Learning Interactions of Local and Non-Local Phonotactic Constraints from Positive Input

Aniello De Santo
University of Utah, aniello.desanto@gmail.com

Alëna Aksënova
Google NYC, alenaks@google.com

Follow this and additional works at: https://scholarworks.umass.edu/scil

Part of the Computational Linguistics Commons, and the Phonetics and Phonology Commons

Recommended Citation
DOI: https://doi.org/10.7275/m1ab-qv64
Available at: https://scholarworks.umass.edu/scil/vol4/iss1/17

This Paper is brought to you for free and open access by ScholarWorks@UMass Amherst. It has been accepted for inclusion in Proceedings of the Society for Computation in Linguistics by an authorized editor of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
Learning Interactions of Local and Non-Local Phonotactic Constraints from Positive Input

Aniello De Santo
Dept. of Linguistics
University of Utah
aniello.desanto@utah.edu

Alêna Aksênova
Google NYC
alenaks@google.com

Abstract
This paper proposes a grammatical inference algorithm to learn input-sensitive tier-based strictly local languages across multiple tiers from positive data only, when the locality of the tier-constraints and the tier-projection function is set to 2 (MITSL$_2$; De Santo and Graf, 2019). We conduct simulations showing that the algorithm succeeds in learning MITSL$_2$ patterns over a set of artificial languages.

1 Introduction
Formal language theory has long been used to study the complexity of linguistic dependencies. Recent research has posited that the phonotactics of natural languages can be described by subclasses of the regular languages (subregular classes; McNaughton and Papert, 1971; Heinz, 2011a,b). In particular, tier-based strictly local (TSL) grammars — an extension of n-gram models — have been shown to be able to capture a variety of non-local, unbounded processes (Heinz et al., 2011; McMullin, 2016; McMullin and Hansson, 2016). Recently however, it has been suggested that the particular notion of relativized locality employed by the TSL class is unable to describe a variety of complex phonotactic patterns cross-linguistically (McMullin, 2016; Mayer and Major, 2018, a.o.). Thus, extensions to TSL have been proposed in the search of the right formal characterization for natural language phonotactics. Specifically, input-sensitive TSL languages (ITSL; De Santo and Graf, 2019) consider the local context of elements in the input string (their immediate surrounding environment), in order to simultaneously encode local and non-local requirements on the wellformedness of strings in the language.

Apart from typological coverage, an important aspect of evaluating the linguistic relevance of these analyses is to understand under which conditions such patterns are efficiently learnable. In this sense, learning approaches grounded in grammatical inferences highlight how knowledge about the formal properties of natural language phonotactics can help restrict the learning space in useful ways; and how they can inform different learning frameworks. TSL languages are efficiently learnable from positive input only (Jardine and Heinz, 2016; Jardine and McMullin, 2017). While ITSL languages have been argued to share the same property, no learning algorithm exists for this class yet. In this paper, we extend McMullin et al. (2019)’s inference algorithm for multiple tier-based strictly 2 local languages (MTSL$_2$), in order to learn patterns in the intersection closure of ITSL$_2$ — which consider 2-local contexts for segments in the input string (MITSL$_2^2$). The intersection closure of these languages is essential, if we strive to provide learning approaches able to capture the whole phonotactics of a language, and not one single pattern at the time. We evaluate our algorithm qualitatively over a variety of formal examples, and discuss known limitations of the framework and possible extensions.

2 MITSL Languages in Phonotactics
Many dependencies in phonology can be captured by strictly local (SL) grammars: local constraints that only make distinctions on the basis of contiguous substrings of segments up to some length $k$ (essentially, $k$-grams; Heinz, 2011a). For example, a $(k=2)$ local dependency requiring /s/ to surface as [z] when followed by [l] can be captured by a grammar that forbids the sequence [sl]. However, while prominent in natural language phonology, (unbounded) long-distance dependencies cannot be captured by local constraints. To account for...
this, work studying linguistic dependencies from a formal language theoretical perspective has characterized long-distance phonotactic patterns as tier-based strictly local (TSL; Heinz et al., 2011).

