Collaborative Filtering For Recommendations

Alexander Barriga, *Galvanize, University of New Haven, Data Science*

***Abstract*—A Recommendation System for Wine.com is built using Spark's MLlib implementation of collaborative filtering. Data prepossessing includes transforming purchases data into categorical ratings to use in place of Wine.com's sparse ratings data. The model is validated using stratified LOOCV cross validation. Results show that the collaborative filter, through latent feature analysis, successfully learns and predicts user's wine preferences.**

Keywords*—*Collaborative Filtering, Spark, Big Data, Recommendation System, Wine.com

I. Introduction

In the past, users of e-commerce sites would face the same problem: they were overwhelmed by the vast amount of choices they have when shopping for a product. Sites that didn't address this issue were requiring that the user spend an undesirable amount of time searching through the large amount of choices and risk having the user leave their site for a competitor that can provide an enhanced shopping experience. The recommendation system provides e-commerce sites with a competitive edge because it addresses this issue for the benefit of the user and the business alike.

A recommendation systems provides personalized product recommendations based on users past purchase history, ratings, reviews, views of products, and other collected information about users. The site can then provide each user with a recommended products page that is populated with products that have a high predicted probability of being purchased by users. A recommendation system benefits the user by reducing the amount of time that she needs to spend on a site before she make a purchase; this is accomplished by presenting products that appeal to the users personal preferences. The business benefits by providing a pleasant and time-saving shopping experience for users. This is likely to help retain users, attract new users, and increase sales significantly. Indeed, sites like Netflix have created an incredibly disruptive and successful business around the recommendation system.

This paper discusses how a recommendation system was built to address the problem of vastness of choice for Wine.com and their users.

A. Overview and Hypotheses

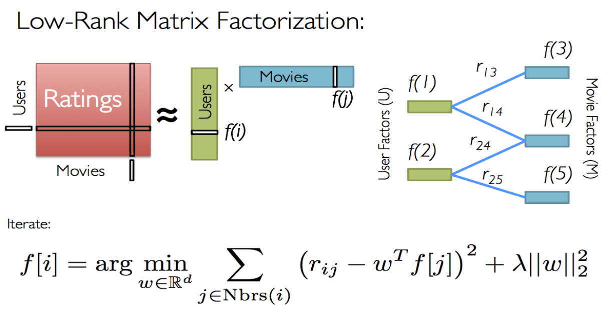
There are many choices for a data scientist when deciding on an approach to build a recommendation system. Due the sparsity of data, sophisticated implementations of matrix factorization have been developed, such as Non-Negative Matrix Factorization with Sparseness Control [1] as well as Factorization Machines [2]. Factorization Machines (FM), in particular, have gained popularity for use as recommenders. FM scale linearly with rows, find latent features, handel sparisity well, and have the option of fining higher-order latent features. Another popular approach based on matrix factorization is a machine learning algorithm known as Collaborative Filtering. Wine.com's recommendation system will center around Spark's implementation of collaborative filtering due to it built in big data technology.

A Collaborative Filter takes data that relates users ratings to products and learns how to predict a users rating on a product that they have not purchased yet. This is done by finding users with similar preferences. If two users have similar preferences, but one has purchased a product that the other has not ,then based on the similarity of their preferences, a rating can be predicted for the user who has not made that purchase.

Although this approach can be very effective, there are issues that must be addressed. First, rating data is almost always spares. Users dont often rate the product they purchase, much less leave a review. In Wine.coms case, only 2.5% of users have rated a wine. Second, there is the Cold Start problem: a Collaborative Filter can only provide predictions for users that have previously rated a product. So new users can not have recommendations made for them with this approach. Also predictions are weak when the user has only rated one or two products.

The issue with the sparse rating data will be address by using the more abundant purchase data as a proxy for rating data. Based on the distribution of total purchases per bottle of wine, wines will be given a categorical value ranging from 1 to 10 in order to create a pseudo ratings matrix, R.

II. Theory of Solution



$$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

Figure 1: Spark's ALS

There are many implementations of Collaborative Filtering, an increasingly popular choice is to use the implementation from Sparks machine learning library, MLlib [3]. Spark is an open source, big data technology that is built on Hadoop and MapReduce and leverages cluster and parallel computing. Sparks scalability across computers, robust protection against data loss, and fast computation time make is a great choice for big data analytics and production models. Sparks implementation of collaborative filtering is called Low-Rank Matrix Factorization (LRMF).

By replacing sparse rating data with more abundant pseudo-rating data based on purchases, the issue of sparsity has only been partially addressed. The missing values are the predicted ratings for products that users have not purchased yet. Those values are imputed by Matrix Factorization (MF), optimized by Alternating Least Squares (ALS). The imputed values are discovered by learning a small set of latent factors. These latent factors are the similarities between products that the collaborative filter learns and uses to predict ratings. These latent features can be the varietal of the wines, the vineyards, or the years that they were made.

