Empirical and Theoretical Arguments for Using Properties of Letters for the Learning of Sequential Functions

Magdalena Markowska MAGDALENA.MARKOWSKA@STONYBROOK.EDU
Jeffrey Heinz JEFFREY.HEINZ@STONYBROOK.EDU
Department of Linguistics
Institute for Advanced Computational Science
Stony Brook, NY, USA

Editors: François Coste, Faissal Ouardi and Guillaume Rabusseau

Keywords: sequential functions, finite state transducers, phonological features

1. Introduction

Sequential functions are those that can be represented with a deterministic finite state transducer (DFT). Jardine et al. (2014) show that classes of sequential functions can be learned given a class-defining structure of such a DFT and a finite characteristic sample in linear time and data. In this work we show that learning sequential functions can be achieved with much less data if the representation of the data is changed from letters to smaller units — the properties of those letters. This begins to address a question raised by de la Higuera (2006) on the grammatical inference with structured alphabets.

Sequential functions are of particular interest in phonology, which is a study of sound patterns and sound changes in natural languages. Heinz and Lai (2013); Chandlee (2014, 2017); Chandlee and Heinz (2018) show that sequential functions can successfully model the majority of phonological processes, describing various types of sound alternations. The equivalent of letters in phonology are phonemes, which are the contrastive sound units within a language. Every phoneme can be described in terms of their articulatory or acoustic properties, called features. A few examples of such features are [nasal], which distinguishes oral from nasal sounds, [consonantal], that differentiates vowels from consonants, or [coronal] that identifies sounds produced with the tongue blade raised (Hayes, 2009). Each of those features typically have two values: + and −. Features are important for modelling phonological processes, as they group together different phonemes into natural classes. That explains why certain sounds are affected by some processes, while others are not. For example, in Polish only the following sounds undergo palatalization: /s,z,t,d,n,l,r/ and they all share the [coronal] feature (Rubach, 1984).

2. Preliminaries

A DFT is a seven-tuple $T = (Q, \Sigma, \Delta, q_0, v_0, \delta, F)$, where Q is a finite set of states; Σ is the input alphabet; Δ is the output alphabet; $q_0 \in Q$ is the *initial* state; $v_0 \in \Delta^*$ is the *initial string*; δ is a *transition function* with domain $Q \times \Sigma$ and co-domain $Q \times \Delta^*$; F is a *final* function with domain Q and co-domain Δ^*. A transition $(q, a, r, v) \in \delta$ means there is a transition from state q to state r reading letter a and writing string v.

One class of sequential functions are input strictly local (ISL) functions defined by Chandlee (2014); Chandlee et al. (2014). For all ISL functions f, there is a k such that there is a DFT representing f which is structured in a particular way: the current state of the machine is always determined by the previous $k-1$ symbols read in the input. Such DFTs are called k-ISL DFTs.

As an example let us consider regressive nasalization (RN), a 2-ISL function. In English, vowels can be nasalized when followed by a nasal consonants. For instance, the *a* [æ] in *man* [mæ̃n] will be most likely produced with air flowing through the nasal cavity, similarly to the consonant *n* that follows it. The *a* [æ] in *mat* [mæt], on the other hand, is produced as a fully oral vowel. To represent this process, a window of length $k = 2$ is required. This is because in order to know whether the input *a* will be an oral or a nasal vowel in the output, we need to consider the sound that follows it.

Phonological features are useful to model such sound changes. During the process of regressive nasalization it is only one feature that changes in a phoneme, as opposed to the entire phoneme. For example, the only difference between [æ] and [æ̃] is that the latter one is [+nasal], while the former is [-nasal]. With features as a tool we can target a specific property of a sound that undergoes a change, and a trigger that causes the change. In the case of RN, the targets are vowels and the triggers are nasal consonants. We can express this generalization as follows "RN: [-consonantal]phonemes become [+nasal] when followed by [+consonantal, +nasal] phonemes." If phonemes were used instead of features, there would have to be rules for every combination of vowel and nasal such as "æ becomes æ̃ followed by n", "i becomes ĩ followed by n" (cf. [mit] 'meet' vs. [mĩn] 'mean'), "i becomes ĩ followed by m (cf. [sip] 'seep' vs. [sĩm] 'seem')," and so on.