Tier-based strictly local languages (TSL) are able to encode a notion of relativized locality inspired by the idea of phonological tier, already popular in autosegmental phonology (Goldsmith, 1976). While a formal introduction to the properties of TSL is beyond the scope of this paper, a TSL dependency is intuitively non-local in the input string but local over a tier. A tier is defined as the projection of a subset of the segments of the input string, and the grammar constraints are characterized as the set of sequences of length $k$ not allowed on the tier. For instance, the example in Figure 1 (from Aari, an Omotic language of south Ethiopia) shows how to enforce long-distance sibilant harmony in anteriority. First we project from the string a tier $T$ that only contains sibilants, and then we ban contiguous $[s\tilde{s}]$ and $[s\tilde{\tilde{s}}]$ on $T$ (see Hayward, 1990).

![Figure 1: Example of sibilant harmony over tier from Aari.](image)

The class of TSL languages has been shown to have good cross-linguistic coverage, accounting for a variety of different phonotactic patterns cross-linguistically (Heinz et al., 2011; McMullin, 2016; Graf, 2017). Moreover, and most interesting to us, TSL$_k$ languages have been shown to be efficiently (polynomial in time and input) learnable in the limit from positive data, even when the tier-alphabet is not known a priori (Jardine and Heinz, 2016; Jardine and McMullin, 2017).

However, there are two known limits to TSL as a good formal account for natural language phonotactics. A first “issue” lies in the simplicity of TSL’s projection mechanism. Recently, several patterns have been reported that cannot be described by the way TSL’s tier-projection masks out parts of a string before enforcing some strictly local constraint (McMullin, 2016; Mayer and Major, 2018; Baek, 2017; Graf and Mayer, 2018; De Santo and Graf, 2019). These patterns include the long-distance sibilant harmony in Imdlawn Tashliyti (McMullin, 2016), the nasal harmony pattern in Yaka (Walker, 2000), the unbounded stress of Classical Arabic (see Baek, 2017, and references therein), and cases of unbounded tone plateauing. The common trait shared by these phenomena is that one has to inspect the local context (i.e., the surrounding environment) of a segment before projecting it on a tier.

Consider the case of Consonantal Nasal harmony in Yaka, in which a nasal stop induces nasalization of voiced consonants occurring at any distance to its right (Hyman, 1995; Walker, 2000). For instance, the segmental alternation shown in Ex. (1) is due to the phoneme /d/ surfacing as [n] after a preceding nasal (cf. Ex. (1a), (1b), vs. (1c)). Vowels and voiceless consonants intervening between the two harmonizing stops remain unaffected (Ex. (2)).

A TSL analysis for this pattern seems straightforward, as the data can be captured by projecting a tier of voiced consonants, and enforcing constraints banning tier adjacent [nd]. However, observe now the examples in Ex. (3): consonantal complexes composed of a nasal and a voiced oral stop neither trigger (Ex. (3a), (3b)) nor block nasality agreement (Ex. (3c)). Fig. 2 exemplifies why this interaction of a local and a non-local dependency is not TSL. Since [nd] is sometimes observed in a string-adjacent context (as in Ex. (3b)), it must be permitted as a 2-gram on a tier — even though it is only allowed when [n] and [d] are immediately adjacent in the string. But then, a TSL grammar would have no means of distinguishing Ex. (1c) from Ex. (3b).

The reader might point out that the difference between Fig. 2.a and Fig. 2.c can be resolved by extending the tier-grammar to consider 3-grams. However, in order to enforce harmony correctly, the tier-projection places every occurrence of voiced stops in the string on the tier, thus making 3-grams
constraints insufficient (e.g., Ex. (3c)). Moreover, since the number of segments between harmonizing elements is potentially unbounded, no TSL grammar can generally account for this pattern, independently of the dimension of the tier $k$-grams.

Let us consider the examples in Ex. (3) once more. Any nasal immediately followed by a voiced stop does not trigger harmony. In fact, since they do not block the harmonic process, neither the nasal nor the stop participate in the harmony at all. If we could make the projection of nasals and stops avoid those segments that appear in specific consonant clusters (e.g., [nd]) the tier constraints discussed above would work once again. This is not possible with TSL as originally defined in (Heinz et al., 2011), as TSL selects tier elements only based on their 1-local properties (i.e., which kind of segment they are). However, this kind of expressivity can be accomplished by increasing the locality window of the tier-projection mechanism.

