FAQs

- Term project
  - 5:00PM March 29, 2018
- PA2
  - Recitation: Friday
- No office hour on Friday (10-11AM)

Today's topics

- Recommendation Systems
  - Collaborative Filtering
  - Amazon's Item-to-Item CF
- Evaluation/Validation Techniques

Collaborative filtering

- Focus on the similarity of the user ratings for items
- Users are similar if their vectors are close according to some distance measure
  - E.g. Jaccard or cosine distance
- Collaborative filtering
  - The process of identifying similar users and recommending what similar users like
Measuring similarity

- How to measure similarity of users or items from their rows or columns in the utility matrix?
  - Jaccard Similarity for A and B: 1/5
  - Jaccard Similarity for A and C: 2/4
- For user A, user C might have similar opinion than user B
- Can user C provide a prediction for A?

<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>8</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Cosine similarity (1/2)

- We can treat blanks as 0 values
- The cosine of the angle between A and B is
  \[
  \cos = \frac{4 \times 5}{\sqrt{4^2 + 5^2} + \sqrt{4^2 + 5^2}} = 0.380
  \]

<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>8</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Cosine similarity (2/2)

- A is slightly closer to B than C

Normalizing ratings (1/2)

- What if we normalize ratings by subtracting from each rating the average rating of that user?
- Some rating (very low) will turn into negative numbers
- If we take the cosine distance, the opposite views of the movies will have vectors in almost opposite directions
  - It can be as far apart as possible

Normalizing ratings (2/2)

- The cosine of the angle between A and B
  \[
  \cos = \frac{(2/3) \times (2/3)}{\sqrt{(2/3)^2 + (2/3)^2 + (2/3)^2}} = 0.092
  \]

<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>2/3</td>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>B</td>
<td>2/3</td>
<td>2/3</td>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>C</td>
<td>2/3</td>
<td>2/3</td>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Computational complexity

• Worst case
  • \( O(MN) \) where \( M \) is the number of customers and \( N \) is the number of product catalog items
  • It examines \( M \) customers and up to \( N \) items for each customer

Computational complexity (1/3)

• The average customer vector is extremely sparse!
  • The algorithm’s performance tends to be closer to \( O(M+N) \)
  • Scanning every customer
    • \( O(M) \) not \( O(MN) \)
    • Almost every customer has very small \( N \)
  • Few customers who have purchased or rated a significant percentage of items
    • Requires \( O(N) \)
    • 10 million customers and 1 million items?

Computational complexity (2/3)

• We can reduce \( M \) by:
  • Randomly sampling the customers
  • Discarding customers with few purchases

• We can reduce \( N \) by:
  • Discarding very popular or unpopular items
  • Partitioning the item space based on the product category or subject classification

Computational complexity (3/3)

• Dimensionality reduction techniques can reduce \( M \) or \( N \) by a large factor
  • Clustering
  • Principal component analysis

Disadvantage of space reduction

• Reduced recommendation quality
  • Sampled customer
  • More similar customers will be dropped
  • Item-space partitioning
    • It will restrict recommendations to a specific product or subject area
  • Discarding most popular or unpopular items
    • They will never appear as recommendations

Large Scale Data Analytics
4. Recommendation Systems
Amazon.com: Item-to-item collaborative filtering
This material is built based on,

- Greg Linden, Brent Smith, and Jeremy York, "Amazon.com Recommendations, Item-to-Item Collaborative Filtering" IEEE Internet Computing, 2003

Amazon.com uses recommendations as a targeted marketing tool
- Email campaigns
- Most of their web pages

Item-to-item collaborative filtering
- It does NOT match the user to similar customers
- Item-to-item collaborative filtering
  - Matches each of the user’s purchased and rated items to similar items
  - Combines those similar items into a recommendation list

Determining the most-similar match
- The algorithm builds a similar-items table
  - By finding items that customers tend to purchase together
- How about building a product-to-product matrix by iterating through all item pairs and computing a similarity metric for each pair?
  - Many product pairs have no common customer
    - If you already bought a TV today, will you buy another TV again today?

