Collaborators

Jane Chandlee (Haverford)

Remi Eyraud (Aix-Marseille)

Jeff Heinz (SBU)

Adam Jardine (Rutgers)
The Talk in a Nutshell

Previously on this Topic

- Efficient Learning of Segmental Phonotactics and Mappings
- Question: How to extend these learners for feature-based constraints? (Cf. Hayes and Wilson 2008)

Today We

- Describe the partial order structure of the space of feature-based constraints
- Show how Learners can utilize this order to generalize constraints.
The Talk in a Nutshell

Previously on this Topic

- Efficient Learning of Segmental Phonotactics and Mappings
- Question: How to extend these learners for feature-based constraints? (Cf. Hayes and Wilson 2008)

Today We

- Describe the partial order structure of the space of feature-based constraints
- Show how Learners can utilize this order to generalize constraints.
The Challenge of Features

Consider:

\[ *NT \rightarrow \{ *nt, *np, *nk, *mt, *mp, *mk, \ldots \} \]

Wilson & Gallagher 2018

“Could there be a non-statistical model that learns by memorizing feature sequences? The problem confronting such a model is that any given segment sequence has may different featural representations. Without a method for deciding which representations are relevant for assessing wellformedness (the role that statistics plays in Maxent-Ftr) learning is doomed.”
Example

Imagine the sequence $nt$ is not present in a corpus. There are many possible equivalent constraints:

* $nt$
  * $[+\text{nasal}][+\text{coronal}]$
  * $[+\text{consonant}][+\text{coronal},-\text{continuant}]$
  * $[+\text{sonorant}][-\text{sonorant}]$
  ....

How can a learner decide which of these constraints is responsible for the absence of $nt$?
Example

Imagine the sequence *nt* is not present in a corpus. There are many possible equivalent constraints:

*nt
* [+nasal][+coronal]
*[+consonant][+coronal,-continuant]
*[+sonorant][-sonorant]
....

How can a learner decide which of these constraints is responsible for the absence of *nt*?
Imagine the sequence *nt is not present in a corpus. There are many possible equivalent constraints:

*nt
* [+nasal][+coronal]
* [+consonant][+coronal,-continuant]
* [+sonorant][-sonorant]

....

How can a learner decide which of these constraints is responsible for the absence of *nt?
Constraint Explosion (Hayes and Wilson 2008)

As we add segments, the amount of feature-based constraints grows larger.

How much larger?

- $S =$ number of segments
- $F =$ number of features, given by UG
- $V =$ number of feature values $+ 1$ (for underspecification)
- Number of possible constraints $= V^{FS}$
- Possible Constraint Sets $= \mathcal{P}((V^F)^S) = 2^{V^{FS}}$

Example

<table>
<thead>
<tr>
<th>$S$</th>
<th>$2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>$3$</td>
</tr>
<tr>
<td>$V$</td>
<td>$2+1$ $= 3$</td>
</tr>
<tr>
<td>$\mathcal{P}((V^F)^S)$</td>
<td>$2^{720}$</td>
</tr>
</tbody>
</table>
Constraint Explosion (Hayes and Wilson 2008)

Table 2
Number of possible constraints for various values of $|C|$ and $n$

|  |  $|C|$  |
|---|---|---|---|---|
|   | 30 | 100 | 200 | 400 |
| 1 | 30 | 100 | 200 | 400 |
| 2 | 900 | 10,000 | 40,000 | 160,000 |
| $n$ 3 | 27,000 | 1,000,000 | 8 million | 64 million |
| 4 | 810,000 | 100 million | 1.6 billion | 26 billion |
| 5 | 24 million | 10 billion | 320 billion | 10 trillion |

$|C|$ is the number of natural classes and $n$ is the length of the constraint.
<table>
<thead>
<tr>
<th>Hayes &amp; Wilson 2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Given an innate feature system, order a list of possible featural constraints by constraint length and generality.</td>
</tr>
<tr>
<td>▶ Input a batch of feature bundle strings as learning data</td>
</tr>
<tr>
<td>▶ Use MaxEnt and Observed/Expected ratios to discover the most general constraints.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Why no structure in the constraint space?</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ The nature of features gives a particular order and structure to the space of possible constraints.</td>
</tr>
<tr>
<td>▶ Let the learner exploit this structure as much as possible when making inferences from data.</td>
</tr>
<tr>
<td>▶ Use its size to our advantage!</td>
</tr>
</tbody>
</table>
Feature Extension

Definition (Feature Extensions)

Let $s$ and $t$ be segments represented as bundles of n-ary features.

Then $t$ is an **feature extension** of $s$ for grammar $G$ ($s \prec_G t$) iff $t$ is the result of inserting one or more n-ary features of $G$ in $s$.

Example

```
[-N,+V,+C]

[-N,+V]         [-N,+C]
          /           /
        [-N]       [-N]
```
Feature Entailments

Feature Ideals

If \( t \) is a feature extension of \( s \) for \( G \) and \( G \) generates \( t \), then \( G \) generates \( s \).

