stop-ejective pairs).

4.3 Dependent Variables

For the linguistic performance of the classifier, we study two main dependent variables. First, the average held-out negative log likelihood (NLL) indicates the ability of the model to assign high probabilities to unseen but attested forms—low NLL indicates higher probabilities. Second, using our nonce forms dataset, we measure the extent to which the model can differentiate the legal forms from the illegal forms using the *difference* in log likelihood for the legal forms minus the illegal forms. This is the same as the legal–illegal

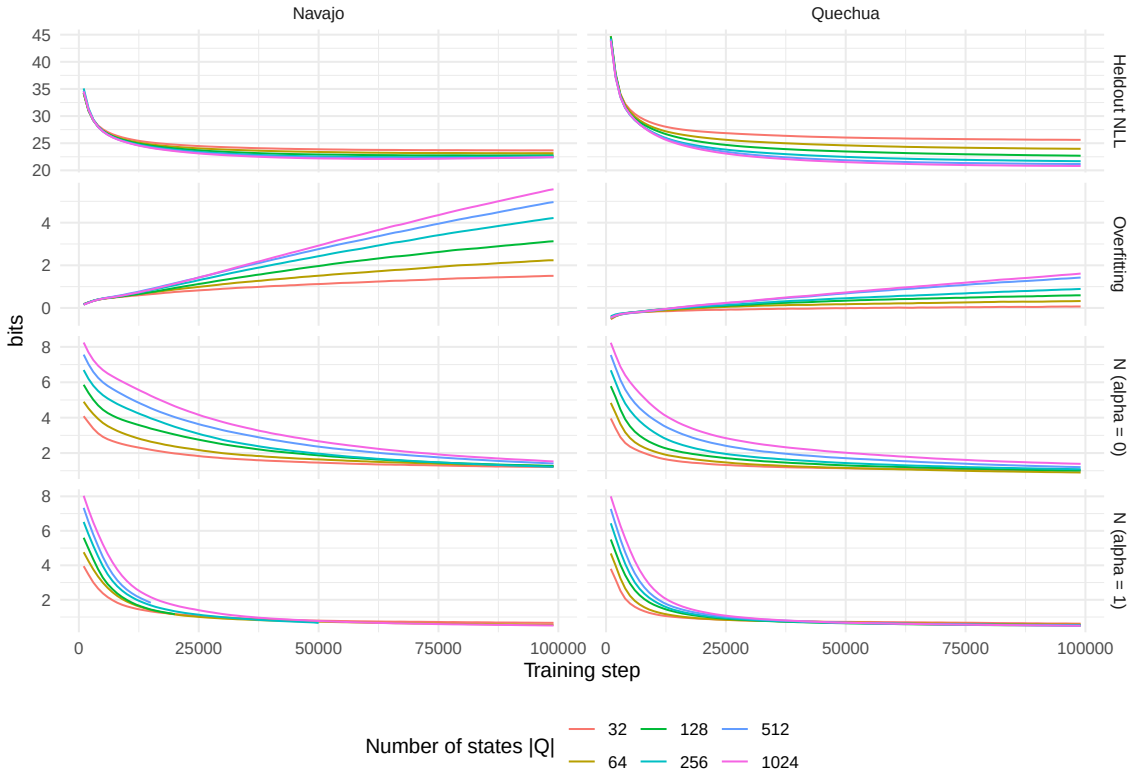

Figure 3: Accuracy and complexity metrics for unrestricted PFA induction. 'Overfitting' is the difference between held-out NLL and training set NLL. N is nondeterminism and alpha is the regularization parameter α (see Section 2.4). Runs with $|Q| = 128, 256, 512$ and $\alpha = 1$ on Navajo data terminated early due to numerical underflow in the calculation of the stationary distribution.

