Machine Learning 202

Recommender Systems

Outline

- Background
- Collaborative Filtering for 0-1 Data
  - User based CF
  - Item based CF
  - Association Rules
- Evaluation of "top-N" recommender algo
- Examples using recommenderlab from cran r on MS weblogs

Netflix Problem

- Customer logs onto Netflix site
  - Has known history w Netflix
  - Past movie ratings
  - Movies watched

- What movies should Netflix promote to the user?

Netflix Prize

- Netflix had a system in place to predict how a user would rate movies they hadn't seen. They wanted better performance

- In 2006 Netflix decided to have a contest

- They offerred $1 million to first person (or team) that could improve upon Netflix prediction performance by 10%

Scale of the problem and the contest

- 480,000 users 17,770 movies, 100 million ratings
- 6 years of data 2000 through 2005
- 2700 teams enter competition
- 3 years to finish.

Recommendation – Ask your Brother

- Find people with similar tastes and ask them for recommendations
- Called "Collaborative Filtering"
- Transaction-based
- Characterize movies based on who gives them the same ratings.

Collaborative Filtering

- Here's a table of ratings

	User1	User2	User3
Movie1	4	3	0
Movie2	5	3	1
Movie3	0	0	5

- Movie1 is closer to Movie2 than it is to Movie3, based on user ratings. (and User1 is closer to User2)

Collaborative Filtering - Binary Data

- Suppose that all we have is data on what movies were watched

	User1	User2	User3
Movie1	1	1	0
Movie2	1	1	1
Movie3	0	0	1

- Without the rating proximity is obvious
- Binary data are completely objective

Collaborative Filtering

- This is significantly more precise than attribute-based
  - Don't need to tell it which attributes are important
  - Exploits judgments of other users
  - Not limited by user's self designated profile
  - Not limited by movie's self-reported profile

And

- **Transaction records become a source of competitive advantage!**

Other Problems Amenable to this Approach

- Movies
  - Based on Movies Watched (versus ratings)

- Books (electronics, cameras, etc)
  - Based on Purchase Transactions (Amazon, ebay, etc.)

- Ad serving
  - Based on Ads clicked
    - double click (do you auto-delete the dc cookie?)
    - google (are you signed in?)

- Others?

Outline

- Background
- Collaborative Filtering for 0-1 Data
  - User based CF
  - Item based CF
  - Association Rules
- Evaluation of "top-N" recommender algo
- Examples using recommenderlab from cran r on MS weblogs

Collaborative Filtering on 0-1 Data

- Set of users - U = {u1, u2, … , um}
- Set of items - I = {i1, i2, … , in}
- Matrix of ratings, or 0-1's
  - R = {rij}
  - rij = 1 if user i has preference for item j
    = 0 otherwise
- See any problems?

0 is different from 1

- An entry of "1" in the matrix means interest (or click or purchase, etc.)
- What does a "0" mean?
  - User not aware of product
  - User hasn't wanted it up to this point in time
  - User dislikes product
- One-class data (recall using one class svm for fraud detection)

What to do with "0"

- Usually don't have data to distinguish the different reasons for inaction (not clicking a link, etc.)
- Could use one-class tech
- Usually treat different meanings as a single class - results legitimize this approach

Problem Formulation

- For user "a" ua ϵ U (called the "active" user)
- Let Ia = I \ {il ϵ I such that ral = 1}
  Ia is the set of items not selected by user "a"

- Predict ratings for all elements of Ia
  or
- Create a list of top N recommendations

Types of Algorithms

- Memory-based - Search whole data base to develop ordered set of recommendations
  - User-based CF
  - Scalability problem

- Model-based - Use db to learn compact representation of answers

Outline

- Background
- Collaborative Filtering for 0-1 Data
  - User based CF
  - Item based CF
  - Association Rules
- Evaluation of "top-N" recommender algo
- Examples using recommenderlab from cran r on MS weblogs

User-based CF

- Mimics word of mouth
- Find a neighborhood of users with similar tastes
- Neighborhood defined by similarity (or distance) measure
  - Pearson correlation
  - Cosine similarity
  - Jacard similarity

Similarity Measures

- Pearson correlation
  Sp(x,y) = $\frac{\sum_{i \in I}(xi - xavg) * (yi - yavg)}{(|I|-1) * sd(x) * sd(y)}$

- Cosine similarity
  Sc(x,y) = $\frac{x \cdot y}{||x||2 * ||y||2}$

- Jacard similarity
  Sj(x,y) = $\frac{|X \cap Y|}{|X \cup Y|}$

Develop Ratings for Ia

- Use similarity measure (or metric) to define a neighborhood N of ua (active user).

- Basically average the other user's ratings to estimate ua's rating.

