NOTES ON COLLABORATIVE FILTERING

1. Intro. Collaborative Filtering

- based on Ch. 9 in Rajaraman and Ullman: *Mining Massive Data Sets.*
- CF – major alternative to content-based approach to RS.
- more popular, because ...
- assumes no meta-data or “features” on items or users available; nor requires any features to be easily extractable.
- Idea: Treat the column of each item as a representation of the item; how is it liked by various users.
- Similarly, the profile of each user is just his/her rating pattern on the items s/he has rated.
- Two approaches, depending on whether we want to work with users or with items: user-user or item-item.
- Orthogonally, two approaches, depending on whether we use the ratings matrix as is, to make our predictions, or want to build models of user behavior – memory-based vs. model-based.
- In all cases, need notion of user-user similarity or item-item similarity.
- cosine distance or Jaccard distance are two possibilities.

2. Some Challenges in Measuring Similarity

- Consider the following (toy) example utility (sometimes called ratings) matrix $M$.

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>3</td>
<td></td>
<td>4</td>
<td>5</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Intuitive observations: A and C seem quite dissimilar, in fact less similar than A and B (be warned – limited and sparse data, so inferences somewhat risky!).

- Let’s try out four possible similarity measures: Jaccard (based on common movies rated, and ignoring ratings), cosine on raw ratings, Jaccard and cosine on the so-called rounded ratings (from the book; not common in research literature on CF), and cosine on normalized ratings obtained by subtracting each user’s mean rating from their raw ratings (thus effectively turning some ratings negative).

- Jaccard on the raw ratings matrix gives $\text{sim}(A, B) = 1/5 < 2/4 = \text{sim}(A, C)$. Unsurprisingly, it seems intuitively wrong. Cosine gives $\text{sim}(A, B) = 4 \times 5/|A| \cdot |B| \approx 0.380$ and $\text{sim}(A, C) = (5 \times 2 + 1 \times 4)/|A| \cdot |C| \approx 0.322$. Seems OK since it says A and B are more similar than A and C, but in general working with raw ratings ignores different rating scales people work with; easy graders vs. hard graders. Ignoring this can lead to low quality predictions.
• “Rounded” ratings: 3, 4, 5 ↦ 1 and 1, 2 ↦ 0 (unrated). Verify that both Jaccard and cosine give the right conclusions on these rounded ratings. But what is the basis for such rounding? The next approach, based on “normalized” ratings, is more principled and is also used by Pearson’s correlation coefficient.

• The normalized ratings matrix is as follows:

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2/3</td>
<td>5/3</td>
<td>-7/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1/3</td>
<td>1/3</td>
<td>-2/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>-5/3</td>
<td>1/3</td>
<td>4/3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

- Verify: cosine similarity of A, B ≈ 0.092. and cosine similarity of A and C ≈ -0.559.

- The last one seems to best match our intuition: notice, cosine sim. ranges in [-1, 1] with -1 indicating complete dissimilarity and 1 indicating identity.

3. How Do We Predict Ratings/Utility?

• user-user approach: One approach: for the distinguished user u, let N(u) denote the top-k most similar users (according to one of the notions of similarity) who have rated item i. Let δ(v, j) be the deviation of v’s raw rating of j from v’s mean rating, i.e., δ(v, j) = M_{v,j} - \overline{v}, where M_{v,j} is the raw rating and \overline{v} is the mean rating of v. Then estimate u’s (unknown) rating on i as

$$M_{u,i} = \frac{1}{k} \sum_{v \in N(u)} \delta(v, i) + \overline{v}.$$ 

- This approach “corrects” for u being a high (or low) rater or most of the users in N(u) being high (or low) raters. The “deviation from the mean” treats the mean as neutral and positive (resp., negative) deviation as a positive (resp., negative) rating of the user in question.

- In principle, there is duality between items and users and hence the above approach works for item-item; but item-item has been found to be better in practice.

• One possible reason: items tend to have several single-valued attributes (e.g., main genre of songs, year, artist, etc). OTOH, users’ liking for items (songs) can be quite varied (classical, jazz, as well as rock).

- Item-item takes more work – why?

• Regardless of approach, need to predict ratings of all unrated (or at least unrated but high predicted ratings) items for a distinguished user. ⇒ corresponding top-N problem. That is, predict the top-N items with highest predicted ratings for user u and then make recs.

- Often top-N items for each user – precomputed; this assume slow evolution of the ratings matrix; what about updating the top-N lists? In fact, often similarity matrix is precomputed and updating it against updates to the ratings matrix is important. What about the chances of doing any of this incrementally?

