Collaborative Filtering

Amazon Recommendation System

## Objective

In this project you will analyze the [Amazon Product Reviews dataset](http://jmcauley.ucsd.edu/data/amazon/) and train an SVD-based recommendation system. You will train the top few factors of the product-reviewer ratings matrix and analyze their properties. Using cross validation you will find the optimal number of factors that you need for your model. You will see how the system prediction compare to the “Also Bought” metadata in Beauty products. You will backtest your model and predict future user-product ratings using information of past ratings.

## Data Description

The Amazon reviews dataset was gathered by Prof Julian McCauley at UCSD and it contains all the reviews of 9.4 milliion products sold on Amazon between 1995-2014. The dataset is part of the Stanford SNAP database and can be found here:

● SNAP raw dataset:<https://snap.stanford.edu/data/web-Amazon.html>

● Julian’s cleaned up dataset:<http://jmcauley.ucsd.edu/data/amazon/>

Each review associated with a reviewer ID and a product ID, has a time stamp, star rating from 0 to 5 and review text. In this dataset the reviews are broken by product categories, such as Books, Electronics, Video Games etc. In addition, courtesy to Julian McCauley, the data is cleaned up from redundancies and restricted to reviews by active users (e.g. users who have reviewed at least 5 items by 2014). Finally, there is also a product metadata database, which contains essential metadata associated with each product ID

This dataset is important because:

● All the reviews of all products on Amazon, contain information about consumer and product trends.

● All the reviews and their corresponding ratings provide for a training corpus for a review sentiment.

● The (product, user rating) pairs can be used to build a user-product recommendation engine.

In this project you will focus on the Electronics, Movies and TV, Clothing Shoes and Jewelry, Cellphones and Accessories, Tools and Home Improvement, Beauty, and Baby ratings categories located in datasets/amazon/ratings\_5core/. Each element in the ratings dataset Each element in these datasets consists of a reviewer ID, product ID, time stamp, and a star rating from 0 to 5. In addition, courtesy to Julian McCauley, the data is cleaned up from redundancies and restricted to reviews by active users (e.g. users who have reviewed at least 5 items by 2014). Finally, there is also a product metadata database, which contains essential metadata associated with each product ID. Examples of the data entries as well as methodology on how to read the data is listed in the Julian McCauley [website](http://jmcauley.ucsd.edu/data/amazon/).

## Methodology and Deliverables

### Create Databases

Create the following databases (in a dataframe or HDF5 format) for each of the product categories Electronics, Movies and TV, Clothing Shoes and Jewelry, Cellphones and Accessories, Tools and Home Improvement, Beauty, and Baby:

* df\_reviews\_categoryname: whose columns are timestamp, productid, reviewerid, rating, review\_text, review\_summary
* df\_products: whose columns are productid, title, imUrl, brand
* df\_products\_also\_bought: indexed productid, contains also\_bought column
* df\_products\_also\_viewed: indexed productid, contains also\_viewed column
* df\_products\_bought\_together: indexed productid, contains bought\_together column
* df\_products\_sales\_rank: indexed by productid contains sales\_rank
* df\_products\_categories: indexed by productid contains categories column

### Sparse-SVD algorithm

For this project the main quantity of interest is the product-reviewer ratings matrix X whose ij-th entry is the review rating for the i-th product by the j-th reviewer. You will need to construct this matrix for various type of product classes and during various periods. In general X will be a very sparse matrix as a typical user reviews very few products. A missing ij entry of X corresponds to a missing review of the i-th product by the j-th reviewer. That doesn’t mean that if asked about the i-th product, the j-th reviewer won’t form an opinion. All this means is that such opinion hasn’t been observed yet.

Based on this interpretation, the Netflix challenge gradient descent algorithm that you did as part of your Homework is appropriate. Why?

Next, do the following:

* Construct X for each of the product categories above for ratings formed in 2013 as well as for the merged dataset of all seven categories above.
* How big is the matrix in each case? What is its level of sparsity?
* Fit the top-K factors of the matrix X in each case. Using cross-validation, find the optimal K and learning rate for your problem. In other words, for each K, find the mean prediction error for each entry of X using 5-fold cross validation. Choose the optimal K that minimizes the cross-validation prediction error. What are the optimal number of factors for each of the product categories as well as for the merged dataset?
* For sanity check, fit the optimal K using 5-fold cross-validation for a 70% sparse Einstein image as in your homework problem. What is the optimal K in this case?

As a result of the previous exercise, you will end up with a decomposition:

* In the above and is a diagonal matrix with decreasing positive entries. What is your interpretation of the columns of and ?
* Since X is a positive matrix, what is your interpretation of the first columns of U and V?
* How would your result change if instead of doing SVD of X you did the SVD of the row-demeaned or column-demeaned version of X? Can you relate the SVD of the row-demeaned and column-demeaned version of X to that of X?

### Sparse-SVD Amazon Recommender

Once you’ve chosen the optimal K:

* Create the product-reviewer matrix. If you need to at first, restrict to a smaller subset of the product-reviews corresponding to certain product category subsets in df\_products\_category\_Beauty.
* Now you have an optimal approximation X’ of the product-reviewer matrix X, whose ij-th entry is the estimated rating for product i of user j.
* Since some of the ratings might be beyond the [1-5] range, use Simon Funk’s suggestions of transforming (i.e. winsorizing) the matrix entries via tanh. Let’s call the properly transformed matrix Y.
* From Y compute a product-product covariance matrix P = YY^T. How are the entries of P related to the “Also bought” and “Also viewed” metadata? Investigate. If you scatter the components of the top eigenvector of P vs the second eigenvector, how can you interpret different regions of the scatterplot? Discuss.
* Do the same exercise for the user-user matrix Q=(Y^T)Y.

### Test Recommender

Once you’ve trained for the optimal K and learning rate for each dataset, you now have a way to complete the missing entries of X from the product ratings in 2013. In other words, let all the ratings done outside of 2013 be considered your test set. They would correspond to entries of X that you didn’t observe in 2013. For these test entries, you can calculate the prediction accuracy based on your model trained from 2013 data

* How does this prediction accuracy compare to that of the *uniform* *guess* accuracy, e.g. the accuracy of predicting each user-product rating to be equal to the mean user-product rating. Plot the histogram of the error of the uniform guess for a set of 100,000 randomly selected ratings from the test set.
* For each of the product class models, plot the prediction error histogram of the same 100,000 randomly selected ratings from the test set as well.
* How does your test error compare to your cross-validation error?
* What are some typical classes of user profiles? What are some typical classes of product profiles?

### README file

Create a README\_db.txt file containing:

* The names and emails of all the teammates so you can be contacted by the next user of the dataset
* Description of the file naming convention, fields for each dataset
* Any comments on data features other users should be aware of when they use your data.