The Structured Onward Sequential Function Inference Algorithm (SOSFIA) was introduced by Jardine et al. (2014). It takes as its arguments a finite sample of input-output pairs $S \subset \Sigma^* \times \Delta^*$, and an output-empty DFT, i.e. a transducer where $v_0 = \square$ for all q, $F(q) = \square$, and for every $(q, a, v, r) \in \delta$, $v = \square$, where $\square$ represents a 'blank' that needs to be filled in. If the characteristic sample S sufficiently represents the process, SOSFIA fills in every $\square$ by calculating the minimal change in the output strings for each transition to be consistent with the sample, and returns a complete DFT. If the training sample S is incompatible with the given DFT, SOSFIA can return an error with that information. SOSFIA identifies classes of sequential functions in time linear in the size of the sample, and the size of the characteristic sample is linear in the size of the given DFT.

3. SOSFIA with features: strategy and theoretical benefits

When a k-ISL DFT is constructed over letters of the alphabet, there are $|\Sigma|^{\leq k-1}$ states, one for each sequence up to length $k-1$. Consequently, as $|\Sigma|$ and k grow, the size of the characteristic sample grows as well. Our goal is to reduce the size of the DFT using properties of the letters of Σ. When the letters of Σ can be characterized with a set of properties, these can be used to effectively reduce the size of the alphabet. For example, n binary features, like the ones mentioned, can represent up to 2^n letters.

Formally, a binary feature ϕ is a homomorphism from Σ to the set $\{+, -\}$. For example, $\phi_{nasal}(mat) = +--$. If Φ is an ordered set of binary features, then Φ is a pointwise product of its homomorphisms. For instance, $\Phi_{[nasal,consonantal]}(mat) = [++][--][-+]$. Given Φ_X, we observe that it effectively maps Σ to an alphabet of size $2^{|X|}$, which we refer to as the *size of the featural alphabet*, denoted $|\Phi_X|$. Finally, given a sample S of input/output pairs, let $\langle \Phi_X, \Phi_Y \rangle(S) = \{(\Phi_X(x), \Phi_Y(y)) : (x, y) \in S\}$.

We aim to decompose the target sequential function $f : \Sigma^* \to \Sigma^*$ into n-many sequential functions, one for each binary feature, each represented with a k-ISL DFT and apply SOSFIA to each of these n DFTs, where the data sample has been projected appropriately. Each DFT can be thought of as aiming to predict a particular feature x of the output. The output alphabet for each DFT is therefore $\{+, -\}$ which will be interpreted as the value of x. It follows then given a sample of input/output string pairs, the output strings will be projected according to ϕ_x.

For each DFT, the input alphabet depends on which homomorphisms Φ is used. Here we propose a simple search strategy. For the DFT predicting feature x, denoted DFT$_x$, project the input strings in the sample with homomorphism $\Phi_{[x]}$. SOSFIA can be run to see whether the sample is compatible or not with this structure. If it is, learning has been successful. If it is not, we choose another feature y, and project the input strings in the sample with homomorphism $\Phi_{[x,y]}$ and try again. We observe that adding y to also affects the size of the DFT for x. This process repeats until learning is successful. In the worst case, the requires an exponential search, but we believe in practice only a few features are necessary to generalize successfully.

To illustrate these ideas, consider regressive nasalization again and the features nasal, consonantal, and coronal. We consider three 2-ISL DFTs, one for each: DFT$_{nasal}$, DFT$_{consonantal}$, DFT$_{coronal}$ and assume some sample of input/output pairs S. In the first stage, SOSFIA runs three times: once with input DFT$_{nasal}$ and sample $\langle \Phi_{[nasal]}, \Phi_{[nasal]} \rangle(S)$, once with input DFT$_{consonantal}$ and sample $\langle \Phi_{[consonantal]}, \Phi_{[consonantal]} \rangle(S)$, and once with input DFT$_{coronal}$ and sample $\langle \Phi_{[coronal]}, \Phi_{[coronal]} \rangle(S)$. For regressive nasalization, it is expected that learning will succeed for DFT$_{consonantal}$ and DFT$_{coronal}$. This is because those features do not change in RN and are in fact well-predicted by the consonantal and coronal features in the input respectively.

However, for the nasal feature, SOSFIA should throw an error since regressive nasalization requires the interaction of two features to correctly predict the nasality as explained in the generalization (RN) in Section 2. In this particular example SOSFIA will send a failure signal because $\langle \Phi_{[nasal]}, \Phi_{[nasal]} \rangle(S)$ will most likely not be functional. This is because a segment that was [-nasal] in the input can change to [+nasal] or remain [-nasal] when followed by [+nasal]. The [consonantal] feature is also needed to determine whether a non-nasal phoneme in the input becomes nasal in the output.