4. Enhancing Our Reach by Clustering

• The sparsity of the ratings matrix hits us quite soon.
• user-user: what if \( u, v \) hypothetically had similar tastes (e.g., both like jazz and both like 60’s rock), but have not rated/bought \( m \)any items (which are instances of their similar tastes) in common?
• item-item: what if \( i, j \) belonged to the same category (both are jazz) but few users rated them both?
• A possible solution – clustering.
• Use any notion of say, item-item similarity and cluster items. Initially try out a large number of clusters. Hierarchical clustering gives good control for such things. This turns the ratings matrix into a Users \( \times \) Item Clusters matrix; entries = cluster averages for that user; no hits in a cluster \( \Rightarrow \) entry = 0.
• E.g., if clustering put all HP movies in one cluster, all SW movies in another and left TW alone, what would the resultant matrix look like?
• Now, use user profiles = vectors from this matrix. Can use any notion of user-user similarity and cluster users. Similar comments to above. Resulting matrix = User clusters \( \times \) Item clusters matrix. Entries calculated by averaging as above.
• Revert to clustering (clusters of) items and clustering (clusters of) users alternately. Stop when desired number of item clusters and user clusters are found.
• For user \( u \) and item \( i \) predict rating as follows:
  – if \( M_{Cl(u),Cl(i)} \) is defined it, use it!
  – otherwise, use method discussed in previous section, but on the clusters \( \times \) clusters matrix to predict it and use it.

5. The “PQ-Decomposition” Approach
• Pretend the characteristics of items are captured by \( k \) features for some small \( k \).
• Ditto for users. So, the \( k \) features are joint features between users and items.
• This leads to factoring \( M \) into factor matrices \( P \) and \( Q \). So, \( M_{n \times m} \approx P_{n \times k} \times Q_{k \times m} \). That is, \( P \) and \( Q \) should approximate \( M \) as closely as possible on all the known ratings, i.e., non-blank entries.
• “As closely as possible” can be measured in different ways. RMSE is one popular measure. For given \( P, Q \), RMSE is computed as follows.
  – Let \( N \) be the number of non-blank entries of \( M \).
  – let \( S = \sum_{M_{i,j} \in \text{nonblank}(M)} (M_{i,j} - P_{i,*} \cdot Q_{*,j})^2 \).
  – Then \( RMSE = S/N \).
• How do we find \( P \) and \( Q \) that minimize RMSE? In general, the problem has multiple local minima. It’s a hard problem. Here is one approach, inspired by gradient descent.
  – Initialize \( P \) and \( Q \) to something: e.g., all 1’s.
  – Compute \( PQ \).
  – Pick an arbitrary “new” entry of \( P \), say \( P_{i,j} \); call its new “optimal” value \( x \).
-- Express RMSE as a function of $x$ (notice everything else is a constant in RMSE). Differentiate and equate to zero, to find the optimal value of $x$ that will bring out greatest reduction in RMSE, when confining attention to making changes to $P_{i,j}$ only.
-- Repeat until ??

• Questions raised by the above approach:
  -- Should we preprocess $M$? In what way?
  -- How to initialize $P, Q$?
  -- Can we try with multiple initializations and pick the best?
  -- In what order should we try to tune the elements of $P, Q$?
  -- When should we stop?

• Preprocessing $M$: What’s the concern we’re trying to address? Differences in rating scales and differences in inherent quality of items could skew the predictions; better to “normalize” them away. Some options:
  -- Subtract from each entry $M_{i,j}$ the average rating of user $i$; from resulting matrix, from each entry subtract average rating of corresponding item.
  -- The other way around.
  -- Subtract from $M_{i,j} 1/2$(avg rating of user $i$ + avg rating of item $j$).

• Once predictions are obtained using $P, Q$, need to add what was subtracted before.

• Initializing $P$ and $Q$: One possibility – set all elements to $\sqrt{a/k}$, where $a = 1/N \sum_{M_{i,j} \in \text{nonblank}(M)} M_{i,j}$. What happens when we do this?

• Since there are multiple local minima, randomization may help. That is, try the above initialization multiple times, each time with a different perturbation to elements of $P$ and $Q$ (perturbation picked from different distributions – e.g., normal or uniform or Poisson). Pick the best result.

• Order of visiting elements of $P$ and $Q$: Most straightforward approach: visit elements of $P$ in row-major order followed a similar order for elements of $Q$. Repeat the process until no element can be improved.