Example

\[
\begin{align*}
[-N, +V, +C] & \\
\quad & [-N, +V] \quad [-N, +C] \\
\quad & [-N] 
\end{align*}
\]
Feature Entailments

Feature Ideals

If \( t \) is a feature extension of \( s \) for \( G \) and \( G \) generates \( t \), then \( G \) generates \( s \).

Example

```
[-N, +V, +C]

[-N, +V]

[-N]

[-N, +C]
```
Feature Entailments

Feature Ideals

If \( t \) is a feature extension of \( s \) for \( G \) and \( G \) generates \( t \), then \( G \) generates \( s \).

Example

\[
\begin{array}{c}
[-N,+V]
\\
\checkmark
\\
[-N]
\\
\end{array}
\quad
\begin{array}{c}
[-N,+C]
\\
[-N,+V,+C]
\\
\end{array}
\]
Feature Entailments

**Feature Ideals**

If \( t \) is a feature extension of \( s \) for \( G \) and \( G \) generates \( t \), then \( G \) generates \( s \).

**Example**

![Feature Entailment Diagram]

- If \([-N,+V,+C]\) is a feature extension of \([-N,+V]\) for \( G \) and \( G \) generates \([-N,+V]\), then \( G \) generates \([-N]\).
Feature Entailments

Feature Ideals

If $t$ is a feature extension of $s$ for $G$ and $G$ generates $t$, then $G$ generates $s$.

Example

\[
\begin{array}{c}
\checkmark \quad [-N,+V] \\
\downarrow
\end{array} \\
\begin{array}{c}
[-N,+V,+C] \\
\downarrow \\
[-N,+C] \\
\downarrow \\
[-N]
\end{array}
\]
Feature Entailments

Feature Ideals

If $t$ is a feature extension of $s$ for $G$ and $G$ generates $t$, then $G$ generates $s$.

Example

```
[-N]
([-N,+V]

([-N,+C]

([-N]

[[-N,+V,+C]
```

✓

*
Feature Entailments

Feature Ideals

If \( t \) is a feature extension of \( s \) for \( G \) and \( G \) generates \( t \), then \( G \) generates \( s \).

Example

\[
\begin{align*}
[-N] & \quad \checkmark \quad [-N, +V] \quad \checkmark \quad [-N, +V, +C] \quad & \Rightarrow \quad [-N, +C] \\

\end{align*}
\]
Feature Entailments

Feature Ideals

If $t$ is a feature extension of $s$ for $G$ and $G$ generates $t$, then $G$ generates $s$.

Example
Parallels to Logical ‘And’

- DOWNWARD ● Grammaticality is Downward Entailing w.r.t. $<_G$
  $a \land b = 1$ implies $a = 1$

- UPWARD ➹ ungrammaticality is upward entailing w.r.t. $<_G$
  $a = 0$ implies $a \land b = 0$

Example

Grammars are Collections of Ideals and Filters

Definition (Ideal)

A non-empty subset $S$ of a poset $\langle A, \leq \rangle$ is an **ideal** iff

- for every $x \in S$, $y \leq x$ implies $y \in S$, and
- for all $x, y \in S$ there is some $z \in S$ s.t $x \leq z$ and $y \leq z$. 
Example with Singular Segments

Input Data: Feature Strings of Length 1

Suppose we observe in a language

- \([+N,+V,+C]\) (voiced nasal consonants),
- \([-N,+V,+C]\) (voiced nonnasal consonants),
- \([-N,-V,+C]\) (voiceless nonnasal consonants),
- \([-N,+V,-C]\) (voiced nonnasal vowels),

What constraints ought to be posited?

\([-N,-V,+C]\) is a feature extension of \([-N,-V]\), \([-N,+C]\), \([-V,+C]\). These are feature extensions of \([-N]\), \([-V]\), \([+C]\). And the empty feature bundle \([\emptyset]\).
Example with Singular Segments
Strictly Local Example: English *sʃ*

**Banned Structure**

\[
\begin{array}{c}
+\text{str} & +\text{str} \\
+\text{ant} & -\text{ant}
\end{array}
\]

\[\begin{array}{c}
1 \\
\prec \\
2
\end{array}\]
Strictly Local Example: English *sʃ*

**Banned Structure**

```
  +str   +str
  +ant   -ant
  1 ➔ 2
```

[Diagram showing banned structures with +str, +ant, -ant features]
Unbounded Dependencies

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

  (1)  
  a.  * hasxintilawaʃ 
  b.  * haʃxintilawas 
  c.  haʃxintilawaʃ

Example: Samala

*$ hasxintilawaʃ$ 

$ haʃxintilawas$
Unbounded Dependencies

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

  (1) a. * hasxintilawaʃ
      b. * haʃxintilawas
      c. haʃxintilawaʃ

Example: Samala

* $hasxintilawaʃ$

$haʃxintilawas$
Unbounded Dependencies

- Samala Sibilant Harmony
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

(1) a. * hasxintilawas
    b. * haʃxintilawas
    c. haʃxintilawas

Example: Samala

*$\text{hasxintilawas}*$

*$\text{haʃxintilawas}*$
Unbounded Dependencies

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

(1) a. * hasxintilawaʃ
    b. * haʃxintilawas
    c. haʃxintilawaʃ

Example: Samala

* $ha\{s\}x\{i\}n\{t\}i\{l\}a\{w\}a\{a\}$*

$ha\{a\}ʃx\{i\}n\{t\}i\{l\}a\{w\}a\{a\}$
Unbounded Dependencies

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

  (1) a. * hasxintilawaʃ 
  b. * haʃxintilawas 
  c. haʃxintilawaʃ 

Example: Samala

- But: Sibilants can be arbitrarily far away from each other!

