Learning ISL functions with features

Magdalena Markowska

1 Motivation

Various small, interpretable models have been proposed for learning morpho-phonological processes represented as deterministic finite-state transducers (DFTs). Those include: the Onward Subsequential Transducer Inference Algorithm (OSTIA, Oncina et al. (1993)) which employs the technique of state-merging to learn the function; the Input Strictly Local Functions Learning Algorithm (ISLFLA, Chandlee et al. (2014)) which learns particularly k-ISL functions; the Structured Onward Subsequential Function Inference Algorithm (SOSFIA, Jardine et al. (2014)) which calculates the minimal change that needs to be made when transitioning from one state to another.

All of these algorithms operate over segments and require a relatively large and often very particular sample to successfully infer the function. We show that by changing the representation of data from segments to collections of ordered features, we can successfully reduce the size of a characteristic sample. As a result, we introduce an algorithm built on SOSFIA that breaks down the learning problem into smaller parts by simultaneously inferring the target function for each feature $\phi \in F$ separately. Such decomposition results in substantially smaller DFTs, which consequently need less data. This approach is particularly appealing for low-resource languages for which the available data is very much limited.

2 Factoring by features

Here we assume a contrastive set of features, where each feature $\phi \in F$ is assigned any value from the set $\{+, -, 0\}$.

	a	a:	m	t
cons	-	-	+	+
nasal	-	-	+	-
long	-	+	0	0
cor	0	0	-	+

Table 1: Feature chart for $\Sigma = \{a, a:, m, t\}$ and $F = \{cons, nasal, long, cor\}$.

A feature $\phi \in F$ is then a function from Σ to the set $\{+, -, 0\}$, which can be lifted to a homomorphism from Σ^* to $\{+, -, 0\}^*$. Eg.

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

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

If Φ is an ordered set of features, then Φ is a pointwise product of its homomorphisms. Eg.

$$\Phi_{[cor,nasal]}(mat) = [-+][0-][+-]$$

Given a sample S of input-output pairs, let $\langle \Phi_X, \Phi_Y \rangle(S) = \{(\Phi_X(x), \Phi_Y(y)) : (x, y) \in S\}$.

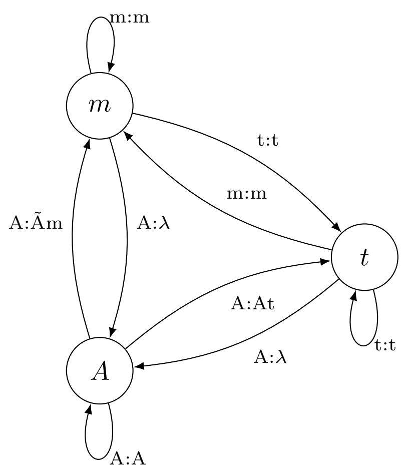
The target function $f : \Sigma^* \to \Delta^*$ is decomposed into n-many functions, one for each $\phi \in F$. For the moment we focus on k-ISL functions (Chandlee, 2014). Each feature ϕ is individually represented with a k-ISL DFT, $\tau_{(\Phi,\phi)}$, where ϕ is the feature for which the outputs will be predicted, and Φ is a combination of features used to predict ϕ.

2.1 Example

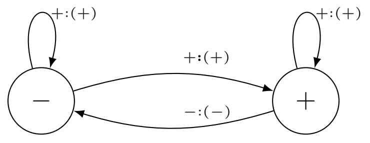
Let us consider a 2-ISL function that represents a vowel vowel nasalization process in English:

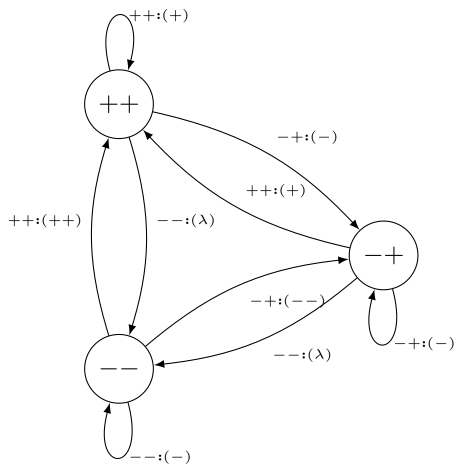
- [-cons] $\longrightarrow$ [+nasal] / ___ [+cons, +nasal]

Given the previously defined Σ, a 2-ISL DFT looks as follows. The A state represents both a and $a:$, but the full machine would have those states separated. Boundary states ($\ltimes, \rtimes$) are not shown.