Co-occurrence in a product-to-product matrix
- Suppose that there are 4 users A, B, C, and E with 5 products P1, P2, P3, P4, and P5
  - A purchased P1, P2, P3
  - B purchased P2, P3, P5
  - C purchased P3, P4
  - D purchased P2, P4
Similarity product-to-product matrix

- Similarity between products
  - P1: T-shirt
  - P2: video cable
  - P3: projector
  - P4: tv
  - P5: smart tv

- Assume that,
  - Similarity (P1, P2) = 0.001
  - Similarity (P1, P3) = 0.001
  - Similarity (P1, P4) = 0.001
  - Similarity (P1, P5) = 0.001
  - Similarity (P2, P3) = 0.013
  - Similarity (P2, P4) = 0.022
  - Similarity (P2, P5) = 0.022
  - Similarity (P3, P4) = 0.26
  - Similarity (P3, P5) = 0.26
  - Similarity (P4, P5) = 0.72

Calculating the similarity between a single product and all related products:

- For each item in product catalog, I1
  - For each customer C who purchased I1
    - For each item I2 purchased by customer C
      - Record that a customer purchased I1 and I2
    - For each item I2
      - Compute the similarity between I1 and I2

Computing similarity

- Using cosine measure
  - Each vector corresponds to an item rather than a customer
  - M dimensions correspond to customers who have purchased that item

Creating a similarity product-to-product table

- Similar-items table is extremely computing intensive
  - Offline computation
    - \(O(N^2M)\) in the worst case
      - Where \(N\) is the number of items and \(M\) is the number of users
    - Average case is closer to \(O(NM)\)
      - Most customers have very few purchases
    - Sampling customers who purchase best-selling titles reduces runtime even more
      - With little reduction in quality

Scalability (1/2)

- Amazon.com has around 300 million customers and more than 562,382,292 cataloged items
- Traditional collaborative filtering does little or no offline computation
- Online computation scales with the number of customers and catalog items.

Scalability (2/2)

- Cluster models can perform much of the computation offline
  - Recommendation quality is relatively poor
- Content-based model
  - It cannot provide recommendations with interesting, targeted titles
  - Not scalable for customers with numerous purchases and ratings
Key scalability strategy for Amazon recommendations

• Creating the expensive similar-items table offline

• Online component
  • Looking up similar items for the user’s purchases and ratings
  • Scales independently of the catalog size or the total number of customers

• It is dependent only on how many titles the user has purchased or rated

Recommendation quality

• The algorithm recommends highly correlated similar items
  • Recommendation quality is excellent
  • Algorithm performs well with limited user data

Why evaluation/validation?

5. Evaluation and Validation

Large Scale Data Analytics

Why evaluation/validation? (1/2)

• Process for model selection and performance estimation

• Model selection (fitting the model)
  • Most of the models have one or more free parameters
  \[ h(x) = \theta_0 + \theta_1 x_1 \]

• How do we select the “optimal” parameter(s) or model for a given classification problem/predictive analytics?

Why evaluation/validation? (2/2)

• Performance estimation
  • Once we have chosen a model, how do we estimate its performance (accuracy)?

• Performance is typically measured by the true error rate
  • e.g. the classifier’s error rate on the entire population

Challenges (1/2)

• If we had access to an unlimited number of examples these questions have a straightforward answer
  • Choose the model that provides the lowest error rate on the entire population
    • Of course, that error rate is the true error rate

• In real applications we only have access to a subset of examples, usually smaller than we wanted
  • What if we use the entire available data to fit our model and estimate the error rate?
    • The final model will normally overfit the training data
    • We already used the test dataset to train the data
Challenges (2/2)

- This problem is more pronounced with models that have a large number of parameters
  - The error rate estimate will be overly optimistic (lower than the true error rate)
    - In fact, it is not uncommon to have 100% correct classification on training data
- A much better approach is to split the training data into disjoint subsets: the holdout method

The Holdout Method

- Split dataset into two groups
  - Training set
    - Used to train the model
  - Test set
    - Used to estimate the error rate of the trained model
  - Simplest cross validation method

Drawbacks of the holdout method

- Drawbacks
  - For a sparse dataset, we may not be able to set aside a portion of the dataset for testing
  - Based on the where “split” happens, the estimate of error can be misleading
    - Sample might not be representative
- The limitations of the holdout can be overcome with a family of resampling methods
  - More computational expense
  - Stratified sampling
  - Cross Validation
    - Random subsampling
    - K-fold cross validation
    - Leave-one-out Cross Validation

Random Subsampling

- K data splits of the dataset
  - Each split randomly selects a fixed number of examples without replacement
  - For each data split, retain the classifier from scratch with the training examples and estimate $E_i$ with the test examples

<table>
<thead>
<tr>
<th>Experiment 1</th>
<th>Experiment 2</th>
<th>Experiment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
True Error Estimate

• The true error estimate is obtained as the average of the separate estimates $E_i$.