  *$ s t a j a n o w o n w a ʃ $*
Unbounded Dependencies

- **Samala Sibilant Harmony**
  Sibilants must not disagree in anteriority.
  (Applegate 1972)

(1) a. *haśxintilawaʃ*
    b. *haʃxintilawas*
    c. haʃxintilawaʃ

Example: Samala

- **But**: Sibilants can be arbitrarily far away from each other!

*$s\text{tajawonwaʃ}$*

* $h a s x i n t i l a w a ʃ\$*

*$s t a j a n o w o n w a ʃ\$*
Two Representations of Order

1. Successor (Immediate Precedence)

![Diagram of Successor (Immediate Precedence)]

2. General precedence

![Diagram of General Precedence]
Example: Samala Log-Distance *sf

Banned Structure

```
+str  +str
+ant -ant
```

Diagram showing banned structures and their constraints.
Example: Samala Log-Distance *sj*

**Banned Structure**

```
+str +str
+ant -ant
```

1 → 2

```
[+voi] [+str]
[+str] [+ant]

[+str] [+ant]

[+str] [+ant]

[+str] [-ant]

[+voi] [+str]
[+str] [-ant]

[+voi] [+str]
[+str] [-ant]
```

...
Two Ways To Learn

Top-Down Induction

- Start at the most specific points (highest) in the space
- Remove all the substructures that are present in the data.
- Collect the most general substructures remaining.

Bottom-Up Induction

- Beginning at the lowest element in the space,
- Check whether this structure is present in the input data.
- If so, move up, either to a point with an adjacent underspecified segment, or a feature extension of a current segment, and repeat.
# Learning Algorithm

<table>
<thead>
<tr>
<th>Bottom-Up Relational Learner</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Prunes Hypothesis space according to ordering relation</td>
</tr>
<tr>
<td>▶ Provably identifies correct constraints for sequential data</td>
</tr>
<tr>
<td>▶ Uses data sparsity to its advantage!</td>
</tr>
</tbody>
</table>
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

**Bottom-Up Relational Learner**

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
## Learning Algorithm

### Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
## Learning Algorithm

### Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

**Bottom-Up Relational Learner**

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Algorithm

Bottom-Up Relational Learner

- Prunes Hypothesis space according to ordering relation
- Provably identifies correct constraints for sequential data
- Uses data sparsity to its advantage!
Learning Guarantees (De Raedt 2008)

This learner is provably guaranteed to find the responsible constraints. With What measures?

Criteria specifying whether a given set of constraints is acceptable w.r.t. data.

We want constraints:

- whose largest forbidden substructure is of size $k$
- which cover the data, i.e. $D \subseteq L(G)$
- which are more specific than all the other constraints $G'$ that cover the data, so $L(G) \subseteq L(G')$
- which forbid structures $S$ that are substructures of structures $S'$ forbidden by other grammars $G'$ that satisfy (1,2,3)
  - For all $S' \in G'$, there exists $S \in G$ such that $S \sqsubseteq S'$. 
NLP Example

In many NLP applications, each text symbol is treated independently.

Alphabet = \{a, \ldots, z, A, \ldots, Z\} = 52 symbols

Forbidding maybe all capitals → Explosion!

If we use feature [capital], only 27! 26 letters + [capital]
Whither Statistics

How to Integrate?

- Structured Hypothesis Spaces allow for correct generalization
- Hayes and Wilson are right to have a generality relation in their MaxEnt Learner, but why not use this space?
- What is the efficiency tradeoff between statistics and structure?
- Is there a constraint learner which can allow this structure?
Conclusion

Today’s Results

- Learning is due to representations and structured hypothesis spaces
- Features structure the space into collections of ideals
- Grammars are collections of ideals and filters
- These entailments allow bottom-up inference to succeed
- There is rich structure in features that order hypotheses, which makes our lives easier!
Conclusion

Things To Do

▶ Determine the trade-off between data sparsity and time complexity. We hypothesize sparser data should yield faster generalization.
▶ Extend algorithm to learning phonological transformations.
▶ Extend feature extensions to other domains.
▶ Incorporate into Statistical Learning Mechanisms.
▶ How does Statistics help or hurt? and when?
Thanks!

Special thanks to Jim Rogers for immensely helpful discussions

This work was supported by NIH under grant #R01HD87133-01