Capstone Proposal

Maher Malaeb
November 20, 2016

Domain Background

Online businesses such as Amazon (Crab), Netflix (Wikipedia, 2016), etc. relies heavily on recommendation algorithms to provide the best user experience and display to the user just what he is interested in seeing. Facebook displays people I am interested in becoming friends with, relevant ads and meaningful posts to me (Facebook, 2015). I have always been fascinated with the predictive power of these companies and the huge advantages it bring to their business, and its impact on everyone's life.

What all these companies share in common is a special category of machine learning algorithms called recommendation systems. These algorithms has moved recently from big companies and research groups to everyone thanks to advances in computational power, availability of easy-to-use programming libraries and accessibility of big datasets.

Problem statement

Throughout the project, we will create a recommendation system based on Amazon's rating data set (McAuley)Based on a user's previous rating of a set of items, the system will try to predict of list of n-items that this particular user will likely give them high rating. This can be achieved by building a user based collaborative filtering algorithm which is a popular method in recommendation systems. The results of can be evaluated using precision and recall metrics, F1 score, root-mean-square error (RMSE) and other more complex approaches.

Datasets and inputs

The dataset that will be used in this project is the "Clothing, Shoes and Jewelry" 5-core rating data which is part of the "Amazon product data". The dataset is publicly available on the web and it represents real reviews of products on Amazon divided by category (McAuley). In this project, the focus will be on the "Clothing, Shoes and Jewelry" . Direct download link is here

What makes this data a good fit to our problem are several characteristics. At a macro level, it is a big dataset with 278,677 reviews. It has "good" content because it in 5-core format where each item has at least 5 reviews and each user made at least 5 reviews.

A sample instance of the dataset is below. The 3 features important for our project are the "reviwerID" , "asin" and "overall" score. After dividing the dataset into test and train partitions, The idea is to predict top N most highly rated items for a specific user from the items list. This can be done by checking the similarity between the items the user under question has rated with other users who had similar rating on similar items.

```
{
        "reviewerID" : "A3EERSWHAI6SO",
        "asin" : "0000031887",
        "reviewerName" : "Jeffrey Hollingshead \"Jillian hollingshead\"",
        "helpful" : [7, 8],
        "reviewText" : "For what I paid for two tutus is unbeatable anywhere!",
        "overall" : 5.0,
        "summary" : "WOW !! ..is all I have to say!",
        "unixReviewTime" : 1349568000,
        "reviewTime" : "10 7, 2012"
}
```


Solution Statement

When it comes to solutions, the options are many. There are a lot of python libraries that implements collaborative testing algorithms. A non-comprehensive list is GraphLab, Crab, LightFM, pysuggest, python-recsys. These libraries takes the users and ratings dataset as input and does predictions using different algorithms. Once we understand the similarities and differences between the libraries after comparing the underlying algorithms, scalability, efficiency, customizability. We can start applying the different libraries method on our data set and compare results and performance.

Benchmark Model

Since this dataset is relatively and hasn't been widely used in the research community. There are no readily available benchmarks to test our model against. Thus what should be done is building our own benchmark. The benchmark will be to give all missing ratings a rating equal to the mean rating of the train set.

Evaluation Metrics

To evaluated our model, we will use 2 kinds of evaluation metrics.

1. Numerical metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE): Since our dataset consists of a list of known ratings, these types of metrics can be used to measure the difference between the actual rating and the predicted rating. Minimizing errors can help validate and evaluate the model.
2. Binary metrics: such as precision and recall can be used after defining a rating threshold where all rating above it can be considered positive and below it negative. Thus the precision will give us the percentage of recommendation that are correct (whether positive or negative) and recall will give us the percentage correct recommendation from the list of actual recommendations. This is important because the aim of recommendation systems is to provide a list of relevant results to the user.

Project Design

First we start by building the user-item matrix. The rows of the matrix are the items and the columns are the users. If a user 'u' has rated an item 'i' as x. Then the matrix entry of [i,u] will be x. It will be zero if the rating is not available.

	It1	It2	It3	It4	It5	It6	It7	It8	It9	...
U1	0	0	2	0	4	0	0	0	0	
U2	1	0	0	0	0	3	0	5	2	
U3	0	2	3	0	3	2	0	0	0	
U4	0	2	1	0	2	1	0	3	0	
U5	0	0	2	0	0	0	4	0	0	
U6	0	0	2	0	0	0	0	0	0	
U7	1	0	0	0	0	2	0	2	0	
U8	0	3	0	3	0	0	5	0	1	
U9	0	0	3	0	0	0	0	0	1	
U10	2	1	3	0	2	0	1	0	0	
U11	0	1	0	1	0	3	0	3	1	
...										

Figure 1 - User-Item Matrix (Bigdata Doc, 2014)

This matrix is then used as an input for the collaborative filtering algorithm which goes as follows

input : Number of items to be recommended $N \in \mathbb{N}$,
Number of neighbors used for ranking $k \in \mathbb{N}$,
User to recommend items to u,
List of all items $Items$,
User-Item matrix of ratings R
output: N items to be recommended

foreach $item \in Items$ **do**
  if $item \notin u.rated\_items$ **then**
    $item.rank \leftarrow$ rank_according_to_nearest_neighbors$(k, u, item)$

descending_rank_sort$(Items)$

return top$(N, Items)$

Figure 2 - High level pseudo code (Kordík, 2016)

Different models and techniques will be researched and tested following the same pseudo code above. The algorithms that are used in implementing collaborative filtering are many and based on Machine learning algorithms such as K-nearest neighbors, Bayesian Networks (Wikipedia, 2016), Matrix Factorization (Sedhain, 2016)

Works Cited

Bigdata Doc. (2014, December 24). *Recommender Systems 101 – a step by step practical example in R*. Retrieved November 20, 2016, from R-bloggers : https://www.r-bloggers.com/recommender-systems-101-a-step-by-step-practical-example-in-r/

Crab. (n.d.). *2. Getting started: an introduction to recommender systems with Crab*. Retrieved November 20, 2016, from Crab A Recommender Framework in Python: http://muricoca.github.io/crab/tutorial.html

Facebook. (2015, 2 June). *Recommending items to more than a billion people*. Retrieved November 20, 2016, from Facebook:

https://code.facebook.com/posts/861999383875667/recommending-items-to-more-than-a-billion-people/

Kordík, P. (2016, July 12). *Recommender systems explained*. Retrieved November 20, 2016, from Medium: https://medium.com/recombee-blog/recommender-systems-explained-d98e8221f468#.doog1nocm

McAuley, J. (n.d.). *Amazon product data*. Retrieved November 20, 2016, from Amazon product data: http://jmcauley.ucsd.edu/data/amazon/

MyMediaLite. (n.d.). *MyMediaLite: Example Experiments*. Retrieved November 20, 2016, from MyMediaLite: http://mymedialite.net/examples/datasets.html

Sedhain, S. (2016, April 13). *How exactly is machine learning used in recommendation engines?* Retrieved November 20, 2016, from Quora: https://www.quora.com/How-exactly-is-machine-learning-used-in-recommendation-engines

Wikipedia. (2016, November 15). *Collaborative filtering*. Retrieved November 20, 2016, from Wikipedia the free encyclopedia: https://en.wikipedia.org/wiki/Collaborative_filtering#Memory-based

Wikipedia. (2016, June 30). *Netflix Prize*. Retrieved November 20, 2016, from Wikipedia the free encyclopedia: https://en.wikipedia.org/wiki/Netflix_Prize