• This estimate is significantly better than the holdout estimate

$$ E = \frac{1}{K} \sum_{i=1}^{K} E_i $$

$k$-Fold Cross-validation

• Create a $k$-fold partition of the dataset.
  • For each of the $k$ experiments use $K-1$ folds for training.
    • The remaining one for testing.

True error estimate

• $k$-fold cross validation is similar to random subsampling.

• The advantage of $k$-Fold Cross validation
  • All the examples in the dataset are eventually used for both training and testing.
  • The true error is estimated as the average error rate

$$ E = \frac{1}{K} \sum_{i=1}^{K} E_i $$

Leave-one-out Cross Validation

• Leave-one-out is the degenerate case of $k$-Fold Cross validation.

• $k$ is chosen as the total number of examples.

• For a dataset with $N$ examples, perform $N$ experiments.

• Use $N-1$ examples for training, the remaining example for testing.

$$ E = \frac{1}{N} \sum_{i=1}^{N} E_i $$
How many folds are needed?

• The choice of the number of folds depends on the size of the dataset
  • For large datasets, even 3-Fold Cross Validation will be quite accurate
  • For very sparse datasets, you may have to consider leave-one-out
    • To get maximum number of experiments

• A common choice for $k$-fold Cross Validation is $k=10$

Three-way data splits

• If model selection and true error estimates are computed simultaneously
  • The data needs to be divided into three disjoint sets
    • Training set
    • $\Omega$ to find the optimal weights
    • Validation set
      • A set of examples used to tune the parameters of a model
      • To find the "optimal" number of hidden units or determine a stopping point for the back propagation algorithm
    • Test set
      • Used only to assess the performance of a fully trained model

• After assessing the final model with the test set, you must not further tune the model

Plain Accuracy

• Classifier accuracy
  • General measure of classifier performance

Accuracy = (Number of correct decisions made) / (Total number of decision made)

• Pros
  • Very easy to measure

• Cons
  • Cannot consider realistic cases

The Confusion Matrix

• A type of contingency table
  • $n$ classes
    • $n \times n$ matrix
      • The columns labeled with actual classes
      • The rows with predicted classes

• Separates out the decisions made by the classifier
  • How one class is being confused for another
  • Different sorts of errors may be dealt with separately
Problems with Unbalanced Classes

- Consider a classification problem where one class is rare
  - Sifting through a large population of normal entities to find a relatively small number of unusual ones
  - Looking for defrauded customers, or defective parts
  - The class distribution is unbalanced or skewed

Which model is better?

Confusion Matrix of A evaluated with 1,000 datapoints

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>500</td>
<td>200</td>
</tr>
<tr>
<td>N</td>
<td>300</td>
<td>500</td>
</tr>
</tbody>
</table>

Confusion Matrix of B evaluated with 1,000 datapoints

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>200</td>
</tr>
</tbody>
</table>

Why accuracy is misleading

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>B</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Which model is better?

True Population

Confusion Matrix of A

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>100</td>
<td>900</td>
</tr>
<tr>
<td>N</td>
<td>100</td>
<td>900</td>
</tr>
</tbody>
</table>

Confusion Matrix of B

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>50</td>
<td>450</td>
</tr>
<tr>
<td>N</td>
<td>450</td>
<td>50</td>
</tr>
</tbody>
</table>

F-measure (F1 score)

- Summarizes confusion matrix
  - True positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN)

\[ F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]

- True positive rate (sensitivity) = TP/(TP+FN)
- False negative rate (miss rate) = FN/(TP+FN)

\[ \text{precision} = \frac{TP}{TP+FP} \]
\[ \text{recall} = \frac{TP}{TP+FN} \]

\[ \text{Accuracy} = \frac{(TP + TN)}{(P + N)} \]

Questions?