Using Program Synthesis for Social Recommendations

Alvin Cheung
Armando Solar-Lezama
Samuel Madden

MIT
Building social networking apps using phone data

• Mobile phones now come with various sensors
  – Accelerometer
  – GPS
  – Proximity

• Data collected from sensors can be used to automatically generate events
  – Enables new social networking apps
LifeJoin: Automatic generation of interesting events

Learned model

Inferred events

Event Ratings
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)
Applying machine learning to LifeJoin

- Does not create decomposable models

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

- Need to encode inputs into feature space representation
  - Create substantial time and space overhead

- Events are highly personalized
  - Hard to leverage labeled data from others
Intro to program synthesis

Input-output spec
\[ f(5) = 12 \quad f(10) = 22 \]

Search space grammar
\[
\begin{align*}
  f(x) \{ \text{return} \ expr(x); \} \\
  expr(x) ::= n \\
  \quad | \quad x \\
  \quad | \quad expr(x) + expr(x)
\end{align*}
\]

Program Synthesizer
\[
  f(x) \{ \text{return} \ x + x + 2; \}
\]

Forms hypothesis
Test on spec

Symbolic encoding of search space
Applying synthesis to LifeJoin

Input-output spec

\[
\text{Interest([Peter, jog, Charles, 5PM])} = \text{Y} \\
\text{Interest([Mary, office, 9AM])} = \text{N} \\
\ldots
\]

Search space grammar

\[
\text{interest}(e) \{ \text{act}(e) | \text{loc}(e) | \text{act}(e) \& \text{loc}(e) | \ldots \}
\]

\[
\text{act}(e) ::= \text{e.user} = u \\
| \text{e.activity} = a \\
| \text{e.time} = t \\
| \text{act}(e) \& \text{act}(e) \\
\ldots
\]

Program Synthesizer

\[
\text{interest}(e) \\
\{ \text{e.user} = \text{Peter and e.activity} = \text{jog} \}
\]

\[
\text{interest}(e) \\
\{ \text{e.user} \neq \text{Mary and e.time} > 4PM \}
\]

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

Program Synthesizer

Interests grammar

Symbolic encoding of search space

Generalization guarantees

Active learning

SVM features

user = Peter
activity = jog
user ≠ Mary
time > 4PM

SVM model

SVs

Interests grammar

Decomposable model

Program Synthesizer

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
Learning user interests in LifeJoin

• Learning user interests from phone data poses new challenges

• Combine machine learning algorithms and programming synthesis techniques to solve the learning problem

http://people.csail.mit.edu/akcheung