3 Strictly Local stringsets

For each k, and each there is a canonical form for strictly k-local stringsets. Here it is.

Definition 4. Given the stucture of the *canonical SL_k DFA* for $L(G)$ is a DFA such that

- $Q = \Sigma^{\leq k-1}$;
- Σ is the alphabet;
- $q_0 = \lambda$;
- $F = Q$; and
- For all $q \in Q, \sigma \in \Sigma$, $\delta(q, \sigma) = \texttt{suff}_{k-1}(qa)$

Here is an example, where $\Sigma = \{a, b, c\}$ and $k = 2$. Observe that the language of this DFA is Σ^*.

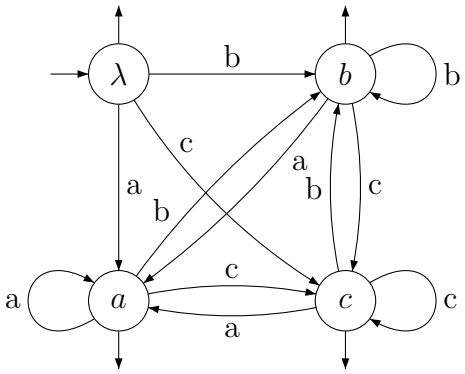

Figure 1: A canonical SL_2 DFA

Other SL_2 languages are obtained by removing transitions or making states non-final. In other words, every SL_2 stringset corresponds to some subgraph of this canonical DFA.

To see why, consider any strictly k-local grammar $G \subseteq \texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\})$.

- For all $w \in \Sigma^*$: $w \in F$ iff $w\ltimes \in G$ or $\rtimes w\ltimes \in G$.
- For all $wa \in \Sigma^*$: $\delta(w, a)$ exists iff $wa \in G$ or $\rtimes wa \in G$.

3.1 Strictly k-Local stringsets

We have already seen an algorithm that learns Strictly k-local stringsets. It operates by recording the k-factors of the observed words appended with word boundaries. These k-factors correspond to transitions in the canonical machine. So as each word is observed, we essenentially carve a path in the canonical SL_k DFA.

Figure 2 illustrates such learning for positive presentations beginning with $\langle a, aaab, bc\rangle$.

It is worthwhile to compare the development of the DFA above compared to string extension learning grammar.

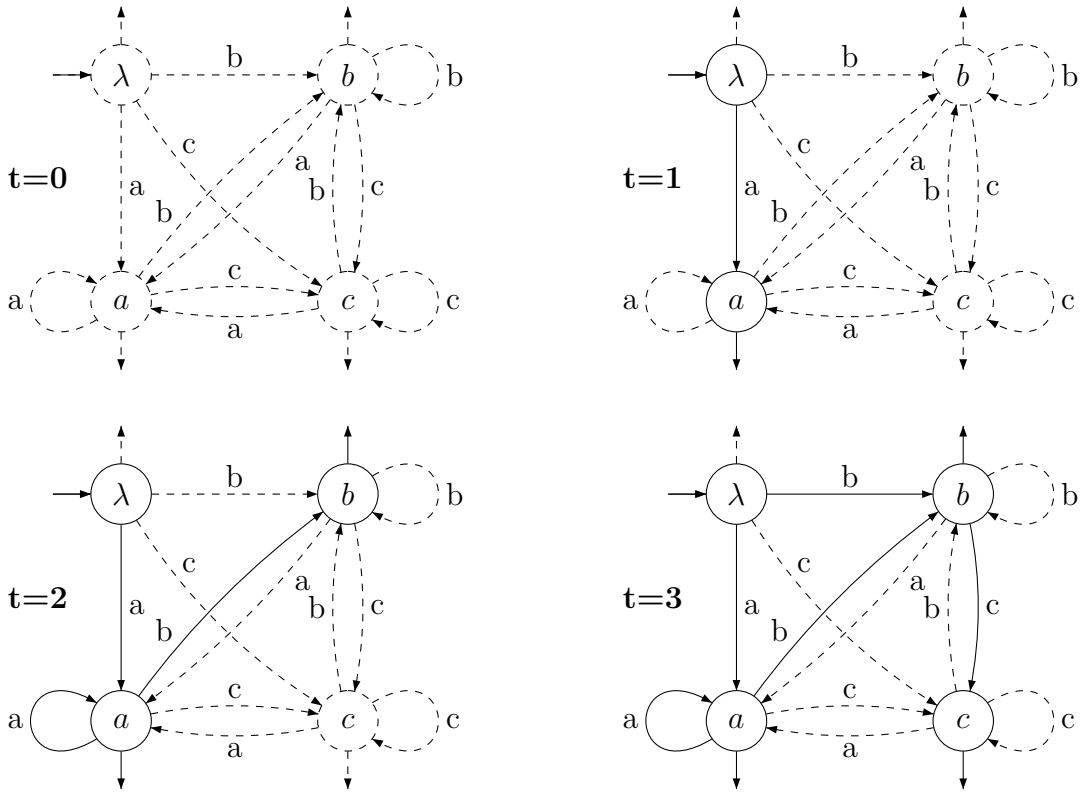

Figure 2: Learning as carving paths in the pre-existing structure. The DFA at time $t = 0$ shows is the structure and subsequent DFA show the paths carved at each point in time with $\varphi(1) = a$, $\varphi(2) = aaab$, $\varphi(2) = bc$.