This is accomplished by De Santo and Graf (2019)'s ITSL class: a TSL grammar is made simultaneously aware of local and non-local properties of segments in the string with a natural change to the definition of the projection function. Fig. 3 shows how, by increasing the locality of the projection to 2, we allow the grammar to project a nasal iff it is not immediately followed by a voiced oral stop, and a voiced stop iff it is not immediately preceded by a nasal. Then, we can use 2-local tier constraints to ban [nd]. This time, possible intermediate clusters are not a problem, since the projection is able to infer that they are in local contexts that make them irrelevant to the harmonic process.

ITSL languages have been shown to properly extend TSL, and fix a gap in its typological coverage. However, there is a second shortcoming to adopting TSL as a model for natural language phonotactics: TSL (and ITSL) languages are not closed under intersection (De Santo and Graf, 2019). Lack of closure under intersection is problematic as it entails that the complexity of phonological dependencies is no longer constant under factorization. This implies that the upper bound for phonological phenomena shifts, depending on whether one treats a constraint as a single phenomenon or the interaction of multiple phenomena. Moreover, we clearly want to be able to consider multiple phenomena at the same time when describing the phonotactics of a language. Consider the following additional data from Yaka.

Ex. (4) shows a vowel alternation that is independent of the nasality process, and is instead due to vowel height harmony. Vowel harmony by itself can be easily accounted for with a TSL grammar. However, this account fails if we try to model nasal harmony and vowel harmony in a single grammar — since vowels projected on the tier would interfere with the nasalization process. To account for this, De Santo and Graf (2019) propose working with the intersection closure of TSL (MTSL) and ITSL languages (MITSL). Intuitively, MTSL and MITSL can be conceptualized as encoding multiple projections (tiers) at the same time, and enforcing independent strictly local constraints over each tier. For a string to belong to the language, it needs to
be well-formed on every tier. For instance, Fig. 4 shows a grammar projecting a tier of vowels, with constraints ensuring height harmony; and a tier enforcing nasal harmony.

Since intersection closure is a desirable property from a linguistic perspective, McMullin et al. (2019) propose an algorithm that efficiently learns multiple tier-based strictly 2-local (i.e., where tier constraints are bigrams) dependencies, with no a-priori knowledge about the tier-segments or the number of tiers required. Given the typological importance of input-sensitive projection, in this paper we expand on McMullin et al. (2019) and present a grammatical inference algorithm able to learn MITSL grammars with 2-local contexts and 2-local tier constraints (k-MITSL), only from positive examples and without a priori knowledge about the number — or the content — of necessary tiers.

3 MITSL\textsuperscript{2} Inference Algorithm

The remainder of the paper discusses our learning algorithm for MITSL languages with projection contexts and tier constraints of size 2 (MITSL\textsuperscript{2}). While the previous section presented an intuitive definition of MITSL languages, a more formal definition is necessary in order to understand the way the algorithm works. Thus, we first introduce some mathematical preliminaries and discuss how the definition of MITSL grammar presented in (De Santo and Graf, 2019) grounds the intuition behind our generalization of McMullin et al. (2019)’s learning algorithm. We also discuss a generalization of the notion of 2-path as introduced by Jardine and Heinz (2016), and qualitative evaluate the performance of the learner over a variety of formal patterns.

3.1 Formal Preliminaries

We assume familiarity with set notation. Given a finite alphabet \( \Sigma \), \( \Sigma^* \) is the set of all possible finite strings of symbols drawn from \( \Sigma \). A language \( L \) is a subset of \( \Sigma^* \). For every string \( w \) and every non-empty string \( u \), \( |w| \) denotes the length of the string, and \( \varepsilon \) is the unique empty string. Left and right word boundaries are marked by \( \times, \kappa \notin \Sigma \) respectively. \( \Sigma_{\times, \kappa} \) denotes the set of strings \( w \in \Sigma^* \) that have been enriched with start and end symbols.

A string \( u \) is a \( k \)-factor of a string \( w \) iff \( \exists x, y \in \Sigma^* \) such that \( w = xuy \) and \( |w| = k \). The function \( fac_k \) maps words to the set of \( k \)-factors within them: \( fac_k(w) := \{u : u \text{ is a } k \text{-factor of } w \text{ if } |w| \geq k, \text{ else } u = w\} \). For example, \( fac_2(aab) = \{aa, ab\} \). The domain of \( fac_k \) is generalized to languages \( L \subseteq \Sigma^* \) in the usual way: \( fac_k(L) = \bigcup_{w \in L} fac_k(w) \).

