Easy and Hard Constraint Ranking in OT

Jason Eisner
U. of Rochester

August 6, 2000 – SIGPHON - Luxembourg

Outline

- The Constraint Ranking problem
- Making fast ranking faster
- Extension: Considering all competitors
- How hard is OT generation?
- Making slow ranking slower

The Constraint Ranking Problem

finite positive data
m items

Constraint Ranker

\(<C_3, C_1, C_2, C_5, C_4>\)
or “fail”

- Find grammar consistent with data
  (or just determine whether one exists)
- How efficient can this be?
- Different from Gold learnability
- Proposed by Tesar & Smolensky

What Is Each Input Datum?

Possibilities from Tesar & Smolensky
- A pairwise ranking \( g > h \)
- An attested form \( g \)
- An attested set \( G \)
  - 1 grammatical element - learner doesn’t know which!
  - Captures uncertainty about the representation or underlying form of the speaker’s utterance
  - Today we’ll assume learner does know underlying

Key Results

- A pairwise ranking \( g > h \) linear time in \( n \)
- An attested form \( g \) coNP-hard even with \( m=1 \)
- An attested set \( G \) \( \Sigma_2 \)-complete
  - 1 grammatical element - learner doesn’t know which!
  - Captures uncertainty about the representation or underlying form of the speaker’s utterance
  - Today we’ll assume learner does know underlying

Outline

- The Constraint Ranking problem
- Making fast ranking faster
- Extension: Considering all competitors
- How hard is OT generation?
- Making slow ranking slower
Pairwise Rankings: $g > h$

<table>
<thead>
<tr>
<th>favor $h$</th>
<th>favor $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>$C_2$</td>
</tr>
<tr>
<td>$g$</td>
<td>*</td>
</tr>
<tr>
<td>$h$</td>
<td>**</td>
</tr>
</tbody>
</table>

Must eliminate $h$ before $C1$ or $C2$ makes it win
$C_4$ or $C_5 > C_1$
$C_4$ or $C_5 > C_2$
Satisfying these is necessary and sufficient

More Pairwise Rankings ...

evidence from more pairs

<table>
<thead>
<tr>
<th>$g &gt; h$</th>
<th>$g' &gt; h'$</th>
<th>$g'' &gt; h''$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_4$ or $C_5 &gt; C_1$</td>
<td>$C_2 &gt; C_1$</td>
<td></td>
</tr>
<tr>
<td>$C_4$ or $C_5 &gt; C_2$</td>
<td></td>
<td>$C_1$ or $C_3$ or $C_5 &gt; C_2$</td>
</tr>
<tr>
<td>$C_2 &gt; C_3$</td>
<td></td>
<td>$C_2 &gt; C_3$</td>
</tr>
<tr>
<td>$C_2 &gt; C_4$</td>
<td></td>
<td>$C_2 &gt; C_4$</td>
</tr>
</tbody>
</table>

We'll now use Recursive Constraint Demotion (RCD)
(Tesar & Smolensky - easy greedy algorithm)

Needn't be dominated by anyone
Recursive Constraint Demotion

<table>
<thead>
<tr>
<th>$g &gt; h$</th>
<th>$g' &gt; h'$</th>
<th>$g'' &gt; h''$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4 or C5 (\rightarrow) C1</td>
<td>C2 (\rightarrow) C1</td>
<td></td>
</tr>
<tr>
<td>C4 or C5 (\rightarrow) C2</td>
<td>C1 or C3 or C5 (\rightarrow) C2</td>
<td></td>
</tr>
<tr>
<td>C2 (\rightarrow) C3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2 (\rightarrow) C4</td>
<td>C1 or C3 or C5 (\rightarrow) C4</td>
<td></td>
</tr>
</tbody>
</table>

- How to find undominated constraint at each step?
- T&S simply search: \(O(mn)\) per search \(\Rightarrow O(mn^2)\)
- But we can do better:
  - Abstraction: Topological sort of a hypergraph
  - Ordinary topological sort is linear-time; same here!