time t	$\varphi(t)$	$\texttt{factor}_k(\rtimes\varphi(t)\ltimes)$	G_t
0			$\emptyset$
1	a	{⋊a, a⋉}	{⋊a, a⋉}
2	aaab	{⋊a, aa, ab, b⋉}	{⋊a, a⋉, aa, ab, b⋉}
3	bc	{⋊b, bc, c⋉}	{⋊a, a⋉, aa, ab, b⋉, ⋊b, bc, c⋉}

For each time t, the stringset generated by the grammar and the DFA are exactly the same!

3.2 Stochastic Strictly k-Local stringsets

"Carving paths" in a fixed determinstic structure can be generalized to learn stochastic stringsets. A stochastic stringset is a probability distribution over Σ^*. So each string has some probability between 0 and 1 (inclusive) and the sum of all the probabilities of all the strings equals 1.

$$\sum_{w \in \Sigma^*} P(w) = 1 \tag{2}$$

Each DFA describes a class of stochastic stringsets as follows. Each transition $\delta(q, a)$ is associated with a probability θ_{qa}. Each state is also associated with a probability $\theta_{q\ltimes}$. The probability on the state is the probability of stopping/accepting/finality. The probabilities associated with each transitiona and each state are the *parameters* of the model the DFA describes. Provided for each state q, the following holds then the DFA describes a probability distribution over Σ^*.

$$\sum_{a \in \Sigma} \theta_{qa} + \theta_{q\ltimes} = 1 \tag{3}$$

It can be said that the DFA is a *parametric* model with $|Q| + |Q| \cdot |\Sigma|$ parameters. For a given DFA whose parameters are fixed, these parameters are often written simply as θ even if there are many of them. You can think of θ as a vector of values. The possible values these parameters can take on while still ensuring a probability distribution over Σ^* is denoted Θ.

So what does it mean to learn a stochastic stringset? Given a parametric model, it basically means finding the true values of the parameters θ from the set of all possible parameter settings Θ. This may not reasonable when the data sample is small. So here is one definition of learning stochastic stringsets that has been proposed that avoids the issue of data sufficiency.

Definition 5 (Maximum Likelihood Estimate (MLE))**.**

> Algorithm A *yields the maximum likelihood estimate* for a class of stochastic stringsets C provided
>
> for all stochastics stringsets $S \in C$,
>
> for all sequences D of independent and identically distributed strings drawn from S
>
> - the parameters θ output by A assign a probability to D which is greater than the probability other parameter choices $\theta' \in \Theta$ assign to D.

In other words, if A outputs parameters θ than any deviation from θ will lower the probability of the data. In this way, the learning algorithm does not try to find the true parameter values, it simply identifies those values that maximize the likelihood (probability) of the data under the assumed parametric model.

Importantly, the MLE is also what is known as a *consistent* estimator. With enough data, it *will* converge to the true values. Calculus expresses convergence over real values with the idea of aribtirary precision. So for any small number ϵ you think of, there will be some large sample of data such that the MLE will return parameters θ that assign a probability to words that are within ϵ of the true values.

Definition 6 (Consistent Estimator)**.**

Algorithm A is a *consistent estimator* for a class of stochastic stringsets C provided

for all stochastics stringsets $S \in C$,

for all $\epsilon > 0$

there is some number $n \in \mathbb{N}$ such that

for all sequences D of independent and identically distributed strings drawn from S with $|D| > n$,

- the parameters θ output by A with D are within ϵ of the true parameter values.

OK, that's a measuring stick by which we can judge our learning algorithms.

If the parametric is a DFA, as we have considered above, then the algorithm that

1. pushes the data through the DFA,
2. counts the times each transition is traversed, and then
3. then normalizes the counts to obtain parameter values

yields the maximum likelihood estimate. This result is a proven theorem (Vidal *et al.*, 2005a,b).

Figure 3 shows how the counting would progress and Table 1 how the parameter values are updated over time.