We allow standard Boolean connectives (\( \land, \lor, \lnot, \to \)), and first-order quantification (\( \exists, \forall \)) over individuals. We let \( x < y \) denote precedence, \( x \approx y \) denote identity, and \( x, y \) denote variables ranging over positions in a finite string \( w \in \Sigma^* \).

As discussed, TSL languages have \( k \)-local constraints only apply to elements of a tier \( T \subseteq \Sigma \). A projection function (also called erasing function) is thus introduced to delete (or mask) all symbols that are not in \( T \). In order to extend the notion of tier in TSL languages to consider local properties of the segments in the input string, De Santo and Graf (2019) follow (Chandlee and Heinz, 2018) and define an input-sensitive projection function in terms of local contexts.

**Definition 1** (Contexts). A \( k \)-context \( c \) over alphabet \( \Sigma \) is a tri-tuple \( \langle \sigma, u, v \rangle \) such that \( \sigma \in \Sigma, u, v \in \Sigma^* \) and \( |u| + |v| \leq k \). A \( k \)-context set is a finite set of \( k \)-contexts.

**Definition 2** (ISL Projection). Let \( C \) be a \( k \)-context set over \( \Sigma \) (where \( \Sigma \) is an arbitrary alphabet also containing edge-markers). Then the input strictly \( k \)-local (ISL-k) tier projection \( \pi_{C} \) maps every \( s \in \Sigma^* \) to \( \pi_{C}(s) \), where \( \pi_{C}(\langle \sigma, u, v \rangle) \) is defined as follows, given \( \sigma \in \Sigma \cup \{\varepsilon\} \) and \( u, v \in \Sigma^* \):

\[
\begin{align*}
\varepsilon & \quad \text{if } \sigma uv = \varepsilon, \\
\sigma \pi_{C}(u \sigma, v) & \quad \text{if } \langle \sigma, u, v \rangle \in C, \\
\pi_{C}(u \sigma, v) & \quad \text{otherwise.}
\end{align*}
\]

Note that an ISL-1 tier projection only determines
projection of \( \sigma \) based on \( \sigma \) itself, showing that this projection function is really just an extension of what happens for TSL languages. The definition of ITSL languages then is as follows.

**Definition 3 (ITSL).** A language \( L \) is \( m \)-input local \( k \)-TSL (ITSL\(^m_k\)) iff there exists an \( m \)-context set \( C \) and a finite set \( R \subseteq \Sigma^k \) such that
\[
L = \{ w \in \Sigma^* : \exists a \in k \cdot \pi_C(w) \cap R = \emptyset \}.
\]

A language is input-local TSL (ITSL) iff it is ITSL\(^m_k\) for some \( k, m \geq 0 \). We call \( \langle C, R \rangle \) an ITSL grammar.

Note that the notion of tier is here expressed by the set of contexts \( C \), which is the set of tier segments with the locality conditions necessary for them to be relevant to the tier constraints. Finally, a \( n \)-MITSL language is defined as the intersection of \( n \) ISTL languages. The MITSL class has been shown to properly extend TSL, while remaining a proper subclass of star-free languages (De Santo and Graf, 2019).

In what follows we focus on learning MITSL\(^2\) languages with an arbitrary number of tiers, but with the locality of the contexts and of the tier constraints fixed to 2. The intuition behind this paper’s proposal is that, from a learning perspective, having to consider 2-local constraints (thus a segment plus its left or right context) is equivalent to treating bigrams as unitary elements of the language, and explore dependencies over them.