Outline

- Background
- Collaborative Filtering for 0-1 Data
  - User based CF
  - Item based CF
  - Association Rules
- Evaluation of "top-N" recommender algo
- Examples using recommenderlab from cran r on MS weblogs

Item-Based CF

- Model Building - Build an item-item similarity matrix - S
- Normalize S so that rows sum to 1.
- For each row (item) set to zero all but the largest similarities (to reduce model size)
- For each item calculate score by adding together similarity with active user's items
- Remove items already in actives user's set

Item-based CF

- More efficient for computer time and storage than user-based
- Only slightly inferior in performance
- Successfully applied to large-scale problems (e.g. Amazon)

Outline – Where are we?

- Background
- Collaborative Filtering for 0-1 Data
  - User based CF
  - Item based CF
  - Association Rules
- Evaluation of "top-N" recommender algo
- Examples using recommenderlab from cran r on MS weblogs

CF Using Association Rules

- What are association rules?
  - Let I = {i1, i2, … ,in} be a set of items (peanut butter, jelly, etc)
  - Let D = {t1, t2, … tm} be a set of transactions
    - Each ti a subset of I - (shopping cart)
- Association rule is an implication of the form:
  X => Y where X,Y are both subsets of I and $X \cap Y = \emptyset$
  (chips => dip)

Support and Confidence

- Support – For a set of items A subset of I support is
  support(A) = |{ti | A is subset of ti}| / |D|
- Support for an a-rule – For disjoint sets X, Y (subsets of I)
  support(X =>Y) = support($X \cup Y$)
- Confidence -
  confidence(X=>Y) = support ($X \cup Y$)/support(X)

A-Rules for DF

- Treat each user's 1's as a single transaction
- Calculate rules of the form X=>Y with highest confidence
- For X's that are subsets of active user's chosen items look up Y's and rank by confidence

Outline – Where are we?

- Background
- Collaborative Filtering for 0-1 Data
  - User based CF
  - Item based CF
  - Association Rules
- Evaluation of "top-N" recommender algo
- Examples using recommenderlab from cran r on MS weblogs

Evaluation of Top-N recommender algorithms

- Given matrix R –
  - Partition R – some rows for "test_set" the rest for "train_set"
  - Train also on train_set
  - Test performance on test_set
- For testing
  - Treat each user as "active" user
  - Remove some of user's actual selections
  - See if given Top-N recommender algo replaces removed selections

How to Split R

- Simple Split (for large data)
  - Pick a reasonable fraction (30% test, 70% train)
  - Sample at random
- Bootstrap Sampling – (for small data)
  - Sample with replacement to form training set
  - Test on users not included in training set
- k-fold Cross-Validation
  - Divide users into k equal groups
  - Run k training/testing passes holding out a different one of k groups for testing on each pass

Delete Items for Test Users

- "Given j" – Keep "j" transactions and build recommender to fill in the others
- "All but j" – Delete "j" transactions

Evaluating Performance

- For each user in test set generate Top-N recommendations
- Build confusion matrix:

Actual/Predicted	Negative	Positive
Negative	a	b
Positive	c	d

- Notice b+d = N, c+d = # withheld
- Some Performance Terms

  Accuracy = (a+d)/(a+b+c+d)

  Precision = d/(b+d)

  Recall = d/(c+d)

  TPR = Recall

  FPR = b/(a+b)

Discussion re Evaluation

- To evaluate performance can use ROC curve AUC and tools we discussed in ML 101
- This scheme doesn't distinguish between getting good recommendation at 1^{st} or 5^{th} in sequence – that may make a difference

Singular Value Decomposition

- Suppose M is an mxn matrix
- Singular Value Decomposition of M is a product of matrices
  $M = U\Sigma V'$ (' means matrix transpose)
  where
  U = mxm unitary matrix ($UU' = U'U = I$)
  Σ = mxn diagonal matrix of singular values – the singular values are all positive and arrange in decreasing magnitude
  V' = nxn unitary matrix

Low-Rank Approximation using SVD

- SVD can be used to generate low-rank approximations as follows.
- Suppose $M = U\Sigma V'$, as above. If we want an approximation to M that is of rank k (less that the rank of M).
- Form $\Sigma_k = \Sigma$ (with singular values smaller than the largest k set to 0)
- Then $M_k = U\Sigma_k V'$ is the closest rank k approximation to M in the sense of Frobenius norm.

How Does SVD Help?

- Think of SVD as finding an abstract concept space where the importance of concepts are indicated by the singular values
- U maps users into the concept space. V' maps items (movies, web pages, ads) into concept space.
- In concept space we can compare a movie and a user directly to one another.

Calculate Similarity Using SVD

- Recall $M = U\Sigma V'$
- M is mxn (by convention m = #users, n = #items)
- Take a unit vector in item-space, call it e_i (vector of 0's except i^{th} element which is 1)
- Me_i maps the i^{th} item from item space to user space (the vector of users who selected the i^{th} item)
- $\Sigma V'e_i$ is a column vector in concept space that represents the i^{th} item.

Calculate Similarity Using SVD

- Users are represented by a vector in item-space (vector with 1's where corresponding to items of interest)
- Items are represented by a vector in item-space (e_i)
- Map the user and the items to concept-space using truncated SVD ($\Sigma_k V'$) and compare using directional similarity like correlation