Empirical and theoretical arguments for using phonological features for the learning of subsequential functions

Magdalena Markowska and Jeffrey Heinz

Department of Linguistics Stony Brook

Stony Brook University

iACS Institute for Advanced Computational Science

LSA, 2024 Annual Meeting

Introduction

- Many phonological processes can be modeled with subsequential functions (Heinz and Lai, 2013; Chandlee et al., 2014; Jardine, 2016; Chandlee, 2017; Chandlee and Heinz, 2018).
- Such functions can be instantiated by deterministic finite state transducers (DFTs) (Sakarovitch, 2009).
- OSTIA (Onward Subsequential Transducer Inference Algorithm, Oncina et al., 1993) learns subsequential functions in cubic time, but is not practical for learning phonological processes (Gildea and Jurafsky, 1996).
- SOSFIA (Structured Onward Subsequential Function Inference Algorithm, Jardine et al., 2014) learns particular subclasses of subsequential functions in linear time and data.
- How practical is SOSFIA? It still requires a large sample.
- **This talk: Reducing the sample size with phonological features.**

OUR CONTRIBUTION

- We introduce a new algorithm, built on SOSFIA, which proposes a solution to learn particular subclasses of sequential functions from substantially smaller samples by **generalizing over phonological features**, as opposed to segments.
- We focus on a subclass of subsequential functions: **input strictly local** (ISL) functions.
  - Chandlee and Heinz (2018) shows that many phonological processes occurring in natural languages can be represented with such functions.
  - This class is well understood in the CS literature and goes by the names 'local' (Sakarovitch, 2009) and 'definite' (Lambert and Heinz, 2023).
  - Every k-ISL function can be represented with a DFT with about $|\Sigma|^k$ states and **each of these DFTs has the same structure**.
  - ISLFLA learns k-ISL functions in quadratic time and data (Chandlee et al., 2014).

Main Idea

1. The size of a sufficient sample is proportional to the size of the underlying DFT.
2. Using features lets us decompose the problem in parallel and reduce $|\Sigma|$ from the number of phonemes to the number of feature values.

OUTLINE

1. Background
   - ISL functions
   - SOSFIA
2. Our algorithm
   - theoretical implications
   - empirical evidence

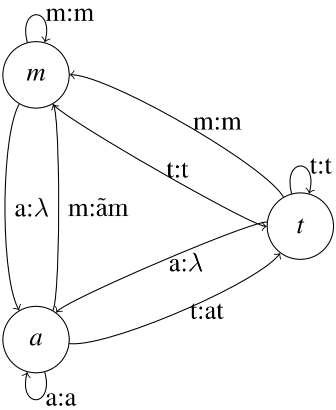
ISL functions in phonology

- **ISL functions**: states and transitions are organized in such a way that the current state of the machine is always determined by the previous $k - 1$ symbols on the input.
- Vowel Nasalization in English

  $[-\text{cons}] \longrightarrow [+\text{nasal}] \,/\, \_\_\_ \, [+\text{cons}, +\text{nasal}]$

  - it can be represented with a 2-ISL ($k = 2$) function since we only need to store in memory a vowel and whatever segment follows it
  - states in a DFT modelling a 2-ISL function are then represented with segments of length one

DFT FOR THE NASALIZATION



All other 2-ISL functions with this alphabet can be represented with the same DFT only changing output labels!

(start state and final function not shown)

SOSFIA

SOSFIA is an algorithm that can learn any class of subsequential functions which can be represented by a single DFT.
Every function in the class has the same DFT structure; only the outputs are different.

It takes two arguments:

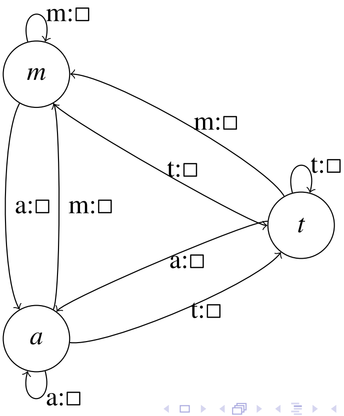
- a finite data sample of input-output pairs
- an output-empty transducer for the relevant class

How does SOSFIA operate?