To understand the impact the featural representation has on the size of the data sample, consider the sizes of the phonemic-based DFT and the n-many feature-based DFTs describing regressive nasalization. If Σ is the set of phonemes then the number of states will be $|\Sigma| + 1$ since RN is 2-ISL. When we consider the featural representations, there are n DFTs, one for each feature. With $k = 2$, the DFTs for features like consonantal will have 3 states (the initial state, and one for each $\{+, -\}$). The only DFT requiring more states is the one for nasal because SOSFIA will eventually apply the homomorphism $\langle \Phi_{[nasal,consonantal]}, \Phi_{[nasal]} \rangle$ on the sample. Since the size of the featural alphabet $|\Phi_{[nasal,consonantal]}| = 4$ this DFT will have 5 states. The number of states no longer depends on the number of phonemes, but instead on the number features interacting.

4. Preliminary experimental results

Preliminary experiments with regressive nasalization support our claim that feature based strategy uses less data. We repeatedly ran SOSFIA to find a minimally sized successful training sample S for both phonemic, letter-based representations (Σ), and featural-based representations using Φ. Table 1 shows the minimal number of strings that were needed for successfully learning where the number of phonemes was varied but the number of features was fixed at 4 (nasal, consonantal, coronal, sonorant).

We first built the phonemic and individual machines for each feature. Then we generate all logically possible input/output string pairs up to the length $k+1$, which meets the requirements of a characteristic sample. Next, using the structure of the machines we created, we "empty" the outputs and we feed the machine together with the appropriately projected training sample to SOSFIA. The algorithm made correct predictions in all the cases except for DFT$_{nasal}$. We then augmented DFT$_{nasal}$ to work with $\langle \Phi_{[nasal,consonantal]}, \Phi_{[nasal]} \rangle$ instead of $\langle \Phi_{[nasal]}, \Phi_{[nasal]} \rangle$. This time SOSFIA successfully output the expected DFT.

In order to find a minimal data sample, we subtracted one data point at a time from the original sample we generated and ran SOSFIA again to check whether this example is crucial for the function we are trying to learn. Repeating this process produced a minimal set of essential examples that are required for the training sample since the moment when SOSFIA fails to return the correct output, the process stops.

DFT Type	Σ size				
	6	7	9	11	14
seg	47	71	196	374	972
feat	62	62	62	62	62

Table 1: A minimal sample needed for SOSFIA depending on size of the input alphabet

While the phonemic machine requires slightly smaller sample for $\Sigma \leq 6$, the featural approach proves to be better with larger alphabet sizes. Since the size of natural languages alphabets are definitely larger than 6, the featural approach definitely looks promising.

5. Future work

While the featural approach to sequential function learning is promising, there is of course much to do. The most immediate tasks include running additional experiments on different processes, as well as automating the procedure by which the homomorphisms are refined. The fact that SOSFIA can indicate when a data sample is incompatible with the given DFT suggests that it can also begin with $k = 1$ and increase this value only when necessary. We plan to test for processes of varied complexities that require larger k values, as well as for simultaneous interactions of multiple processes. Finally, a further step will be to test the approach for natural data, i.e. one that resembles what children are exposed to in the early years of language acquisition.

References

Jane Chandlee. *Strictly Local Phonological Processes.* PhD thesis, University of Delaware, 2014.

Jane Chandlee. Computational locality in morphological maps. *Morphology*, 27(4):599–641, 2017.

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

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

Colin de la Higuera. Ten open problems in grammatical inference. In Yasubumi Sakakibara, Satoshi Kobayashi, Kengo Sato, Tetsuro Nishino, and Etsuji Tomita, editors, *International Colloquium of Grammatical Inference*, volume 4201 of *Lecture Notes in Computer Science*, pages 32–44. Springer, 2006.

Bruce Hayes. *Introductory Phonology.* Wiley-Blackwell, 2009.

Jeffrey Heinz and Regine Lai. Vowel harmony and subsequentiality. In Andras Kornai and Marco Kuhlmann, editors, *Proceedings of the 13th Meeting on Mathematics of Language*, Sofia, Bulgaria, 2013.

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.

Jerzy Rubach. *Cyclic and lexical phonology.* De Gruyter Mouton, Berlin, New York, 1984.