Using Program Synthesis for Social Recommendations

Alvin Cheung
Armando Solar-Lezama
Samuel Madden

MIT
Learning example

Given labeled data:

<table>
<thead>
<tr>
<th>User</th>
<th>Location</th>
<th>Time</th>
<th>Interested?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Office</td>
<td>10am</td>
<td>N</td>
</tr>
<tr>
<td>Bill</td>
<td>Home</td>
<td>3pm</td>
<td>N</td>
</tr>
<tr>
<td>Joe</td>
<td>Office</td>
<td>11pm</td>
<td>Y</td>
</tr>
<tr>
<td>Joe</td>
<td>Bar</td>
<td>6am</td>
<td>Y</td>
</tr>
</tbody>
</table>

Possible classifier:
(User = Joe) and
(location = Office or location = Bar) and
(time < 7 am or time > 10pm)
Challenge 1: Mobile Interface

Initial database is large and queries are sparse

- A small sample of the data is unlikely to have many of the records you want
- Standard PBE interface not suitable for mobile
Challenge 2: Decomposable classifiers

(User = Joe) and (location = Office or location = Bar) and (time < 7 am or time > 10pm) vs. 0.25x_1 + 0.65x_2 > 0

Want to avoid black-box classifiers

– Data is aggregated from multiple sources
– Want to push filters to the source
Applying synthesis to LifeJoin

<table>
<thead>
<tr>
<th>Input-output spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interest([Peter, jog, Charles, 5PM]) = Y</td>
</tr>
<tr>
<td>Interest([Mary, office, 9AM]) = N</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Search space grammar

interest(e) { act(e) | loc(e) |
act(e) & loc(e) | ...}

act(e) ::= e.user = u |
e.activity = a |
e.time = t |
act(e) & act(e) 
...

Program Synthesizer

interest(e) { e.user = Peter and e.activity = jog }
interest(e) { e.user ≠ Mary and e.time > 4PM }

Shortcomings:

Generalization guarantees

Active learning
Our new hybrid approach

- Labeled data
- Interest grammar

Program Synthesizer

Interest functions

user = Peter and activity = jog

user ≠ Mary and time > 4PM
Our new hybrid approach

- Labeled data
- Interest grammar
- Program Synthesizer
- SVM features
  - user = Peter
  - activity = jog
  - user ≠ Mary
  - time > 4PM
- SVM model
  - SVs
  - Interest grammar
- Program Synthesizer

Symbolic encoding of search space

Generalization guarantees

Active learning

Decomposable model

user = Peter
and
time > 4PM
Experimental results

Implemented different feature selection algorithms and classifiers

<table>
<thead>
<tr>
<th>Name</th>
<th>Feature selection</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>None</td>
<td>Linear SVM</td>
</tr>
<tr>
<td>Poly</td>
<td>Unary features only</td>
<td>Poly. kernel SVM</td>
</tr>
<tr>
<td>L1</td>
<td>LASSO</td>
<td>Linear SVM</td>
</tr>
<tr>
<td>MI</td>
<td>Mutual information</td>
<td>Linear SVM</td>
</tr>
<tr>
<td>Tree</td>
<td>C4.5 on unary features</td>
<td>Linear SVM</td>
</tr>
<tr>
<td>Hybrid</td>
<td>Features extracted from 10 synthesized functions</td>
<td>Linear SVM</td>
</tr>
</tbody>
</table>
Experimental results: active learning

Hybrid approach has a much faster learning rate
Conclusion

Programming by example is important for mobile

Synthesis and ML can interact in fruitful ways