Hacking Desire

“Reverse-engineering what people want”

Ian Clarke, CEO Uprizer Labs

ian@uprizer.com
Why?

• Everyone has needs and desires
• If we can predict these, then we can give people what they want
• This will make them happy
• ???
• Profit!
Specific problems we could solve

- Music (last.fm, Indy, Pandora)
- Movies (Netflix)
- Advertising ("behavioral advertising")
- Dating
**Existing approach: Item based CF**

- “People who liked X also liked these”

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple to implement</td>
<td>Naive - relies on a single piece of information about the user</td>
</tr>
<tr>
<td>Easy for end-users to understand</td>
<td>Limited diversity in recommendations</td>
</tr>
</tbody>
</table>
### Existing approach: User based CF

- “People like you liked these”

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can develop quite nuanced view of a user’s preferences</td>
<td>Requires a lot of data per-user to accurately determine similarity</td>
</tr>
<tr>
<td>Easy for end-users to understand</td>
<td>Can be hard to scale - naive implementation is $O(N^2)$</td>
</tr>
</tbody>
</table>
Representing user preference

Ian:

Robocop:

Action  | Violence  | Sci-Fi    | Romance
-------|-----------|-----------|--------
11.25   | 3.75      | -3.75     | -11.25
10.0    | 5.0       | 0         | -5.0
-10.0   | -5.0      | 0         | 0

Ian: Robocop:
Computing user preference
Computing user preference

- User preferences are $A \ B \ C \ D$
Computing user preference

- User preferences are $A B C D$
- Item features are $e f g h$
Computing user preference

• User preferences are $A \ B \ C \ D$
• Item features are $e \ f \ g \ h$
• Rating = $Ae + Bf + Cg + Dh$
Computing user preference

- User preferences are $A$, $B$, $C$, $D$
- Item features are $e$, $f$, $g$, $h$
- Rating = $Ae + Bf + Cg + Dh$
- But...
Computing user preference

- User preferences are $A B C D$
- Item features are $e f g h$
- Rating $= Ae + Bf + Cg + Dh$
- But...
- How do we determine values for $A, B, C, D, e, f, g, and h$?
Optimization through Gradient Descent

- Find the optimal by gradually moving towards it
- Similar to a ball rolling down a hill
- Be careful of local minima
Choosing features
Choosing features

• How do we decide what the important features of something are?
Choosing features

• How do we decide what the important features of something are?

• We don’t need to! We let the gradient descent algorithm figure it out for us!
Choosing features

• How do we decide what the important features of something are?

• We don’t need to! We let the gradient descent algorithm figure it out for us!

• We let the algorithm determine what features make sense for accurate predictions
Choosing features

• How do we decide what the important features of something are?

• We don’t need to! We let the gradient descent algorithm figure it out for us!

• We let the algorithm determine what features make sense for accurate predictions

• They may correspond to qualities we have names for, or they may not
Does it work?

• Netflix Prize has become de-facto standard for testing collaborative filters
• Half a million users
• 20,000 movies
• 100 million ratings
Root Mean Squared Error

- Netflix Prize measures prediction accuracy
- Mean difference between what was predicted and what user actually did
- Square the differences, and take root of their mean
- Has effect of punishing very bad predictions more than simple mean would
- Uses an unseen “probe set” so that algorithm can’t just memorize data
Our algorithm’s performance

• We score 0.905 on Netflix probe set
• This is about 5% lower and therefore better than Netflix’ own algorithm
• But, some algorithms get down as low as 0.864
• How do they do this?
• Can we beat them?
• Do we want to?
Flaws in RMSE metric

• In most CF applications:
  • Predictions only matter *relative* to each other
  • Accuracy of high predictions is much more important than low predictions

• RMSE accounts for neither of these facts

• So: A better RMSE doesn’t necessarily translate into better real-world performance