Matrix Factorization with ALS: the goal is to recreate an approximation of R by multiplying matrices U and V.

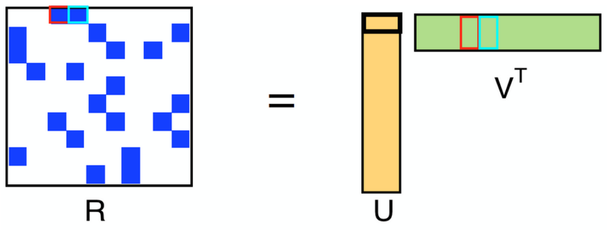


Figure 2: ALS cost function

1) We want $R \approx UV$
2) Start with random U_1, V_1
3) Solve for U_2 to minimize $||R - U_2V_1^T||$
4) Solve for V_2 to minimize $||R - U_2V_2^T||$
5) Repeat until convergence

Sparks LRMF operates a little differently than typical implementations of matrix factorization with ALS. It cache 2 copies of R in memory, one partitioned by rows and one by columns, It keeps U and V partitioned in corresponding ways, and operates on blocks to lower communication between processors.

III. Methodology

All data preparation and machine learning testing is implemented in Jupiter Notebooks and can be found at this project's github repository.

A. Prepping Data

Exploratory Data Analysis (EDA) and preliminary model testing indicated the degree to which the purchase data needed to be filtered.

First, all non red or white wine products are removed from the purchase data.The motivations for a user purchasing wines are, presumably, different from the motivations for purchasing whisky or champaign. Next, filter out customers with less than 5 red or white wine purchases of distinct wines. In order to detect latent features (wine preferences of users) there needs to be enough purchase data present, from each user, in order for patterns to emerge. Then filter out customers with no subsequent purchases. The goal of the recommender is to model the purchasing behavior of recurring users, so that recurring users remain as such and new users can become recurring users. Users who visit the site once and never return provide no predictive signal, only noise. Lastly, remove single purchases with more than 10 bottles of wine. We want to remove the bulk purchases made by wedding and party planers.These bulk purchases don't reflect any deep wine preferences made by individuals.

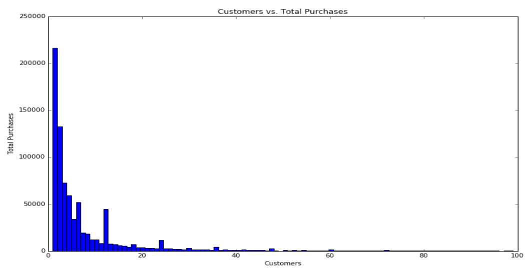


Figure 3: Number of purchases per user

Figure 3 is a Pareto distribution that shows that the vast majority of users, between 2005 and 2015, only purchased a few bottles of wine. About half of users who purchased wines, purchased 3 or less bottles in those 10 years. A quarter of users purchased a single bottle of wine. After all filtering of the purchase data was complete, it was discovered that about 20% of users are responsible for 70% of purchases.

Now that the purchase data has been filtered to maximize predictive signal, the data needs to be transformed into categorical ratings. There are many ways to transform raw user purchases into ratings. The approach taken here is to extract a relative rating for each wine based on the user's frequency of purchases for red and white wines.

For each distinct bottle of wine,

$$Rating_{u,w} = \frac{N_{bottles_{u,w}} - \mu_{(bottle \mid type)_{u,w}}}{\sum N_{(bottles \mid type))_{u,w}}}$$

$N_{bottles_{u,w}}$ is the number of purchases of wine, w. $\mu_{(bottle \mid type)_{u,w}}$ is the frequency of purchases for this type of wine (i.e. red or white) by user, u. $\sum N_{(bottles \mid type)_{w,u}}$ is the total number of purchase for this type of wine, by this user, $Rating_{u,w}$ is the rating of wine w by user u.

B. Model Validation

Typically, supervised machine learning models are validated with hold out cross validation; the model is fitted on a training set (x_{train}, y_{train}), then its performance is tested on the previously unseen holdout set (x_{test}, y_{test}). Collaborative Filtering is different in that it is trained on a portion of X in order to predict another portion of X. This difference is addressed by using Leave One Out Cross Validation (LOOCV).

LOOCV trains the model by iteratively increasing the size of the training set with the expectation that the test error will monotonically decrease. This testing is implemented with a stratified version of LOOCV; about equal proportions of all 10 categories are included in the train and test sets.

Further model validation is performed by plotting a ROC Curve. ROC curves show the relationship between the True Positive Rate and the False Positive Rate as a hyper-parameter, or some other threshold value, is varied. In order to plot roc cruves, the non-binary predicted ratings of the model mush be transformed into a binary classification. This is achieved in the following way.