The Constraint Ranking problem

Comparison: Constraint Demotion

- Tesar & Smolensky 1996
- Formerly same speed, but now RCD is faster
- Advantage: CD maintains a full ranking at all times
  - Can be run online (memoryless)
  - This eventually converges, but not a conservative strategy
    - Current grammar is often inconsistent with past data
    - To make it conservative:
      - On each new datum, rerank from scratch using all data (memorized)
      - Might as well use faster RCD for this
    - Modifying the previous ranking is no faster, in worst case

Outline

- The Constraint Ranking problem
- Making fast ranking faster
- Extension: Considering all competitors
- How hard is OT generation?
- Making slow ranking slower
New Problem

- Observed data: $g, g', \ldots$
- Must beat or tie all competitors
  - (Not enough to ensure $g > h, g' > h' \ldots$)
- Just use RCD?
  - Try to divide $g$'s competitors $h$ into equiv. classes
  - But can get exponentially many classes
  - Hence exponentially many blue nodes ☹

But Greedy Algorithm Still Works

- Preserves spirit of RCD
- Greedily extend grammar 1 constraint at a time
- No compilation into hypergraph
- But must run OT generation $mn^2$ times
  - To pick each of $n$ constraints, check $m$ forms under $n$ grammars
  - We'll see that this is hard ...
- T&S's solution also runs OT generation $mn^2$ times
  - Error-Driven Constraint Demotion
  - For $n^2$ CD passes, for $m$ forms, find (profile of) optimal competitor
  - That requires more info from generation - we’ll return to this!

Continuous Algorithms

- Simulated annealing
  - Boersma 1997: Gradual Learning Algorithm
  - Constraint ranking is stochastic, with real-valued bias & variance
- Maximum likelihood
  - Johnson 2000: Generalized Iterative Scaling (maxent)
  - Constraint weights instead of strict ranking
  - Deal with noise and free variation!
- How many iterations to convergence?

Outline

- The Constraint Ranking problem
- Making fast ranking faster
- Extension: Considering all competitors
- How hard is OT generation?
- Making slow ranking slower

Complexity Classes: Boolean

X-hard $\geq$ X-complete = hardest in X
Complexity Classes: Integer

- Integer-valued functions have classes too
  - **FP** (like P) Turing-machine polytime
  - **OptP** (like **NP** \( \exists \Psi(x) \)) \( \min (f) \)
  - **FPNP** (like **P** = \( \Delta_2 \))
- Note: **OptP**-complete \( \Rightarrow **FPNP**\)-complete
- Can ask Boolean questions about output of an **OptP**-complete function; often yields complete decision problems

OptP-complete Functions

- Traveling Salesperson
  - Minimum cost for touring a graph?
- Minimum Satisfying Assignment
  - Minimum bitstring \( b_1 b_2 \ldots b_n \) satisfying \( \phi(b_1, b_2, \ldots, b_n) \), a Boolean formula?
- Optimal violation profile in OT!
  - Given underlying form
  - Given grammar of bounded finite-state constraints
  - Clearly in **OptP**; \( \min f(x) \) where \( f \) computes violation profile
  - As hard as Minimum Satisfying Assignment

Hardness Proof

- Given formula \( \phi(b_1, b_2, \ldots, b_n) \)
- Need minimum satisfier \( b_1 b_2 \ldots b_n \) (or 11..1 if unsat)
- Reduce to finding minimum violation profile
- Let OT candidates be bitstrings \( b_1 b_2 \ldots b_n \)
- Let constraint \( C(\phi) \) be satisfied if \( \phi(b_1, b_2, \ldots, b_n) \)

<table>
<thead>
<tr>
<th>( C(\phi) )</th>
<th>( C(-b_1) )</th>
<th>( C(-b_2) )</th>
<th>( C(-b_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>only</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>satisfies</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>survive</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>past here</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Subtlety in the Proof

- Turning \( \phi \) into a DFA for \( C(\phi) \) might blow it up exponentially - so not poly reduction!
- Luckily, we're allowed to assume \( \phi \) is in CNF:
  \[ \phi = D_1 \land D_2 \land \ldots \land D_m \]
  \[ C(D_1) \]
  \[ C(D_2) \]
  \[ C(D_3) \]
  \[ \ldots \]
  \[ C(D_m) \]
  \[ \phi \]