Vowel Nasalization

[-cons] → [+nasal] / __ [+nasal, +cons]

...		...
/kæn/	→	[kæ̃n]
/kæt/	→	[kæt]
/mæn/	→	[mæ̃n]
/mæt/	→	[mæt]
...		...



How does SOSFIA operate?

1. SOSFIA makes string comparisons using the longest common prefix to determine the outputs of the transitions.
2. Provided there is a sufficient sample, SOSFIA is guaranteed to succeed.
3. A sufficient sample has a size which is linear in the size of the DFT.

(Jardine et al., 2014)

A minimal sample for Vowel Nasalization in English

- We use 43 English segments, 18 of which are vowels evenly divided by the feature [nasal].
- Given the alphabet Σ, we generated a sample S from all logically possible strings up to $k + 1$.
- $|S| = 81399$
- Then we ran SOSFIA and checked for success.

A minimal sample for Vowel Nasalization in English

- We use 43 English segments, 18 of which are vowels evenly divided by the feature [nasal].
- Given the alphabet Σ, we generated a sample S from all logically possible strings up to $k + 1$.
- $|S| = 81399$
- Then we ran SOSFIA and checked for success.

How can we make this sample substantially smaller?

A minimal sample for Vowel Nasalization in English

- We use 43 English segments, 18 of which are vowels evenly divided by the feature [nasal].
- Given the alphabet Σ, we generated a sample S from all logically possible strings up to $k + 1$.
- $|S| = 81399$
- Then we ran SOSFIA and checked for success.

How can we make this sample substantially smaller?

- By projecting necessary features from full segments and running SOSFIA on those projections.

A MINIMAL SAMPLE FOR VOWEL NASALIZATION IN ENGLISH

- We use 43 English segments, 18 of which are vowels evenly divided by the feature [nasal].
- Given the alphabet Σ, we generated a sample S from all logically possible strings up to $k + 1$.
- $|S| = 81399$
- Then we ran SOSFIA and checked for success.

How can we make this sample substantially smaller?

- By projecting necessary features from full segments and running SOSFIA on those projections.
- How do we do it?

PROJECTION OF FEATURES

Given a segment $\sigma \in \Sigma$, and a set of binary and ternary phonological features, a feature ϕ is a homomorphism from Σ to the set $\{+,-,0\}$.

Let Φ be an ordered set of features, and w be a string in Σ^*.

- Φ_{eng} ={'high', 'low', 'tense', 'front', 'back', 'round', 'long', 'cons', 'son', 'cont', 'delrel', 'approx', 'nasal', 'voice', 'lab', 'labdent', 'cor', 'ant', 'distr', 'strid', 'lat', 'dor'}
- w = 'mat'

Then:

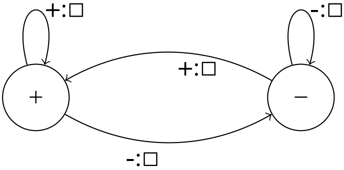
$\phi_{coronal}(mat) = -0+$

$\phi_{consonantal}(mat) = +-+$

$\Phi_{[coronal,consonantal]}(mat) = [-+][0-][++]$

Projecting features

- Instead of predicting segments in the output-empty transducer, our algorithm predicts features.
- Each feature is represented with a separate transducer.
- The states are the k-1 feature values for the feature the transducer represents, e.g. {+, -}

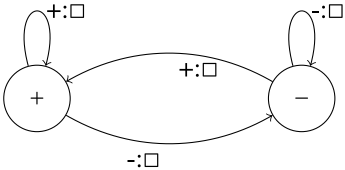


ISL Transducer representing a single binary feature with k=2 (initial state not shown).