• A slightly more sophisticated approach: In each iteration, follow different permutation of elements of $P$ and $Q$ for visiting (and tuning) them.

• In each iteration, pick elements to visit at random.

• When to Stop?: Will RMSE become zero? A few possibilities again.

• Add up the improvement resulting from a round. Stop when it falls below a chosen threshold.

• Instead of sum, we can take the max.

6. Matrix Factorization

• Sophisticated version of $PQ$ decomposition, with more bells and whistles added.

• Problem – same as before: Given utility/ratings matrix $M$, you want to find $k$ latent features and matrices $P, Q$, such that $M \approx P_{n \times k} \times Q_{k \times m}$.

• Want to minimize the (squared) error: $\sum_{(u,v) \in M} (M_{u,v} - P_{u,*} \cdot Q_{*,v})^2$. 
• Recall, minimizing raw error doesn’t make sense as error terms can be positive or negative. And minimizing RMSE amounts to minimizing the squared error.

• The minimization is normally done over the “training” subset of the known entries in $M$. Results evaluated on the “test” (or “holdout”) subset of the known entries in $M$.

• So, $P$ describes associations between users and latent features: how much a user likes items scoring high on each feature.

• $Q$ describes associations between items and features: how much does an item score on each feature.

• Dot product of $P_{u,*}$ and $Q_{*,v}$ says how much user $u$ is likely to like item $v$.

• The predicted rating is $\hat{M}_{u,v} = \sum_{f=1}^{k} P_{u,f} Q_{f,v}$. Problem is to find $P, Q$ that minimizes the sum of squared errors. The idea is similar to that of $PQ$ decomposition.

• Initialize $P, Q$ in some way. All considerations for initialization that we discussed before apply here, for the initialization: arbitrary, perturbation, pick the best from several trials, etc.

• $e_{u,v}^2 = (M_{u,v} - \hat{M}_{u,v})^2 = (M_{u,v} - \sum_{f=1}^{k} P_{u,f} Q_{f,v})^2$.

• As discussed before, there are multiple local minima. We can follow the gradient descent method. We hinted at this when discussing the $PQ$ decom. approach. Keep in mind you don’t want to make bold strides in the direction of steepest descent or you risk oscillating around the minimum.

• $\partial e_{u,v}^2 / \partial P_{u,f} = -2e_{u,v} Q_{f,v}$.

• $\partial e_{u,v}^2 / \partial Q_{f,v} = -2e_{u,v} P_{u,f}$.

• Here is how we can update $P_{u,f}$ and $Q_{f,v}$ based on gradient descent:
  - Update $P_{u,f}$ by adding the partial derivative (in the opposite direction), but tempered by a small parameter, say $\gamma << 1$. Typically, pick $\gamma = 0.0001$. $\Rightarrow P_{u,f} \leftarrow P_{u,f} + 2\gamma e_{u,v} Q_{f,v}$.
  - Similarly, $Q_{f,v} \leftarrow Q_{f,v} + 2\gamma e_{u,v} P_{u,f}$.
  - Stop iterating when the aggregate of the error (sum, max, etc) is below a set threshold.

6.1. Enhancements.

• many enhancements have been designed: e.g., adding bias, alternating least squares, regularization, etc. We will discuss regularization and leave the rest for reading assignment.

• The basic MF algorithm above runs the risk of overfitting. Regularization can be used to overcome this.

• Redefine the “error” as:

$$e_{u,v}^2 = (M_{u,v} - \sum_{f=1}^{k} P_{u,f} Q_{f,v})^2 + \lambda(||P_{u,*}||^2 + ||Q_{*,v}||^2).$$

• What’s the rationale?
  - When you minimize the new “error” squared, you don’t let components of the vectors $P_{u,*}$ and $Q_{*,v}$ grow too big. Notice these are the only vectors from $P, Q$ that determine $M_{u,v}$.
  - Finding entries of $P$ and $Q$ that minimize the new error function gives us matrices $P$ and $Q$ whose entries are not too large while approximating $M$.

• Value of $\lambda$ is chosen empirically, $\lambda = 0.05$ being typical.
• We then get the following modified update statements:
  \[ P_{u,f} \leftarrow P_{u,f} + 2\gamma(e_{u,v}Q_{f,v} - \lambda P_{u,f}). \]
  \[ Q_{f,v} \leftarrow Q_{f,v} + 2\gamma(e_{u,v}P_{u,f} - \lambda Q_{f,v}). \]

References