To do so, the algorithm incorporates the notion of a 2-path (Jardine and Heinz, 2016), generalized over bigrams. A 2-path is a 3-tuple \( \langle \rho_1, X, \rho_2 \rangle \), where \( \rho_1, \rho_2 \) are elements in \( \Sigma_{\prec, \prec} \) and \( X \) is a subset of \( \Sigma \). The 2-paths of a string \( w = \sigma_1 \sigma_2 \ldots \sigma_n \) are denoted \( \text{paths}_2(\cdot) \):
\[
\text{paths}_2 = \{ \langle \sigma_i, X, \sigma_j \rangle \mid i < j \text{ and } X = \{ \sigma_k \mid i < k < j \} \}
\]

Intuitively, a 2-path can be thought of as a precedence relation \( \rho_1 \ldots \rho_2 \) accompanied by the set \( X \) of symbols that intervene between \( \rho_1 \) and \( \rho_2 \). Formally, each 2-path is therefore a 3-path of the form \( \langle \rho_1, X, \rho_2 \rangle \). For example, the string \( \astabec\ast \) includes the following 2-paths:
\[
\langle a, \{ \emptyset \}, b \rangle, \langle a, \{ b \}, c \rangle, \langle a, \{ b, c \}, c \rangle, \langle b, \{ \emptyset \}, c \rangle, \langle b, \{ c \}, c \rangle, \langle a, \{ \emptyset \}, a \rangle, \langle a, \{ a \}, b \rangle, \langle a, \{ b, a \}, b \rangle, \langle a, \{ b, c \}, c \rangle, \langle a, \{ b, c \}, c \rangle, \langle a, \{ b, c \}, \kappa \rangle.
\]

We can extend \( \text{paths}_2(\cdot) \) from strings to languages as \( \text{paths}_2(L) = \bigcup_{w \in L} \text{paths}_2(w) \).

In order to have 2-paths capture the notion of context, we have \( \rho_1, \rho_2 \) be elements in \( \varepsilon \cdot \Delta \cdot \Sigma \) and \( X \) a subset of \( \varepsilon \cdot \Delta \cdot \Sigma \) instead of \( \Sigma \) proper. The definition of \( \text{paths}_2(\cdot) \) stays as above, considering \( \sigma_i, \sigma_j, \sigma_z \) to be 2-factors in \( w \). As this is the only notion of paths relevant for this paper, from now on we use paths, 2-paths, or \( \text{paths}_2 \) interchangeably to refer to this extended notion of paths over 2-factors. Consider once more the string \( \astabec\ast \), the set of 2-paths is now the following:
\[
\langle a, \{ ab \}, bc \rangle, \langle ab, \{ bc \}, cc \rangle, \langle \astabec, \{ bc, cc \}, \ast \rangle, \langle \astabec, \{ bc \}, c \rangle, \langle ab, \{ bc \}, c \rangle, \langle ab, \{ bc \}, c \rangle, \langle ab, \{ cc \}, c \rangle.
\]

Note that Jardine and Heinz (2016) show that the paths of a string \( w \) can be calculated in time at most quadratic in the size of \( w \). This result is unaffected, once we factor the cost of generating the set of 2-factors for \( \Sigma \).

### 3.2 The Algorithm

This paper’s algorithm takes as input a set \( I \) of strings over an alphabet \( \Sigma \), and returns an \( n \)-MITSL\(^2\) grammar \( G = \bigwedge \langle C_i, R_i \rangle \) — where each \( C_i \) is a set of contexts in bigram formats, and each \( R_i \) is a set of 2-local constraints over contexts, represented as 4-factors.

As mentioned, we adopt an approach rooted in grammatical inference, following the identification in the limit learning paradigm (Gold, 1967), with polynomial bounds on time and data (De la Higuera, 2010). Because of this, we make the fundamental assumption that the sample data in input to the learning algorithm is a characteristic sample for the targeted MITSL\(^2\) language — that is, it contains all the information necessary to distinguish a specific learning target (i.e., the phonological MITSL\(^2\) phenomenon) from any other potential targets present in the input. In other words, we assume that the input is fully descriptive of the target pattern. Recall once again that 2-factors (bigrams) are unitary symbols for the algorithm, and thus 2-local tier constraints are in fact 4-grams.