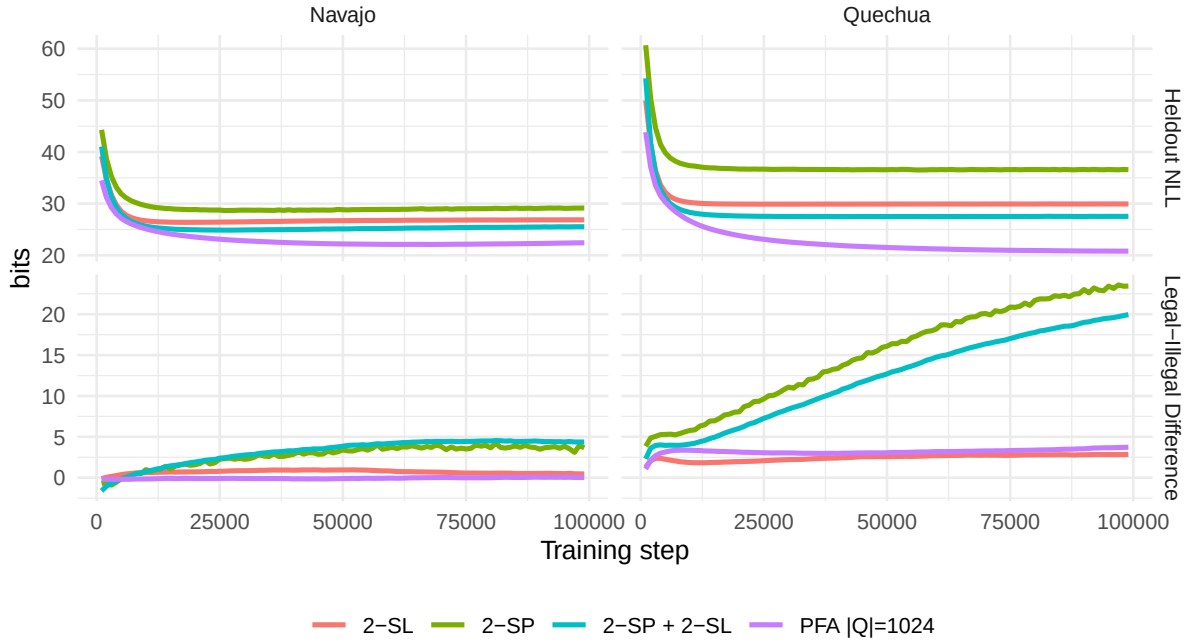

Figure 4: Performance of a 2-SP automaton, a 2-SL automaton, a 2-SP + 2-SL product automaton, and an unrestricted PFA with $1,024$ states and $\alpha = 0$. 'Heldout NLL' is the average NLL of a form in the set of attested forms never seen during training. 'Legal–illegal difference' is the difference in log likelihood between 'legal' and 'illegal' forms in the nonce test set.

difference described in Section 3.1, but now as an average over many legal–illegal nonce pairs instead of a difference for one pair.

4.4 Results

Unrestricted PFA induction Figure 3 shows results from induction of unrestricted PFAs with various numbers of states. We find that show the average NLL of forms in the heldout data, as well as 'overfitting', defined as the average held-out NLL minus the average training set NLL. This number shows the extent to which the model assigns higher probabilities to forms in the training set as opposed to the held-out set, an index of overfitting. We find that automata with more states fit the data better, but are also more prone to overfitting to the training set.

In Figure 3 (bottom two rows) we also show the measured nondeterminism N of the induced automata throughout training, for different values of the regularization parameter α (see Section 2.4). We find that, even without an explicit constraint for determinism, the induced PFAs tend towards determinism over time, with N reaching around 1.5 bits by the final training step. Explicit regularization (with $\alpha = 1$) makes this process faster, with N reaching around 0.5 bits. Regularization for determinism has only a minimal effect on the NLL values.

Linguistic performance and restricted models Figure 4 shows held-out NLL and the legal–illegal difference for both languages, comparing the SL automaton, the SP automaton, the product SP + SL automaton, and a PFA with $1,024$ states and $\alpha = 0$.

In terms of the ability to predict attested held-out forms, the best model is consistently the unrestricted PFA, with the SP automaton performing the worst. However, in terms of predicting the ill-formedness of artificial forms violating nonlocal phonotactic constraints, the best model is either the SP automaton or the SP + SL product automaton. Both of these automata successfully induce the nonlocal constraint.

On the other hand, the unrestricted PFA learner shows no evidence at all of having learned the difference between legal and illegal forms in the artificial data, despite having the capacity to do so in theory, and despite succeeding in inducing a 2-SP language in Section 3.

4.5 Discussion

We find that an unrestricted PFA learner performs most accurately when predicting real held-out forms, while an SP learner is most effective in learning certain nonlocal constraints. In fact, in terms of its ability to model the nonlocal constraints, the PFA learner ends up comparable to an SL learner, which cannot learn the constraints at all. Meanwhile, the SP learner, which is unable to model *local* constraints, fares much worse than even the SL learner on predicting held-out forms. The product SP + SL learner combines the strengths of both restricted learners, but still does not assign as high probability to the real held-out forms as the unrestricted PFA learner.

This pattern of performance suggests that the PFA learner is using most of its states to model local constraints beyond those captured in a 2-SL language. These constraints are important for predicting real held-out forms. The SP automaton is unable to achieve strong performance on held-out forms without the ability to model these local constraints. On the other hand, the unrestricted PFA tends to overfit to its training data, perhaps explaining its failure to detect nonlocal constraints which are picked up by the appropriate restricted automata.

5 Conclusion

We introduced a framework for phonotactic learning based on simple induction of probabilistic finite-state automata by stochastic gradient descent. We showed how this framework can be used to learn unrestricted PFAs, in addition to PFAs restricted to certain formal language classes such as Strictly Local and Strictly Piecewise, via constraints on the transition matrices that define the automata. Furthermore, we showed that the framework is successful in learning some phonotactic phenomena, with unrestricted automata performing best in a wide-coverage evaluation on attested but held-out forms, and Strictly Piecewise automata performing best in a targeted evaluation using nonce forms focusing on nonlocal constraints.