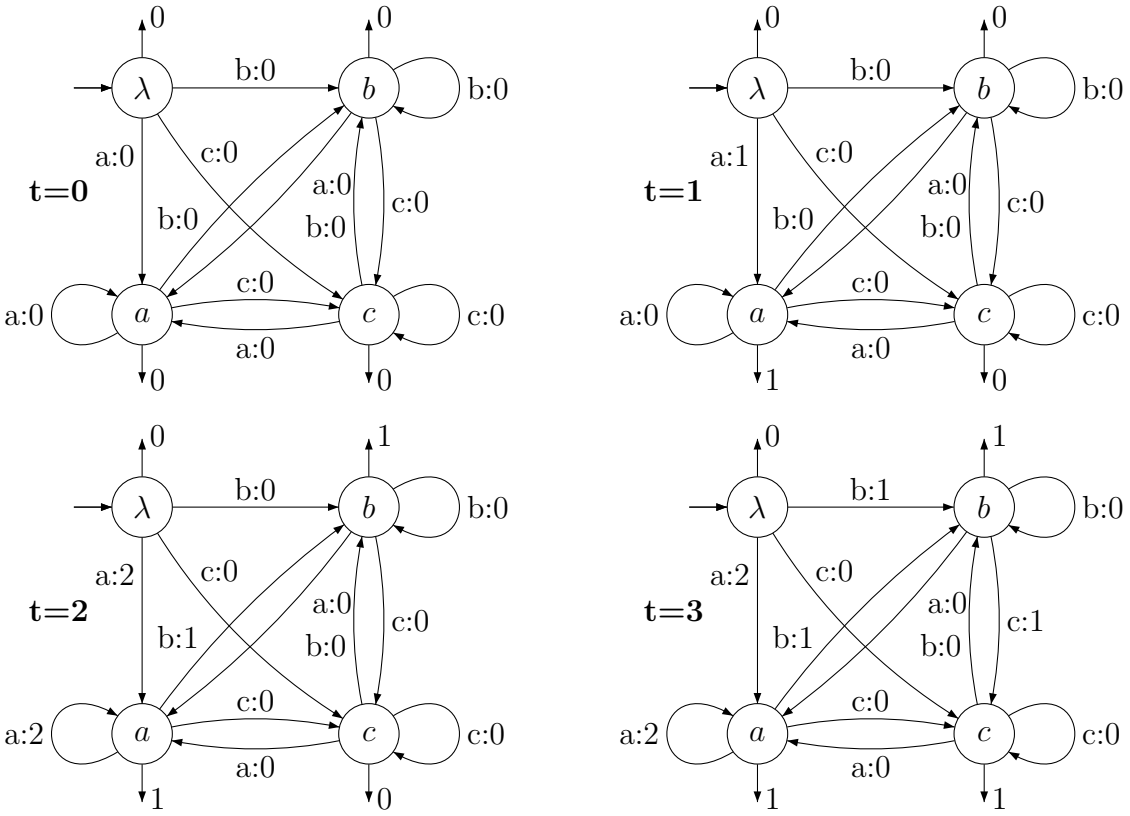

Figure 3: Learning as *counting* paths in the pre-existing structure. The DFA at time $t = 0$ shows is the structure and subsequent DFA show the paths carved at each point in time with $\varphi(1) = a$, $\varphi(2) = aaab$, $\varphi(2) = bc$.

	0	1	2	3		0	1	2	3
$\theta_{\lambda\ltimes}$	0	0	0	0	$\theta_{b\ltimes}$	0	0	1	0.5
$\theta_{\lambda a}$	0	1	1	0.67	θ_{ba}	0	0	0	0
$\theta_{\lambda b}$	0	0	0	0.33	θ_{bb}	0	0	0	0
$\theta_{\lambda c}$	0	0	0	0	θ_{bc}	0	0	0	0.5
$\theta_{a\ltimes}$	0	1	0.25	0.25	$\theta_{c\ltimes}$	0	0	0	1
θ_{aa}	0	0	0.5	0.5	θ_{ca}	0	0	0	0
θ_{ab}	0	0	0.25	0.25	θ_{cb}	0	0	0	0
θ_{ac}	0	0	0	0	θ_{cc}	0	0	0	0

Table 1: Parameter values for the SL_2 parametric model with the sample

References

Chandlee, Jane. 2014. Strictly local phonological processes. Doctoral dissertation, The University of Delaware.

Chandlee, Jane, Rémi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics* 2:491–503.

Chandlee, Jane, Rémi Eyraud, and Jeffrey Heinz. 2015. Output strictly local functions. In *Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015)*, edited by Marco Kuhlmann, Makoto Kanazawa, and Gregory M. Kobele, 112–125. Chicago, USA.

Chandlee, Jane, and Jeffrey Heinz. Forthcoming. Strictly local phonological processes. *Linguistic Inquiry* .

Chandlee, Jane, Jeffrey Heinz, and Adam Jardine. To appear. Input strictly local opaque maps. *Phonology* .

Vidal, Enrique, 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* 27:1013–1025.

Vidal, Enrique, Frank 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* 27:1026–1039.