<table>
<thead>
<tr>
<th>( C(D_1) )</th>
<th>( C(D_2) )</th>
<th>( C(D_3) )</th>
<th>( C(D_m) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>equivalent to ( C(\phi) ):</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>only satisfies</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>survive past here</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Another Subtlety

- Must ensure that if there is no satisfying assignment, 11..1 wins
- Modify each \( C(D_i) \) so that 11..1 satisfies it
- At worst, this doubles the size of the DFA

<table>
<thead>
<tr>
<th>( C(D_i) )</th>
<th>( C(-b_1) )</th>
<th>( C(-b_2) )</th>
<th>( C(-b_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>equivalent to ( C(\phi) ):</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>only satisfies</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>survive past here</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Associated Decision Problems

- OptVal
  - **FPNP**-complete
- OptVal < \( k \)
  - **NP**-complete
- OptVal = \( k \)
  - \( \Delta_2 \)-complete
- Last bit of OptVal?
  - \( \Delta_2 \)-complete
- Is \( g \) optimal?
  - **coNP**-complete
- Is some \( g \in G \) optimal?
  - \( \Delta_2 \)-complete

**EDCD**

**RCD** (mult. competitors)
Is some \( g \in G \) optimal?

- Problem is in \( \Delta_2 = \text{P}^{\text{NP}} \).
- OptVal < \( k \) is in \( \text{NP} \).
- So binary search for OptVal via \( \text{NP} \) oracle.
- Then ask oracle: \( \exists g \in G \) with profile OptVal?

Completeness:
- Given \( \phi \), we built grammar making the MSA optimal.
- \( \Delta_2 \)-complete problem: Is final bit of MSA zero?
- Reduction: Is some \( g \) in \( \{0,1\}^{10} \) optimal?
- Notice that \( \{0,1\}^{10} \) is a natural attested set.

Outline

- The Constraint Ranking problem
- Making fast ranking faster
- Extension: Considering all competitors
- How hard is OT generation?
- Making slow ranking slower

Ranking With Attested Forms

- Complexity of ranking?
- If restricted to 1 form: \( \text{coNP} \)-complete
  - no worse than checking correctness of ranking!
- General lower bound: \( \text{coNP} \)-hard
- General upper bound: \( \Delta_2 = \text{P}^{\text{NP}} \)
  - because RCD solves with \( O(mn^2) \) many checks

Ranking With Attested Sets

- Problem is in \( \Sigma_2 \) \( \exists \forall \Psi(x,y) \)
  - \( \exists \) \( (\text{ranking}, g \in G) \) \( \forall : g > h \)
  - In fact \( \Sigma_2 \)-complete!
  - Proof by reduction from QSAT \( \Sigma_2 \)
    - \( \exists b_1, \ldots, b_r \forall x_1, \ldots, x_s \phi(b_1, \ldots, b_r, c_1, \ldots, c_r) \)
  - Few natural problems in this category
    - Some learning problems that get positive and negative evidence
    - OT only has implicit negative evidence: no other form can do better than the attested form

Conclusions

- Easy ranking easier than known
- Hard ranking harder than known
- Adding bits of realism quickly drives complexity of ranking through the roof
- Optimization adds a quantifier:

<table>
<thead>
<tr>
<th>generation</th>
<th>ranking</th>
<th>w/ uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>derivational</td>
<td>FP</td>
<td>( \text{NP-complete} )</td>
</tr>
<tr>
<td>OT</td>
<td>( \text{OptP-complete} )</td>
<td>( \text{coNP-hard, in } \Delta_2 )</td>
</tr>
<tr>
<td></td>
<td>( \Sigma_2 )-complete</td>
<td>( \Sigma_2 )-complete</td>
</tr>
</tbody>
</table>

Open Questions

- Rescue OT by restricting something?
- Effect of relaxing restrictions?
- Unbounded violations
- Non-finite-state constraints
- Non-poly-bounded candidates
- Uncertainty about underlying form
- Parameterized analysis (Wareham 1998)
- Should exploit structure of Con
  - huge (linear time is too long!) but universal