Now we project every feature individually. Below are shown 2-ISL $\tau_{([cons],cons)}$, on the left, and 2-ISL $\tau_{([nasal,cons],nasal)}$, on the right.





3 How does the new algorithm operate?

SOSFIA FxF iterates through every feature $\phi \in F$ and searches for the right featural combination (if needed) to determine the outputs for that feature. It is done by first extracting the featural sample $\langle \Phi_X, \Phi_Y \rangle(S) = \{(\Phi_X(x), \Phi_Y(y)) : (x,y) \in S\}$ and checking if it is functional. It it is, a $k-$ISL DFT is projected from the 'original' segmental machine. If it is not, the algorithm considers 2 features on the input, exctracts a new sample and checks if it's functional. And so on.

```
Data: An output empty DFT $\tau_\square = (Q, \Sigma, \Delta = \{\square\}, q_0, q_f, \delta)$, a sample $S \subset \rtimes\Sigma^* \ltimes \times\Delta^*$, a feature set $F$
Result: G containing $n$ many DFT$_{(\Phi,\phi)}$s, one for each $\phi \in F$
G ← $\emptyset$
for $\phi$ in $F$ do
    $\tau_{(\Phi,\phi)}$ ← FeatSearch($\tau_\square$, S, F, $\phi$, [$\{\phi\}$]);
    G ← G $\cup\{\tau_{(\Phi,\phi)}\}$;
end
return G
```

Algorithm 1: SOSFIA Factor by Feature (SOSFIA FxF)

Definition 1. Given a function f, a data sample S, projection functions Φ_x and Φ_y, is functional iff:

$$(u,v),(u',v') \notin S, \text{ where } \Phi_x(u) = \Phi_x(u') \text{ and } \Phi_y(v) \neq \Phi_y(v')$$

Definition 2. The feat_combo(ϕ,Φ,F) of a feature ϕ is a list of combinations of length $|\Phi| + 1$ of all features in F with ϕ with no repetitions allowed:

$$feat\_combo(\phi, \Phi, F) = \{\{\phi\} \cup \Phi \,|\Phi \subseteq F \setminus \{\phi\}\}$$

```
Data: An output empty DFT $\tau_{\square} = (Q, \Sigma, \Delta = \{\square\}, q_0, q_f, \delta)$, a sample $S \subset \rtimes\Sigma^* \ltimes \times\Delta^*$, a feature set $F$, a
      feature $\phi$, a list of sets of features $FeatLIST \subseteq F$
Result: a DFT $\tau_{(\Phi,\phi)}$
while FeatLIST is not empty do
    $\Phi \leftarrow$ dequeue(FeatLIST);
    $S_\phi \leftarrow$ factor_S($S, \phi, \Phi$);
    if check_if_functional($S_\phi$) then
        return SOSFIA(factor_DFT($\tau_{\square}, \phi, \Phi$), $S_\phi$);
    end
end
NewFeatLIST $\leftarrow$ feat_combo($\phi, \Phi, F$);
FeatSearch($\tau_{\square}$, S, F, $\phi$, NewFeatLIST);
```

Algorithm 2: FeatSearch

4 Case studies

In the particular case of the k-ISL functions inference, the minimum size of the longest string in a sufficient sample equals $2k - 1$. The properties of k-ISL DFTs and the way SOSFIA operates give us an intuition to why that is. The depth of a k-ISL DFT is $k - 1$ long. In other words, while in one of the furthest states from $\rtimes$, excluding $\ltimes$, the machine has been passed a string of minimal length equal to $k - 1$. However, from the definition of k-ISL functions the shortest substring that needs to be considered at a time is k-long. It is then necessary to allow for a possibility of the k-long substring to be passed through a DFT from *every* state $q \in Q$. Hence, the sum of the depth of a k-ISL DFT (k-1) and the memory window (k) is equal to $2k - 1$.