The learner exploits the fact that if a 4-gram \( \rho_1 \rho_2 \) is banned on some tier, then it will never appear in string-adjacent contexts. Thus, it establishes a canonical form for an MITSL\(^2\) grammar by associating a tier to each individual constraint. It then
Data: A finite input sample $I \subset \Sigma^* \\
Result: MITSL^2 \\
2 grammar of the form 
\[ G = \wedge\langle C_i, R_i \rangle \]
Initialize $F = \text{fac}_4(\Sigma^*)$ ... consequence,
the algorithm’s generalizations are weak with re-
spect to noisy data, since exceptions to the target
will remain on the tier. Specifically, if all of the at-
we are currently constructing. Given a tier asso-
Algorithm 1: Pseudocode for the MITSL

\[ G = G_1 \wedge G_2 \wedge \ldots \wedge G_{|F|} \]

\begin{algorithm}
\caption{Pseudocode for the MITSL}_2 In-
\end{algorithm}

explores the set of contexts relevant to each 
specific constraint, one tier at the time, starting from 
the assumption that each tier projects the full set 
of symbols in the input string. That is, we want 
to explore which symbols can act as blockers for 
a specific constraint. For each factor $\rho_1, \rho_2$ absent 
from the training data, the goal is therefore to deter-
mine which symbols can be safely removed from 
the associated tier. Recall now the notion of 2-
paths, which denote precedence relations between 
two symbols in the language, augmented with sets 
of all intervening symbols. By examining the set 
of 2-paths present in the training data, we can de-
termine which bigrams are freely distributed with 
respect to the 4-gram $\rho_1, \rho_2$ associated to the tier 
we are currently constructing. Given a tier asso-
ciated to the constraint $\rho_1, \rho_2$, only those elements 
that are not freely distributed with respect to $\rho_1, \rho_2$ 
will remain on the tier. Specifically, if all of the at-
tested $\langle \rho_1, X, \rho_2 \rangle$ 2-paths that include an interven-
ing $\sigma \in \text{fac}_2(\Sigma^*)$ are likewise attested without an 
intervening $\sigma$, the algorithm removes $\sigma$ from the 
tier, since the presence of $\rho_1 \ldots \rho_2$ is not dependent 
on that intervening bigram. As the algorithm will 
instantiate a tier for each unattested $\rho_1, \rho_2$ in the 
input sample — and with the assumption that the 
input is a characteristic sample — this elimination 
procedure guarantees that the algorithms will con-
verge to the full set of constraints and blockers for 
the target language. The crucial difference from 
McMullin et al. (2019)’s MTSL algorithm is that 
here the input sample needs to be representative of 
alternations between bigrams in the language, 
instead of elements in $\Sigma$. Note however that this 
complication is implicit within the definition of 
ITSL constraints, since they tie the distribution of 
segments in a language to their local and non-local 
contexts simultaneously. Note that, because of this 
“project everything and then remove” strategy, the 
learner trivially also infers simple local constraints 
in the input string, which are enforced on tiers 
where every element of $\Sigma$ is also an element of 
the tier (i.e., a trivial tier). This is consistent with 
the definition of MITSL, and it is actually optimal, 
since it makes the algorithm truly able to capture 
both local and long-distance dependencies in the 
phonotactic of a language. The reader is also in-
vited to observe how the extension to the notion 
of 2-paths doesn’t affect the formal guarantees in 
(Jarde and Heinz, 2016) in any significant way.

Finally, a peculiarity of our specific implementation 
is that the MITSL grammar returned is in a specific 
“canonical form”. That is, by assigning each tier 
to a single constraint, it also ties constraints that 
could co-exist on the same tier to distinct tiers. Ad-
ditionally, as a consequence of treating bigrams 
as unary symbols, when a segment is freely dis-
tributed with respect to its contexts — i.e., it gets 
projected on a tier independently on the context — 
the algorithm will still treat each bigram as a dis-
tinct element. For instance, consider $\Sigma = \{a, e, o\}$ 
and assume $e$ a tier element independently of con-
text. What our algorithm will infer is that any bi-
gram containing $e$ will be a tier element, so that 
$C = \{xe, ae, ea, oe, ee, e\}$ In practice, it is 
trivially possible to add a unification step to the tier 
and context selection, in order to have a representa-
cion closer to the grammar a human phonologist 
would write. The following section discusses the 
performance of the algorithm on an initial set of 
test data. We also discuss how the way the algo-
ithm instantiates tiers affects the class of learnable 

4 Qualitative Evaluation

Consistently with previous work on subregular 
learners, probability plays no role in the learning 
approach we outlined. As an obvious consequence, 
the algorithm’s generalizations are weak with re-
spect to noisy data, since exceptions to the target
pattern are treated equally to any other strings independently of their frequency. Thus, in what follows we set up preliminary evaluation cases that produce input samples for every language. For instance, in order to infer “project a”, the current learner only needs to consider every possible context. Thus, our measure of learnability is in terms of consistency with the grammar generating the input.

