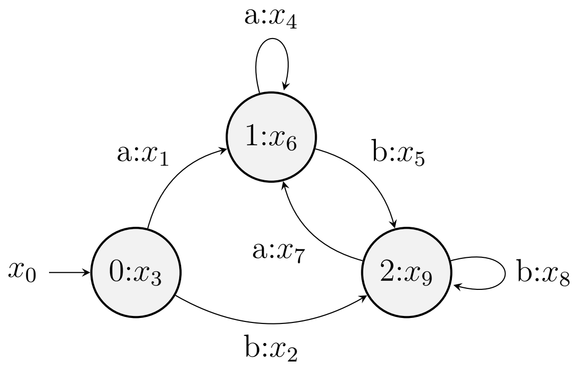
Learning String Transductions by Solving Equations

(joint work with Rémi Eyraud and Dakotah Lambert)

The basic problem. You are given a deterministic finite-state transducer (DFT), whose states and transitions are known to you, except for the outputs on the transitions, which are hidden from you. You are also given a finitely many pairs of strings: an input string and the output string that the transducer produces with this input. With this information, what can you deduce about the outputs that are hidden from you? What information would you need to correctly identify the hidden outputs?

Example. You know that there is a lawful relationship between the input/output pairs of strings below that can be expressed by the DFT in diagram below.

(ba, cbcac) (abba, aabbcbbcac)
(baba, cbcbbbcac) (aba, aabbbcac)



Motivation 1. Many phonological processes belong to classes of string-to-string functions that can be described with a single deterministic transducer. For example, there is a single transducer for k-ISL functions [1]. Learning these processes from (UR,SR) pairs can then be construed as a form of this problem.

Motivation 2. Algorithms for learning DFTs currently do not necessarily make use of all the information in a sample to identify the hidden output transitions. In some cases (SOSFIA [2]), the sufficient sample is very particular, and the algorithm ignores data points not in this sample. In other cases (ISLFLA [3]), OSTIA [4]), a DFT is returned for any sample, but it is unknown how to measure the "goodness" of this DFT compared to the target one in the absence of a sufficient sample.

Proposed Solution. Solve a system of equations provided by the data. Each input string corresponds to a single path, which is a sequence of transitions. For example, the four data points above yield the following equational system.

$$\begin{aligned}
x_0x_1x_5x_9 &= cbcac && (1)\\
x_0x_2x_7x_5x_7x_6 &= cbcbbbcac && (2)\\
x_0x_1x_5x_8x_7x_6 &= aabbcbbcac && (3)\\
x_0x_1x_5x_7x_6 &= aabbbcac && (4)
\end{aligned}$$

How to solve. We don't have anything clever to say here. A brute force calculation of all the possibilities is possible because of the finiteness of the system. One advantage of this approach is that it can be applied incrementally, and at every moment, one can say exactly what is known and what is unknown about any particular transition.

Complexity. A brute force consideration of all the possibilities grows very fast! I won't provide the exact calculation here, but it suffices to notice that for a single equation with the number of solutions grows like "n choose k," which is equal to $\frac{n!}{k!(n-k)!}$. Furthermore, if there are two equations for *non-overlapping* input paths, then the number of solutions is *product* of the numbers of solutions of each equation!

On the other hand, if input paths overlap, then the correct values must be common to both paths, which restricts the viable solutions. This fact can be leveraged to process inputs in a way that attempts to order the inputs in a way to take advantage of overlapping paths in order to keep the number of hypotheses entertained at any given point small. We can also limit the possible solutions by only considering ones that are *onward*, which ensures that there is a unique way to assign outputs to the DFT for any target function.

Conjecture. Dakotah conjectured that the equational method will always yield the correct solution provided the sample includes, for each pair of successive transitions (x, y) in the given DFT, an (input, output) data point yielding an equation including xy on its left hand side.

If it were true. The sample Dakotah conjectures is much more flexible than the sufficient samples of SOSFIA and ISFLA in the sense that many different data sets could count as sufficient. It would also be the case that the same set of input strings would be sufficient for learning every function in the class! There are also some excellent learning results we can also obtain if it assumed that the DFT generates the data stochastically.

Status. Anton gave a counterexample! It's not true! So we are looking for a new conjecture. Anton's counterexample helps here.

References

[1] Jane Chandlee and Jeffrey Heinz. Strict locality and phonological maps. *Linguistic Inquiry*, 49(1):23–60, Jan 2018.

[2] Adam Jardine, Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. Very efficient learning of structured classes of subsequential functions from positive data. In Alexander Clark, Makoto Kanazawa, and Ryo Yoshinaka, editors, *Proceedings of the Twelfth International Conference on Grammatical Inference (ICGI 2014)*, volume 34, pages 94–108. JMLR: Workshop and Conference Proceedings, September 2014.

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

[4] José Oncina, Pedro García, and Enrique Vidal. Learning subsequential transducers for pattern recognition tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:448–458, May 1993.