In order to find a smallest characteristic sample, we also employ another algorithm that recursively chooses 10 random input-output pairs from the CS until it SOSFIA FxF successfully find the sufficient one.

4.1 Sample types

- an original sample (**S**): input-output pairs generated by the original DFT, where each segment is represented with values for *all* features.
- a minimal *segments* sample (**M**): a minimal sample randomly drawn from S with AMβA calling the original SOSFIA algorithm.
- a feature sample ($S_{(\Phi,\phi)}$): a sample extracted from S for each feature $\phi \in F$. For example, $S_{([\text{high}],\text{high})}$ is a sample extracted from S to predict [high] from [high], while $S_{([\text{high, round}],\text{high})}$ means that two features [high] and [round] are projected to predict [high].
- a minimal *all* features sample (M_F): a minimal sample randomly drawn from S with AMβA calling SOSFIA FxF.
- a minimal *feature* sample ($M_{(\Phi,\phi)}$): a minimal sample drawn from $M_{(\Phi,\phi)}$ for each feature $\phi \in F$. For example, $M_{([\text{high}],\text{high})}$ is a sample extracted from $M_{(\Phi,\phi)}$ to predict [high] from [high], while $M_{([\text{high, round}],\text{high})}$ means that two features [high] and [round] are projected to predict [high].
- max($M_{(\Phi,\phi)}$) is the maximum cardinality of all $M_{(\Phi,\phi)}$s for all $\phi \in F$.
- the union of the $M_{(\Phi,\phi)}$ samples ($\mathbf{U}_\phi$): $\bigcup_{\phi \in F} M_{(\Phi,\phi)}$. In case $|U_\phi| \geq |M_F|$, we set U_ϕ equal to $M_{(\Phi,\phi)}$.

4.2 English vowel nasalization

Given the alphabet of English phonemes of length $|\Sigma| = 26$, a characteristic sample of strings of length up to 3 was generated ($|S| = 18,278$).

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**nasal**, cons]	84	83	1.14
[**nasal**, long]	84	83	1.14
[**nasal**, approx]	258	243	5.56
[**nasal**, lab]	258	243	9.90
[**nasal**, labdent]	155	152	2.83
[**nasal**, cor]	258	248	5.07
[**nasal**, lat]	258	232	5.12
[**nasal**, dor]	258	240	4.26
[**delrel**]	39	38	0.67
[**voice**]	14	14	0.42

Table 2:
The size of $S_{(\Phi,\phi)}$, derived by SOSFIA FxF, averaged $M_{(\Phi,\phi)}$ samples, derived by AMβA, when run on SOSFIA FxF, and standard deviation (SD) for selected sufficient and insufficient features in inferring the English VN function.

segments		**features**		
$\lvert S\rvert$	$\lvert M\rvert$	max($M_{(\mathbf{\Phi},\boldsymbol{\phi})}$)	Avg. $\lvert M_F\rvert$	$\lvert U_\phi\rvert$
18,278	7,430	83	4,142	904

Table 3: Summary of five sample sizes for the segmental and all featural machines combined in inferring the English NV function.

4.3 Yawelmani vowel shortenning

- [-cons, +long] $\longrightarrow$ [-long] / ___ [+cons][+cons]

Because this process requires 3-ISL function, we had to use much smaller alphabet to generate the sample of approximate size to the previous case study. Here, $|\Sigma| = 7$ and $|S| = 19,607$

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**long**]	363	360	2.63
[**cons**]	62	62	0.53
[**nasal**]	62	58	3.38
[**ant**]	363	316	29.99

Table 4:
The size of $S_{(\Phi,\phi)}$, derived by SOSFIA FxF, averaged $M_{(\Phi,\phi)}$ samples, derived by AMβA when run on SOSFIA FxF, and standard deviation (SD) for selected sufficient and insufficient features in inferring the Yawelmani VS function.

segments		**features**		
$\lvert S\rvert$	$\lvert M\rvert$	max($M_{(\mathbf{\Phi},\boldsymbol{\phi})}$)	Avg. $\lvert M_F\rvert$	$\lvert U_\phi\rvert$
19,607	19,170	360	9,158	4,075

Table 5: Summary of five sample sizes for the segmental and all featural machines combined in inferring the Yawelmani VS function.