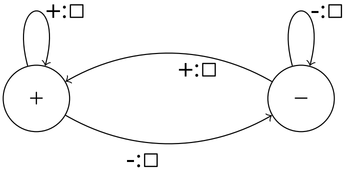
STEP I

I. Predicting a feature f from a single feature f.

DFT predicting CONS feature from CONS feature:



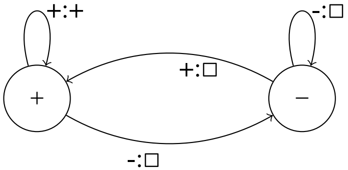
DFT predicting NASAL feature from NASAL feature:



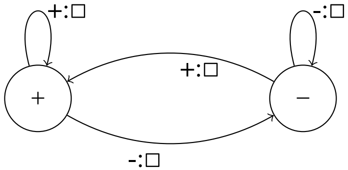
STEP I

I. Predicting a feature f from a single feature f.

DFT predicting CONS feature from CONS feature:



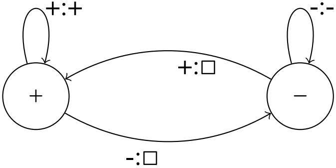
DFT predicting NASAL feature from NASAL feature:



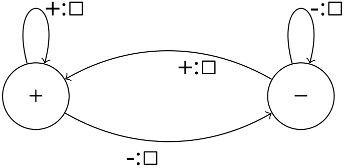
STEP I

I. Predicting a feature f from a single feature f.

DFT predicting CONS feature from CONS feature:



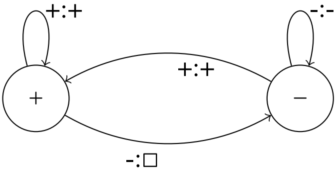
DFT predicting NASAL feature from NASAL feature:



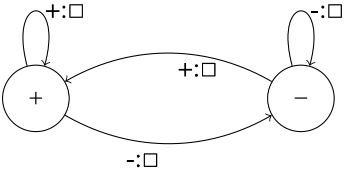
STEP I

I. Predicting a feature f from a single feature f.

DFT predicting CONS feature from CONS feature:



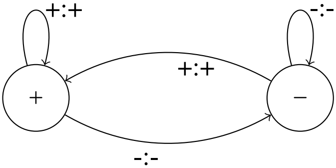
DFT predicting NASAL feature from NASAL feature:



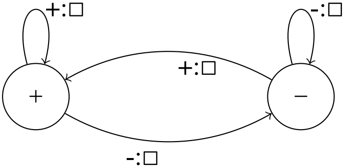
STEP I

I. Predicting a feature f from a single feature f.

DFT predicting CONS feature from CONS feature:



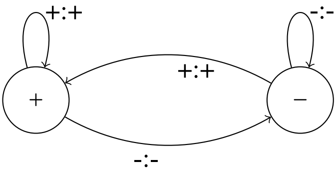
DFT predicting NASAL feature from NASAL feature:



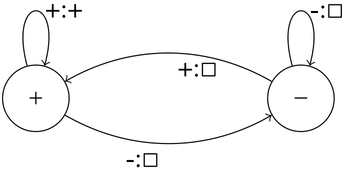
Step I

I. Predicting a feature f from a single feature f.

DFT predicting CONS feature from CONS feature:

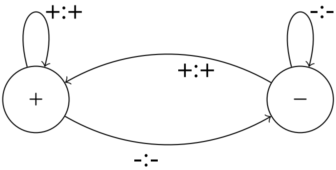


DFT predicting NASAL feature from NASAL feature:

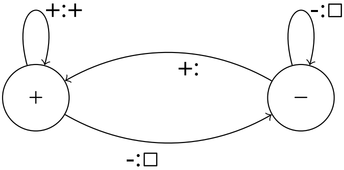


Step I

DFT predicting CONS feature from CONS feature:



DFT predicting NASAL feature from NASAL feature:



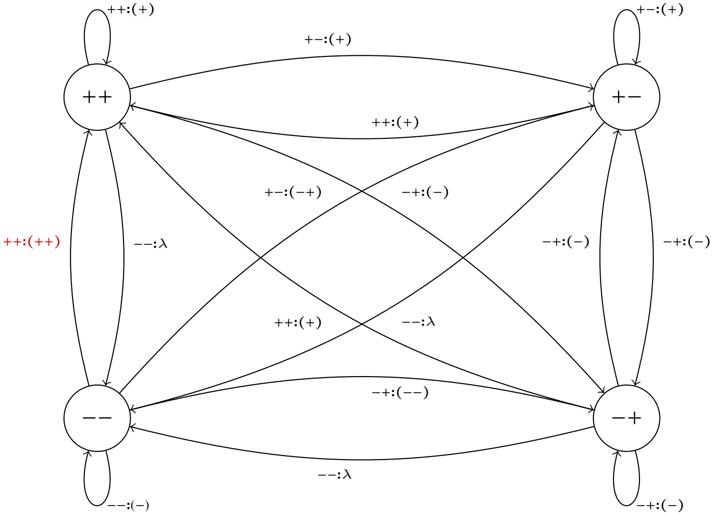
DANGER

The training sample for NASAL is not functional!

Step II

II. If a single feature is not enough to infer correct values from a single feature, add another one. Repeat until you find sufficient featural combination to infer that feature.

DFT predicting Nasal from Nasal and Cons



SOSFIA VS. OUR ALGORITHM: SOME RESULTS

Type of representation	Size of a min sample	Total
Segments from segments	81399	81399
NASAL from [NASAL, CONS]	84	
HIGH from HIGH	39	
LOW from LOW	39	
TENSE from TENSE	39	
FRONT from FRONT	39	
BACK from BACK	39	
ROUND from ROUND	14	753
LONG from LONG	14	
CONS from CONS	14	
SON from SON	14	
CONT from CONT	14	
...	..	

WHY DO FEATURES AFFECT A SUFFICIENT SAMPLE?

- The size of the type of transducers we are interested in depends on:
  - the complexity of the process – how many segments need to be remembered to make a correct prediction, e.g. vowel nasalization requires keeping track of the last read segment (state)
  - the size of Σ
- Since elements of Σ represent the states, it is easy to see that the transducer will quickly grow in size if we are operating on segments.
- In the featural transducers the state space is much smaller, e.g. $\Sigma = \{+, -\}$ for single binary feature and $\Sigma = \{++, --, +-, -+\}$ for 2 binary features, **regardless of the number of segments**.

WHY DO FEATURES AFFECT A SUFFICIENT SAMPLE?

k	$\lvert\Sigma_{seg}\rvert$	$\lvert Q_{seg}\rvert$	$\lvert\Sigma_{NasCon}\rvert$	$\lvert Q_{NasCon}\rvert$
	4	4		
2	10	10	4	4
	50	50		
	4	20		
3	10	110	4	20
	50	2550		

The relation between the size of the alphabet and the complexity of the process.

CONCLUSION AND FUTURE WORK

1. We presented a new algorithm, building on SOSFIA, that infers subsequential functions from featural representation of words.
    - We showed that with this approach, we obtain an exponential reduction in the size of the training sample.
2. We motivated our work by its application to phonological transductions observed in natural languages.
3. Future work will address issues with noise in data, which cannot be handled by SOSFIA at the moment, as well as learning the underlying representations (Hua et al., 2021).

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

Chandlee, J., R. Eyraud, and J. Heinz (2014). Learning strictly local subsequential functions. Transactions of the Association for Computational Linguistics 2, 491–504.

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

Gildea, D. and D. Jurafsky (1996). Learning bias and phonological-rule induction. Computational Linguistics 22(4), 497–530.

Heinz, J. and R. Lai (2013). Vowel harmony and subsequentiality. In A. Kornai and M. Kuhlmann (Eds.), Proceedings of the 13th Meeting on Mathematics of Language, Sofia, Bulgaria.

Hua, W., H. Dai, and A. Jardine (2021). Learning underlying representations and input-strictly-local functions. In D. K. E. Reisinger and M. Huijsmans (Eds.), Proceedings of the 37th West Coast Conference on Formal Linguistics, pp. 143–151. Cascadilla Proceedings Project.

Jardine, A. (2016). Computationally, tone is different. Phonology 32(2), 247–283.

