Data Mining Techniques for Recommending Stores to Shopping Malls

John Wu
Ying Zhang
Purnima Padmanabhan

New York University

jsw452@nyu.edu
zy674@nyu.edu pp1492@nyu.edu

April 30, 2015

Overview

Abstract

- Recommend the best store for a shopping mall (main focus)
- Run Hadoop technologies to filter data and obtain preliminary data analysis tasks
- Combine recommendation system algorithms to recommend stores for malls

Prior Work and Background

- Work by Dennis Shasha, Roy Lowrance, Joe Jean
  - Recommended category of stores to malls
  - Used distance of different mall
- Extensions
  - Recommending actual stores to malls
  - Include extra data sources such demographic information of malls and information of individual stores.

Motivation

- Users and Beneficiaries
  - Mall owners
  - Customers who would like to discover new stores
  - Search Engine Services such as yelp
- Discover new stores for users that are important to them.

Mall Data

Web-scraped basic information of shopping malls from MallsInfo.com (Jean):

- Location
- Stores that each mall has
- Count of Store Categories

Malls are further represented by their geographic county information based (census.org/bls.gov):

- Demographic Information
    - Racial Composition and Age Statistics
- Industry Information
    - Average Biweekly Earnings of Industries

Mall-Store Data Filter

Background

- stores listed in the mall dataset are not store, for example: atm, vending machines , community service, ... etc. These are delete these items
- Most stores have different names that need to be aligned the name the same
- Lots of stores have special characters and misspelling

Solution

- created three different regular expression rules to filter our data and create unique id for every store
- Most Stores have different names, for example: Starbucks and Starbucks Coffee, and they need to be aligned the name the

Industry Data Filter

Problem

- Some counties do not have the same industry information
-
- Need to have counties to have the same information as all of the industries

Solution

- Find the intersection of all counties industry, and just keep the common industry in the data.
- Implemented a join for all geographic data , industry data and mall data

Store Data

Web-scraped store information (yelp.com):

- Average rating of stores (out of 5 stars)
- Number of Ratings
- Indicator of expensiveness (out of 4 dollar signs)
- Category of store

Hadoop Technologies

Implementation is done through AWS platform

- Pig: Format the mall data into a nicer form. Provides easy table manipulation compared to SQL
- MLLib: Implementing Logistic Regression
- Hive: Complement to MLLib

Recommendation Systems

Background

- Typical recommendation system problems have large amounts of missing entries in an user-item matrix and each filled entry indicates the amount of utility that a user has for an item
- Recommendation systems use information from filled entries to extrapolate values of missing entries

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- estimate rating of movie 1 by user 5

Collaborative Filtering

- Consider user ***x***
- Find set ***N*** of other users whose ratings are "similar" to ***x***'s ratings
- Estimate ***x***'s ratings based on ratings of users in ***N***

User-Based Collaborative Filtering

Matrices

- $U = (u_{jf})$: Feature User matrix
- $S = (s_{ij})$: Similarity Matrix for users
  - Measures similarity between *ith* and *jth* user by using an arbitrary function $f(U_{i.}, U_{j.})$
- $X = (x_{ij})$: Item-User Matrix

Algorithm

- Compute $\forall s_{ij} \in S$: $s_{ij} = f(U_{i.}, U_{j.})$
- For all missing entries X_{ij}
  - Compute $X_{ij} = \frac{\sum_{l \in N(j;i)} s_{il} X_{il}}{\sum_{l \in N(j;i)} s_{il}}$
    - $N(j;i)$ is the set of users that rated i and is in the same neighborhood as user j

Item-Based Collaborative Filtering

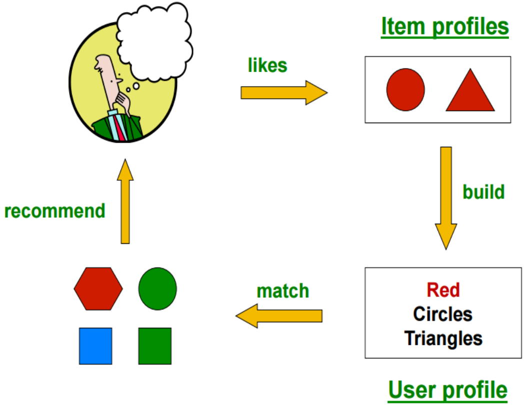
Matrices

- $I = (i_{if})$: Feature Item matrix
- $S = (s_{ij})$: Similarity Matrix for items
  - Measures similarity between *ith* and *jth* item by using an arbitrary function $f(I_{i.}, I_{j.})$
- $X = (x_{ij})$: Item-User Matrix

Algorithm

- Compute $\forall s_{ij} \in S$: $s_{ij} = f(I_{i.}, I_{j.})$
- For all missing entries X_{ij}
  - Compute $X_{ij} = \frac{\sum_{l \in N(j;i)} s_{il} X_{lj}}{\sum_{l \in N(j;i)} s_{il}}$
    - $N(j;i)$ is the set of users that rated i and the same neighborhood as user j