4.4 Final epenthesis in Chukchi

- ∅⟶ ə / [+cons]___[+cons]⋉

Feature set Φ	$\lvert S_{(\Phi,\phi)}\rvert$	Avg. $\lvert M_{(\Phi,\phi)}\rvert$	SD
[**round**, high]	1,364	1,347	12.05
[**round**, back]	363	363	0.58
[**strid**, low]	1,364	1,361	3.83
[**strid**, front]	3,905	3,886	8.54
[**approx**]	62	62	0
[**lab**]	363	363	0

Table 6:
The size of $S_{(\Phi,\phi)}$, derived by SOSFIA FxF, averaged $M_{(\Phi,\phi)}$ samples, derived by AMβA when run on SOSFIA FxF, and standard deviation (SD) for selected sufficient and insufficient features in inferring the Chukchi FE function.

segments		**features**		
$\lvert S\rvert$	$\lvert M\rvert$	$\max(M_{(\boldsymbol{\Phi},\boldsymbol{\phi})})$	Avg. $\lvert M_F\rvert$	$\lvert U_\phi\rvert$
19,607	19,260	3,886	18,786	18,786

Table 7: Summary of five sample sizes for the segmental and all featural machines combined in inferring the Chukchi FE function.

4.5 Polish *o*-raising and final devoicing

- [-sonorant] ⟶ [-voice] / ___ ⋉
- [-cons, +back, -low] ⟶ [+high] / ___ [+cons, +voice, -nasal]⋉

Feature set Φ	**Σ = {t,d,n,ʂ,a,ɔ,u}**		**Σ ={t,d,n,ʂ,ʐ,a,ɔ,u}**		**Σ ={t,d,n,ʂ,ʐ,a,ɔ,ɔ̃,u}**		**Σ ={t,d,n,ʂ,ʐ,d͡ʑ,a,ɔ,ɔ̃,u}**	
	$\lvert\mathbf{S}_{(\Phi,\phi)}\rvert$	$\lvert\mathbf{M}_{(\Phi,\phi)}\rvert$	$\lvert\mathbf{S}_{(\Phi,\phi)}\rvert$	$\lvert\mathbf{M}_{(\Phi,\phi)}\rvert$	$\lvert\mathbf{S}_{(\Phi,\phi)}\rvert$	$\lvert\mathbf{M}_{(\Phi,\phi)}\rvert$	$\lvert\mathbf{S}_{(\Phi,\phi)}\rvert$	$\lvert\mathbf{M}_{(\Phi,\phi)}\rvert$
$\begin{bmatrix}\textbf{voice}\\ \textit{son}\end{bmatrix}$	363	363	363	363	363	363	363	363
$\begin{bmatrix}\textbf{voice}\\ \textit{delrel}\end{bmatrix}$	1,364	1,361	3,905	3,871	3,905	3,879	3,905	3,789
$\begin{bmatrix}\textit{voice} & \textbf{high}\\ \textit{son} & \textit{low}\end{bmatrix}$	9,330	9,308	9,330	9,295	-	-	-	-
$\begin{bmatrix}\textit{voice} & \textbf{high}\\ \textit{son} & \textit{round}\end{bmatrix}$	9,330	9,308	9,330	9,295	-	-	-	-
$\begin{bmatrix}\textit{voice} & \textbf{high}\\ \textit{delrel} & \textit{low}\end{bmatrix}$	19,607	-	37,448	-	-	-	-	-
$\begin{bmatrix}\textit{voice} & \textbf{high}\\ \textit{delrel} & \textit{round}\end{bmatrix}$	19,607	-	37,448	-	-	-	-	-
$\begin{bmatrix}\textit{voice} & \textbf{high}\\ \textit{nasal} & \textit{low}\end{bmatrix}$	9,330	-	9,330	-	19,607	19,522	19,607	19,497
$\begin{bmatrix}\textit{voice} & \textbf{high}\\ \textit{nasal} & \textit{round}\end{bmatrix}$	9,330	-	9,330	-	19,607	19,522	19,607	19,497
$\begin{bmatrix}\textit{voice} & \textit{low}\\ \textit{son} & \textbf{tense}\end{bmatrix}$	9,330	9,308	9,330	9,295	-	-	-	-
$\begin{bmatrix}\textit{voice} & \textbf{tense}\\ \textit{son} & \textit{round}\end{bmatrix}$	9,330	9,308	9,330	9,295	-	-	-	-
$\begin{bmatrix}\textit{voice} & \textit{low}\\ \textit{delrel} & \textbf{tense}\end{bmatrix}$	19,607	-	37,448	-	-	-	-	-
$\begin{bmatrix}\textit{voice} & \textbf{tense}\\ \textit{delrel} & \textit{round}\end{bmatrix}$	19,607	-	37,448	-	-	-	-	-
$\begin{bmatrix}\textit{voice} & \textit{low}\\ \textit{nasal} & \textbf{tense}\end{bmatrix}$	9,330	-	9,330	-	19,607	19,522	19,607	19,497
$\begin{bmatrix}\textit{voice} & \textbf{tense}\\ \textit{nasal} & \textit{round}\end{bmatrix}$	9,330	-	9,330	-	19,607	19,522	19,607	19,497