Jardine, A., J. Chandlee, R. Eyraud, and J. Heinz (2014, September). Very efficient learning of structured classes of subsequential functions from positive data. In A. Clark, M. Kanazawa, and R. Yoshinaka (Eds.), Proceedings of the Twelfth International Conference on Grammatical Inference (ICGI 2014), Volume 34, pp. 94–108. JMLR: Workshop and Conference Proceedings.

Lambert, D. and J. Heinz (2023). An algebraic characterization of total input strictly local functions. In Proceedings of the Society for Computation in Linguistics, Volume 6.

Oncina, J., P. García, and E. Vidal (1993, May). Learning subsequential transducers for pattern recognition tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence 15, 448–458.

Sakarovitch, J. (2009). Elements of Automata Theory. Cambridge University Press. Translated by Reuben Thomas from the 2003 edition published by Vuibert, Paris.

How does SOSFIA operate?

The outputs are calculated with two functions:

- common_out – given a substring, it returns the *longest common prefix* (lcp) of all the outputs associated with inputs starting with that prefix
- min_change – given a substring and its extension, it returns the difference of *common_out* of this substring and its extension

HOW DOES *common_out* WORK?

$$common\_out_S(w) = lcp(\{x \in \Sigma^* | \exists v \ s.t. \ (wv, u) \in S\})$$

aaa	$\longrightarrow$	aaa
ama	$\longrightarrow$	ãma
mma	$\longrightarrow$	mma
mam	$\longrightarrow$	mãm
ata	$\longrightarrow$	ata
mta	$\longrightarrow$	mta
mat	$\longrightarrow$	mat
amt	$\longrightarrow$	ãmt

How does *common_out* work?

$$common\_out_S(a) = \quad ???$$

HOW DOES *common_out* WORK?

$common\_out_S(a) =$???

aaa	$\longrightarrow$	aaa
ama	$\longrightarrow$	ãma
mma	$\longrightarrow$	mma
mam	$\longrightarrow$	mãm
ata	$\longrightarrow$	ata
mta	$\longrightarrow$	mta
mat	$\longrightarrow$	mat
amt	$\longrightarrow$	ãmt

HOW DOES *common_out* WORK?

$$common\_out_S(a) = \lambda$$

$$\begin{array}{lcl} \text{aaa} & \longrightarrow & \text{aaa} \\ \text{ama} & \longrightarrow & \tilde{\text{a}}\text{ma} \\ \text{mma} & \longrightarrow & \text{mma} \\ \text{mam} & \longrightarrow & \text{m}\tilde{\text{a}}\text{m} \\ \text{ata} & \longrightarrow & \text{ata} \\ \text{mta} & \longrightarrow & \text{mta} \\ \text{mat} & \longrightarrow & \text{mat} \\ \text{amt} & \longrightarrow & \tilde{\text{a}}\text{mt} \end{array}$$

HOW DOES *common_out* WORK?

$$common\_out_S(am) = \quad ???$$

aaa	$\longrightarrow$	aaa
ama	$\longrightarrow$	ãma
mma	$\longrightarrow$	mma
mam	$\longrightarrow$	mãm
ata	$\longrightarrow$	ata
mta	$\longrightarrow$	mta
mat	$\longrightarrow$	mat
amt	$\longrightarrow$	ãmt

HOW DOES *common_out* WORK?

$$common\_out_S(am) = \quad \tilde{a}m$$

aaa	$\longrightarrow$	aaa
ama	$\longrightarrow$	ãma
mma	$\longrightarrow$	mma
mam	$\longrightarrow$	mãm
ata	$\longrightarrow$	ata
mta	$\longrightarrow$	mta
mat	$\longrightarrow$	mat
amt	$\longrightarrow$	ãmt

HOW DOES *min_change* WORK?

$$min\_changes_S(\sigma, w) = \begin{cases} common\_outs_S(\sigma) & \text{if } w = \lambda \\ common\_outs_S(w)^{-1} common\_outs_S(w\sigma) & otherwise \end{cases}$$

- $common\_outs_S(\text{a}) = \lambda$
- $common\_outs_S(\text{am}) = \text{ãm}$
- $min\_changes_S(a, m) = \text{ãm}$

SOSFIA's output