Content-Based Recommendations



Content-Based Recommendations

Matrices

- $U = (u_{jf})$: Feature User matrix
- $I = (i_{if})$: Feature Item matrix
- $S = (s_{ij})$: Similarity Matrix between item i and user j.
    - Measures similarity between *ith* and *jth* user by using an arbitrary function $f(I_{i.}, U_{j.})$
- $X = (s_{ij})$: Item-User Matrix

Algorithm

- Compute $\forall s_{ij} \in S$: $s_{ij} = f(I_{i.}, U_{j.})$

One-Class Data

- All the entries are binary and every entries are filled in
- All of the negative examples and missing positive examples are mixed together as 0s and cannot be distinguished
  - Positive examples are 1s
  - Traditional recommendation techniques do not necessarily apply to these problems

One-Class Collaborative Filtering (OCCF)

Matrix

- $X = (X_{ij})$: Binary Matrix
- $X = (X_{ij})$: Prediction Matrix (Result)
- $W = (W_{ij})$: Weight Matrix

Optimization Problem

- Called One-Class Matrix Factorization (OCMF) to avoid ambiguity
- Common solution for OCMF

$$\underset{R}{\text{minimize}} \quad \sum_{ij} W_{ij}(R_{ij} - X ij)^2$$

Intuition of W

- We are confident that some of the values X_{ij} must hold true. We know that a user likes a item since X_{ij} is 1, so we must give it high penalization weight W_{ij} if it is the prediction R_{ij} is greatly different from X_{ij}

Factorizing Prediction

- To have a low-rank approximation of X and to prevent overfitting, factorize X

New Optimization Problem [Rong (2001)]

$$\underset{U,V}{\text{minimize}} \quad \sum_{ij} W_{ij}((UV)_{ij} - Xij)^2 + \lambda(\|U_{i.}\|_2^2 + \|V_{.j}\|_2^2)$$

- Alternating Least-Squares can solve this optimization problem in a straightforward manner

Modeling the Problem

Store-Mall Matrix $\mathbf{X}$

- $S =$ store set
- $M =$ Mall set, $\exists_{S_{sub} \subset S} \exists_{m_i \in M} \forall_{s \in S_{sub}} s \in m_i$
- $X_{ij} = \begin{cases} 1 \text{ if } s_i \in m_j, s_i \in S, m_j \in M \\ 0 \text{ if } s_i \notin m_j, s_i \in S, m_j \in M \end{cases}$
- Ultimate goal is to recommend a store to a mall by converting an entry of X from a 0 to a 1
- Our problem is a One-Class data problem

Binary Classification

Algorithm

- Train a logistic regression and random forest classifier to predict whether a store i is in mall j.
- $\forall s \in S$
  - X: Demographic Information of the Malls
  - $y_j = \begin{cases} 1 \text{ if } s \in m_j \\ 0 \text{ otherwise} \end{cases}$
  - Regression on y_j from X for S
  - Recommend s to m_j if $y_j = 1$

Popularity

- Recommend the most frequently appeared stores to malls

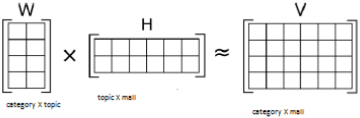
Algorithm

- K: Top K recommendations
- Compute $Pop = RowSum(X)$
- $SortedPop = Sort(Pop)$
- $\forall m_j \in M$
  - X: Recommend top k stores s_i from $SortedPop$ where $X_{ij} = 0$

Nonnegative Matrix Factorization

- It is useful to find the underlying "topics" of malls
- Malls are assumed to be linear combination of topics
- Each topic is a distribution over the category of stores
- We further enforce sparsity constraints on W and H

$$\underset{U,V}{\text{minimize}} \quad \frac{1}{2}\|X - UV\|_2^2 + \lambda(\Sigma_{s_i \in S}\|U_{i.}\|_1^2 + \Sigma_{m_j \in M}\|V_{.j}\|_1^2)$$



Discovered Topics

Modified Collaborative Filtering

- Traditional Collaborative Filtering are not necessarily applicable to this problem, since there are no missing values
- Modify $N(j;i)$: the neighbors of user j (Li 2010)
  - Recompute X_{ij} by the average binary rating of all users with similarity metric

Algorithm

- Compute $\forall s_{ij} \in S$: $s_{ij} = f(U_{i.}, U_{j.})$
- $\forall\ X_{ij} = 0$
  - Compute $\hat{X}_{ij} = \frac{\sum_{l \in N(j;i)} s_{il} X_{il}}{\sum_{l \in N(j;i)} s_{il}}$