$\|\Sigma\|$	**segments**	**features**		
	$\|S\|$	$\max(M_{(\Phi,\phi)})$	$\|M_F\|$	$\|U_\phi\|$
7	19,607	9,308	19,500	19,500
8	37,448	9,295	36,350	36,350
9	66,429	19,522	64,900	64,900
10	111,110	19,497	103,100	103,100

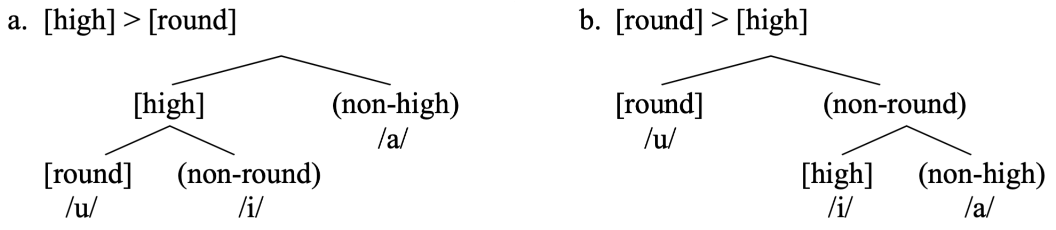
Table 8: Summary of four sample sizes for the segmental and all featural machines combined in inferring the composition of FD and OR functions in Polish with the incremental increase of the alphabet.

5 Identity as a default

In another set of experiments, we changed the output-empty DFT, an argument to SOSFIA and SOSFIA FxF, to identity one with the assumption that it will require a smaller sample to make right predictions. The assumption significantly affected only the size of the averaged sizes of the M_F for the English and Yawelmani cases. In the former, Avg M_F was reduced approximately by half (from 4143 to 1838). In the latter the size was reduced from 9158 to 7696. In the more complex cases, no significant differences were observed.

6 Future work

- Instead of pre-determining k value, we could start with $k = 1$ by default, and increase it only when necessary. The assumption here is that there will be features that will never change their values and could be represented with one state DFT.
- The feature search process is still pretty expensive. Two ideas on how to address the issue:
  - consider only constrastive features
  - organize features s.t. the combinations are 'easier' to find
    Perhaps both can be resolved by implementing Dresher's Successive Division Algorithm (Dresher, 2009)



- Here we presented the feature factoring approach on SOSFIA FxF, but this algorithm can be replaced with other ones.
- learning lexicon: Chandlee and Jardine (prep)

References

Chandlee, J. (2014). *Strictly Local Phonological Processes.* Ph. D. thesis, University of Delaware.

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 A. Jardine (inprep). Using locality and natural classes to infer underlying representations and a phonological grammar.

Dresher, E. B. (2009). *The contrastive hierarchy in phonology.* Cambridge: Cambridge University Press.

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.

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.