Our results leave open the question of whether the unrestricted learner or one of the restricted learners is 'best' for learning phonotactics, since they perform differently on different metrics. A key question for future work is whether there might be some model that could do well in inducing *both* local and nonlocal constraints simultaneously, and

performing well on both the held-out evaluation and the nonce form evaluation. Such a model could come in the form of another restricted language class such as Tier-Based Strictly Local languages (Heinz et al., 2011; Jardine and Heinz, 2016; McMullin, 2016; Jardine and McMullin, 2017), or perhaps in the form of a regularization term in the training objective which enforces an inductive bias that favors certain nonlocal interactions.

The code for this project is available at `http://github.com/hutengdai/PFA-learner`.

Acknowledgments

This work was supported by a GPU Grant from the NVIDIA corporation. We thank the three anonymous reviewers and Adam Jardine, Jeff Heinz, and Dakotah Lambert for their comments.

References

Raphael Bailly. 2011. Quadratic weighted automata: Spectral algorithm and likelihood maximization. In *Asian Conference on Machine Learning*, pages 147–163. PMLR.

Leonard E. Baum, Ted Petrie, George Soules, and Normal Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171.

Léon Bottou. 1991. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nimes*, 91(8):12.

Noam Chomsky and Morris Halle. 1968. *The Sound Pattern of English*. Harper & Row.

Huteng Dai. 2021. Learning nonlocal phonotactics in Strictly Piecewise phonotactic model. In *Proceedings of the 2020 Annual Meeting on Phonology*.

Colin de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Bernard Derrida. 1981. Random-energy model: An exactly solvable model of disordered systems. *Physical Review B*, 24(5):2613.

Richard Futrell, Adam Albright, Peter Graff, and Timothy J. O'Donnell. 2017. A generative model of phonotactics. *Transactions of the Association for Computational Linguistics*, 5:73–86.

Jianfeng Gao and Mark Johnson. 2008. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 344–352, Honolulu, Hawaii. Association for Computational Linguistics.

Maria Gouskova and Gillian Gallagher. 2020. Inducing nonlocal constraints from baseline phonotactics. *Natural Language & Linguistic Theory*, 38(1):77–116.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39(3):379–440.

Jeffrey Heinz. 2007. *The inductive learning of phonotactic patterns*. Ph.D. thesis, PhD dissertation, University of California, Los Angeles.

Jeffrey Heinz. 2010. Learning long-distance phonotactics. *Linguistic Inquiry*, 41(4):623–661.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. *Phonological Typology, Phonetics and Phonology*, pages 126–195.

Jeffrey Heinz, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based Strictly Local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 58–64, Portland, Oregon, USA. Association for Computational Linguistics.

Jeffrey Heinz and James Rogers. 2010. Estimating Strictly Piecewise distributions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 886–896, Uppsala, Sweden. Association for Computational Linguistics.

Jeffrey Heinz and James Rogers. 2013. Learning subregular classes of languages with factored deterministic automata. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 64–71, Sofia, Bulgaria. Association for Computational Linguistics.

Adam Jardine and Jeffrey Heinz. 2016. Learning Tier-based Strictly 2-Local languages. *Transactions of the Association for Computational Linguistics*, 4:87–98.

Adam Jardine and Kevin McMullin. 2017. Efficient learning of Tier-based Strictly k-Local languages. In *International Conference on Language and Automata Theory and Applications*, pages 64–76. Springer.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, San Diego, CA.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International*

Conference on Machine Learning, pages 1614–1623. PMLR.

Kevin James McMullin. 2016. *Tier-based locality in long-distance phonotactics: learnability and typology*. Ph.D. thesis, University of British Columbia.

Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, Florence, Italy. Association for Computational Linguistics.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar*, pages 90–108. Springer.

James Rogers and Geoffrey K. Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20(3):329–342.

Chihiro Shibata and Jeffrey Heinz. 2019. Maximum likelihood estimation of factored regular deterministic stochastic languages. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 102–113, Toronto, Canada. Association for Computational Linguistics.

Chihiro Shibata and Ryo Yoshinaka. 2012. Marginalizing out transition probabilities for several subclasses of PFAs. *Journal of Machine Learning Research - Workshops and Conference Proceedings*, 21:259–263.

Sicco Verwer, Rémi Eyraud, and Colin de la Higuera. 2012. PAutomaC: A PFA/HMM learning competition. *Journal of Machine Learning Research - Workshops and Conference Proceedings*, 21.

Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. 2005a. Probabilistic finite-state machines – Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025.

Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. 2005b. Probabilistic finite-state machines – Part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1026–1039.