$X_{ij} = \hat{X}_{ij},\ \forall\ X_{ij} = 0$

Mall Comparison

Metrics

$$Cosine(u, v) = \frac{u \cdot v}{\|u\|\|v\|}$$

$$Gauss(u, v) = ce^{-\frac{\|u-v\|^2}{\sigma}}$$

- S is based on comparing features
- Candidate features for comparing malls:
  - Demographic
  - Industry
  - Demographic + Industry
  - Store Category Composition
  - Demographic + Industry + Store Category Composition
  - Latent Topic Composition

Mall Comparison

Weighted Metrics

- Motivation: Nearby malls compete with each other
- Nearby malls should be dissimilar with respect to distance.

$$s = s_{old}(m_i, m_j) d_H(m_i, m_j)$$

- $d_H(m_i, m_j)$: Haversine distance (arclength distance between two points on a sphere)

Store Comparison

Metrics

- Functions are the same as malls
- Candidate features for comparing stores:
  - Category
  - Price Range
  - Rating
  - Category + Price Range + Rating

Content-Based Recommendation Systems

- It is difficult to find features that allow features for stores to be related to features of malls
- Could be extremely useful

Comparing Malls and Stores

- Create new store feature profile $\hat{s}$ by averaging all the feature of malls that have that new store
- $\hat{s}$ has the same features as m, thus use the same comparison metrics as malls.

One-Class Matrix Factorization Weights

- More geared to the actual problem
- Finding the best values for W_{ij}

	Pos Examples	Neg Examples
Uniform (Rong 2008)	1	$W_ij = \delta$
Mall-Oriented (Rong 2008)	1	$W_ij \propto \Sigma_j X_{ij}$
Store-Oriented (Rong 2008)	1	$W_ij \propto 1 - \Sigma_i X_{ij}$

Table: Finding weights

Simulated Annealing

- Many of the algorithms mentioned previously have their pros and cons
    - (Will go over more details later)
- Many recommendation systems are successful by combining the strengths of these models together with a linear combination (Netflix)
- Simulated Annealing is popular:

Algorithm 2: Pseudo code of Simulated Annealing.

Data: $\hat{r}_L^k(u,i)$ from different models on the validation data
Result: linear combination parameters β
$\beta \leftarrow$ random number vector $error \leftarrow Error(\beta)$ $T \leftarrow 10$ **while** $T > \varepsilon$ **do**
  $\beta' \leftarrow RandomModify(\beta)$ $error' \leftarrow Error(\beta')$ **if** $\min(\exp(\frac{error - error'}{T}), 1) > rand(0,1)$ **then**
    $\beta \leftarrow \beta'$ $error \leftarrow error'$
  end
  $T \leftarrow T \times 0.99$
end
Function: $RandomModify(\beta)$
begin
  $\beta' \leftarrow \beta$ **if** $rand(0,1) > 0.5$ **then**
    Choose two different positions i, j randomly $x \leftarrow \min(\beta_i, \beta_j) \times rand(0, 0.2)$ $\beta'_i \leftarrow \beta_i + x$ $\beta'_j \leftarrow \beta_j - x$
  else
    Choose one position i randomly $\beta'_i \leftarrow \beta_i \times rand(0.8, 1.2)$
  end
  return β'
end

Cross-Validation (CV)

CV for Recommendation Systems

- A 80/20 fold validation will be used for evaluating these algorithms.
- For the X data matrix, 80% will be noted as training data and the other 20% will be testing.
- Goal is to accurately predict the actual values of the training data

Cross-Validation for Recommendation Systems

Table: Actual Data

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Table: Training Data

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Table: Transforming Data

- Goal: Accurately transform 0s in training data to the ground truth of 1s

Recommendation Evaluation

Accuracy

Can be misleading since there are so many positive examples compared to negative and vice versa.

Precision

The percentage of items that are correctly labeled as 1s given the items that the recommendation system predicted as 1s

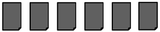
$Precision \frac{TP}{TP+FP}$

Mean Average Precision (MAP)

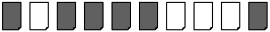
Summarize rankings from multiple queries by averaging average precision of all users

$AP_{\mu} = \frac{\Sigma_{i=1}^{N} prec(i) \times pref(i)}{\# \text{ of preferred items}}$

Precision Example

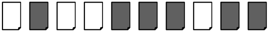

= the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2



Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

- Recall $= 5/6 = 0.83$
- Precision $= 5/6 = 0.83$
- Precision $= \dfrac{RelevantRetrieved}{Retrieved}$
- Recall $= \dfrac{RelevantRetrieved}{Relevant}$

MAP Example

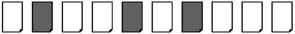

= relevant documents for query 1

Ranking #1

Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5


= relevant documents for query 2

Ranking #2



Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

- Average Precision Query 1 $= (1.0 + .67 + .5 + .44 + .5)/5 = 0.62$
- Average Precision Query 2 $= (.5 + .4 + .43)/3 = .44$
- Mean Average Precision $= (.62 + .44)/2$

Conclusion

- Bibliography could not be shown because of using technical errors using latex.