4.1 Learnable Patterns

We evaluate the MITSL learner on patterns representative of four distinct language classes, inspecting the ability of the learner to infer grammars fully representative of the input language. Using artificial patterns instead of (even simplified) natural language ones allows us to keep the target generalization transparent by relying on a small alphabet set. Note that we generate input samples from every language $L$ we test via random generation of strings from $\Sigma^*$ consistent with specified tier constraints. Thus, each sample is not guaranteed to be a characteristic sample. In this first evaluation, we set the cardinality of the input sample for each language at 1000 — which simulations show being sufficient for the learner to infer the correct pattern for every language analyzed.

An Artificial ITSL Pattern Since MITSL is a proper extension of ITSL, we first test the learner on a language with a single input-sensitive rule. Specifically, we generate an input sample for an ITSL language $L_1$ with $\Sigma = \{a, e, o, x\}$ such that $o$ immediately before $x$ prohibits $e$ to appear anywhere in the string. For instance, $eaaxae, axaexeeexx, eaoxaao \in L_1$, while $oxxe \notin L_1$. The learner should infer that the correct grammar projects $o$ on the tier banning $^*oe$ if $o$ is immediately followed by $x$, and also projects $e$ in every context. This is exactly what the algorithm learns, with the representational peculiarities discussed above. In particular, since the projection of $e$ is independent of context, instead of a single tier the learner instantiates a tier for every 4-gram $^*oe\sigma$ and $^*oxe\sigma$, where $\sigma$ is every element of $\Sigma = \{a, e, o, x\}$. For each such tiers, the algorithm infers that $ox$ is always projected, as is the corresponding bigram containing $e$.

First-Last Harmony We then test a second ITSL often discussed in the formal language literature on the complexity of phonotactics: an harmonic dependency between the first and the last element in the string. For instance, assuming that the first and last symbols in a string are vowels, the requirement might be that they agree in height. Specifically, we consider a language $L_{FL}$ with $\Sigma = \{a, o, x\}$. Then, we generate data such that $^* \times aox, ^* \times oax$ should be banned tiers. For instance, $aaaxaoaa, aaxaxaxxoxao \notin L_{FL}$ but $oxaaaxaa, aaxaaxxax \notin L_{FL}$. This pattern is worth testing in addition to the one we used above, as it requires both elements in the constraint to be sensitive to their local context (the end and start symbols, respectively). As expected, the learner succeeds in generalizing from the input sample to the full set of required constraints, instantiating a distinct tier for each constraint combination.

An Artificial MITSL Pattern As the algorithm is fully successful on ITSL patterns, we then test it on the other relevant subclass of MITSL: an MITSL language with constraints to be enforced on multiple tiers, but with a TSL projection for each tier which does not depend on context. Specifically, we generate an artificial language $L_2$ which replicates the idea of a consonant harmony and a vowel harmony process occurring within the same language. We consider $L_2$ with $\Sigma = \{a, o, p, b\}$, $T_1 = \{a, o\}, T_2 = \{p, b\}$ and enforce constraints such that $R_1 = \{^*aa, ^*oa\}, R_2 = \{^*bp, ^*pb\}$. For instance, $appp, apbpp, popopp, apoppopp \in L_2$ but $apabo, apoppopp \notin L_2$. As expected, the learner successfully infers the full grammar. As the algorithm still considers the pattern an MITSL one, it actually needs to consider every possible context for each symbol, and it thus requires a more extensive input set than McMullin et al. (2019)’s. For example, in order to infer “project a”, the current

---

1 A Python implementation of the learning algorithm and testing tools is available here.
algorithm actually learns to project \{ao, oa, ap, pa, ab, ba\}. Importantly, this is exactly what is expected when learning an input-sensitive projection.

