0.1 Strictly k-Local Stringsets

Here we present an algorithm and prove that it identifies the Strictly k-Local (SL_k) stringsets in the limit from positive data. The first proof of this result was presented by Garcia *et al.* (1990), though the Markovian principles underlying this result were understood in a statistical context much earlier. The learning scheme discussed there exemplifies more general ideas (Heinz, 2010; Heinz *et al.*, 2012).

The notion of *substring* is integral to SL stringsets. Formally, a string u is substring of string v ($u \trianglelefteq v$) provided there are strings $x, y \in \Sigma^*$ and $v = xuy$. Another term for substring is *factor*. So we also say that u is a factor of v. If u is of length k then we say u is a k-factor of v.

A stringset S is Strictly k-Local if and only if there is a number k such that for all strings $u_1, v_1, u_2, v_2, x \in \Sigma^*$ such that if $|x| = k$ and $u_1xv_1, u_2xv_2 \in S$ then $u_1xv_2 \in S$. We say S is closed under suffix substitution (Rogers and Pullum, 2011).

A theorem shows that every SL_k stringset S has a basis in a finite set of strings (Rogers and Pullum, 2011). These strings can be understood as *forbidden* substrings. Informally, this means any string s containing any one of the forbidden substrings is not in S. Conversely, any string s which <u>does not</u> contain <u>any</u> forbidden substring belongs to S.

The same theorem shows that a SL stringset S can be defined in terms of a finite set of *permissible* substrings. In this case, s belongs to S if and only if <u>every</u> one of its k-factors is permissible.

We formalize the above notions by first defining a function the $\texttt{factor}_k$, which extracts the substrings of length k present in a string, or those present in a set of strings. If a string s is of length less than k then $\texttt{factor}_k$ just returns s.

Formally, let $\texttt{factor}_k(s)$ equal $\{u \mid u \trianglelefteq s, |u| = k\}$ whenever $k \leq |s|$ and let $\texttt{factor}_k(s) = \{s\}$ whenever $|s| < k$. We expand the domain of this function to include sets of strings as follows: $\texttt{factor}_k(S) = \bigcup_{s\in S} \texttt{factor}_k(s)$.

To formally define SL_k grammars, we introduce the symbols $\rtimes$ and $\ltimes$, which denote left and right word boundaries, respectively. These symbols are introduced because we also want to be able to forbid specific strings at the beginning and ends of words, and traditionally strictly local stringsets were defined to make such distinctions (McNaughton and Papert, 1971). Then let a grammar G be a finite subset of $\texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\})$.

The "language of the grammar" $L(G)$ is defined as the stringset $\{s \mid \texttt{factor}_k(\rtimes s \ltimes) \subseteq G\}$. We are going to be interested in the collection of stringsets SL_k, defined as those stringsets generated from grammars G with a longest string k. Formally,

$$\mathrm{SL}_k \overset{\mathrm{def}}{=} \{S \mid G \subseteq \texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\}), L(G) = S\} \ .$$

This is the collection C of learning targets.

For all $S \in \mathrm{SL}_k$, for any presentation ϕ and time t, define k-SPIA (Strictly k-Local Inference Algorithm) as follows

$$k\text{-SLIA}\big(\varphi\langle t\rangle\big) = \begin{cases} \varnothing & \text{if } t = 0 \\ k\text{-SLIA}(\varphi\langle t-1\rangle) \cup \texttt{factor}_k(\rtimes\varphi(t)\ltimes) & \text{otherwise} \end{cases}$$

Exercise 1. Prove algorithm k-SLIA identifies in the limit from positive data the collection of stringsets SL_k.

Note that we are being a little sloppy here. Technically, the output of k-SLIA given some input sequence is a set of subsequences G, not a program. What we really mean with the above is that $k-$SLIA outputs a program which uses G to solve the membership problem for $L(G) = \{w \mid \mathtt{subseq}_k(w) \subseteq G\}$. This program looks something like this.

1. Input: any word w.
2. Check whether $\mathtt{factor}_k(\rtimes w \ltimes) \subseteq G$.
3. If so, OUTPUT Yes, otherwise OUTPUT No.

All $k-$SLIA does is update this program simply by updating the contents of G.

Theorem 1. *For each k, $k-$SLIA identifies in the limit from positive data the collection of stringsets SL_k.*

Proof Consider any $k \in \mathbb{N}$. Consider any $S \in \mathrm{SL}_k$. Consider any positive presentation φ for S. It is sufficient to show there exists a point in time t_ℓ such that for all $m \geq t_\ell$ the following holds:

1. k-SLIA$(\langle m \rangle) = k-$SLIA$(\langle t_\ell \rangle)$ (convergence), and
2. k-SLIA$(\langle m \rangle)$ is a program that solves the membership probem for S.

Since $S \in \mathrm{SL}_k$, there is a finite set $G \subseteq \Sigma^{\leq k}$ such that $S = L(G)$.

Consider any factor $g \in G$. Since $g \in G$ there is some word $w \in S$ which contains g as a k-factor. Since G is finite, there are finitely many such w, one for each g in G. Because φ is a positive presentation for S, there is a time t where each of these w occurs. For each w let t be the first occurence of w in φ. Let t_ℓ denote the latest time point of all of these time points t. Next we argue that for all time points m larger than this t_ℓ, the output of $k-$SLIA correctly solves the membership problem for S and does not change.

Consider any $m \geq t_\ell$. The claim is that k-SLIA$(\langle m \rangle) = k-$SLIA$(\langle t_\ell \rangle) = G$. For each g in G, a word containing g as a factor occurs at or earlier than t_ℓ and so $g \in k-$SLIA$(\langle m \rangle)$. Since g was arbitrary in G, $G \subseteq k-$SLIA$(\langle m \rangle)$.

Similarly, for each $g \in k-$SLIA$(\langle m \rangle)$, there was some word w in φ such that w contains g as a factor. Since φ is a positive presentation for S, w is in S. Since w belongs to S, $\mathtt{factor}_k(w) \subseteq G$ and so g belongs to G. Since g was arbitrary in k-SLIA$(\langle m \rangle)$ it follows that k-SLIA$(\langle m \rangle) \subseteq G$.

It follows k-SLIA$(\langle m \rangle) = G$.

Since m was arbitrarily larger than t_ℓ we have both convergence and correctness.

Since φ was arbitrary for S, S arbitrary in SL_k and k arbitrary, the proof is concluded. $\square$

References

Garcia, Pedro, Enrique Vidal, and José Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*, 325–338.

Heinz, Jeffrey. 2010. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 897–906. Uppsala, Sweden: Association for Computational Linguistics.

Heinz, Jeffrey, Anna Kasprzik, and Timo Kötzing. 2012. Learning with lattice-structured hypothesis spaces. *Theoretical Computer Science* 457:111–127.

McNaughton, Robert, and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

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