An Artificial MITSL$_2^2$ Pattern  Finally, we test a proper MITSL$_2^2$ pattern, which combines instances of the dependencies observed above. In particular, we generate a language that respects two independent ITSL$_2^2$ constraints. A first constraint is the one we exploited in our artificial ITSL$_2^2$ example, such that \( o \) immediately before \( x \) prohibits \( e \) to appear anywhere in the string. The second constraint enforces that if \( b \) immediately precedes \( y \), then any following instance of \( d \) is forbidden. This replicates a simplified pattern of dissimilation. Thus, the whole alphabet for the language is \( \Sigma = \{a, e, o, x, b, p, d, y\} \) with \( \{o, e, x\} \) being relevant only to the first constraint, and \( \{b, y, d\} \) only to the second. This corresponds to a language \( L_3 \) such that \( badox, edyo, byoxpa \in L_3 \) but \( bydax, ozdye, byoxde \notin L_3 \). As expected given the previous results, the learner is fully successful on this language, correctly generalizing over the distinct ITSL constraints independently.

4.2 Unlearnable Patterns

Our simulations show that our MITSL$_2^2$ learner succeeds on a variety of complex patterns, given a positive sample representative of the target language. However, since our approach relies directly on Mc-Mullin et al. (2019)’s idea for instantiating multiple tiers, it suffers from the same limitations. That is, there is a portion of logically-possible M(I)TSL patterns which cannot be captured using the current learning strategy. To understand why, note again that the algorithm instantiates a tier for each potential constraint. From this, it follows immediately that each restriction can only be enforced on at most one tier. Thus, the learner fails on those patterns that have overlapping tier alphabets — i.e., tiers that share some symbols, but that are not overall in a set/subset relation. Specifically, the algorithm is not able to consider overlapping tiers that are associated with a single \(*_{p_1p_2}\) restriction (e.g., if such constraint needs to be independently blocked by a different symbol on each tier). It is important to note that languages exhibiting patterns with these problematic dependencies appear to be unattested, and for reasons that have been speculated to be directly related to ease of learnability, as excluding overlapping tiers exponentially reduces a learner’s hypothesis space (Aksēnova and Deshmukh, 2018).

5 Discussion

This paper proposes a grammatical inference algorithm to learn multiple input-sensitive tier-based strictly local languages (MITSL; De Santo and Graf, 2019) from positive data only, once the locality of the tier-constraints and of the tier-projection function is set to two (MITSL$_2^2$). The algorithm makes use of the characteristics of the language class in order to guide its exploration of the learning space. We then discussed simulations demonstrating the learner’s success over four artificial languages belonging to subclasses of MITSL$_2^2$ languages. Thus, this paper contributes to the growing array of practical tools for the efficient learnability of classes in the subregular hierarchy.

One usual critique of algorithms of this kind is the unrealistic assumptions made about the nature of the input sample. Importantly, while it is true that these algorithms are only guaranteed to converge to the correct grammar if the sample is a characteristic, that doesn’t mean that they fail in other cases. The condition on the input “simply” gives us a converging guarantee. In this sense, our qualitative evaluation is meant as a first step toward a more extensive study of the general learning performance of these approaches, when varying the coverage of the input sample — in the spirit of what recently suggested in (Aksēnova, 2020).

Similarly, an obvious issue when applying and testing this algorithm to natural language data will come from its inability to deal with exceptions and noisy input in general. Obviously, exceptions in the data could be handled by developing probabilistic versions of the learner. Note that, while formal languages were discussed here as categorical in nature, they admit stochastic counterparts. A stochastic version of an MITSL algorithm could for instance take into consideration the frequency of a specific path before removing a symbol from the tier. It is also reasonable to conceive of versions of this algorithm taking natural phonotactic classes into account in order to discriminate between potential hypotheses about the a tier in presence of contradicting data. Crucially though, the results in this paper show how inferring tiers and input-sensitive tier constraints can be achieved without \textit{a priori} information about them. The strategy formulated here could then be combined with learning approaches that take advantage of different aspects of the nature of the input data (Gouskova and Gallagher, 2020; Rasin
et al., 2019).

Finally, the transparent way our algorithm explores the learning space suggests that it could be used as a baseline to evaluate the needs of more opaque learning strategies with respect to different kinds of input, in line with previous work combining grammatical inference techniques to blackbox learning models (Avcu et al., 2017; Mahalunkar and Kelleher, 2018, a.o.).

References


E Mark Gold. 1967. Language identification in the limit. Information and control, 10(5).