Ratings vary from 1 to 10. Create a threshold value, t, and vary it from 2 to 10. Ratings below the threshold, t, are given the label 0, meaning don't recommend. Ratings above he threshold, t, are given the label 1, meaning do recommend. Test set ratings and predicted ratings are both compared to the threshold value as we iterate through threshold values (i.e. for t = i and rating = r, if r ¿= t then 1 else if r ¡ t then 0). Those 0 and 1 binary classifications enable us to build a confusion matrix. Which we can then use to get true positive and false positive rates. A more detail explanation of this procedure can be found in *An Evaluation Methodology for Collaborative Recommender Systems*. [4]

IV. Experimental results

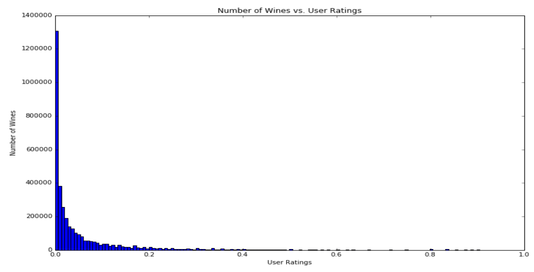


Figure 4: Purchases transformed into ratings

Figure 4 shows the distribution of the purchase data after having been transformed into floating point ratings. Unsurprisingly, the floating point ratings distribution is also a Pareto distribution, only scaled differently and smoother than figure 3. Figure 4 shows that over 90% of floating point ratings occur between 0 and 1; the maximum value of 8.9 is not seen on the figure.

Next, the floating point ratings data is transformed into categorical ratings ranging from 1 to 10. This is done by first sorting the ratings from smallest to largest values (as seen in 4), dividing the total number of ratings by the number of desired bins (i.e. 10) to get the bin count, and iteratively storing the ratings into bins until the bin count is reached, then moving on to the next bin. This numerical method ensure that each bin receives an equal amount of area under the curve from figure 4.This process results in a balanced set of labels.

The results from model validation were very encouraging. The figure 5 shows that, as the size of the training set increased, the error of the test set monotonically decreased and approaches the error of the training set. These results were obtained by setting the rank of the collaborative filter to 16. Rank is the number of latent features that the model will discover, and use, in order make predicted ratings.

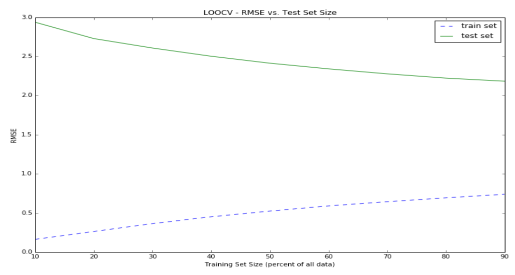


Figure 5: Model Error as Test Size Increases

Figure 5 shows the result from LOOCV for the optimized model. We see that, as the size of the test set increases, the test error monotonically decreases. Ultimately, the test error decreases by about 1 RMSE unit, appearing the asymtope at about 2.0 RMSE.

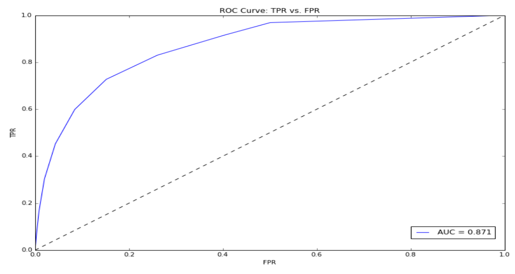


Figure 6: ROC Curve produced with all data

Figure 6 shows a relatively smooth curve with an area under of the curve of 87%. As mentioned in the model validation section, the threshold values were varied between 2 and 10, corresponding to the possible ratings. A threshold value needs to be chosen in order to determine if a wine should be recommended or not based on the predicted rating. The results from figure 6 suggest that a threshold value of 9 should be used, which has a TPR of about 82% and a FPR of about 25%.

V. Conclusion and future work

The experimental results show that the Collaborative Filter has capture the latent features that predict user ratings on previously unpurchased wines. This model can now be used to provide recommendations for users with a purchase history of 5 or more wines.

The model validation results show that the model reduces predictive error as the number of training samples increase. This indicates that the model has successfully learned how users rate their wines. However, the saturation at 2.0 RMSE suggest that the model as reached it's learning potential. It is suspected that model's learning is limited by the data. Although transforming the purchase data into ratings has successfully trained the model, more nuisance is needed. Perhaps an ensemble recommender that takes into consideration user features, such as age and gender, will improve the learning of the system. f

Although the ROC cruve shows that users there is a high probability that users will like their recommendations, the only way to truly validate a recommendation system is to implement an online A/B test on users to determine if the recommendations are leading to significantly more purchases.

The cold start problem has not been addressed here. However a possible solution is to provide users with a short list of questions about previous purchases and wine preferences.

All work on Jupyter Notebooks displaying EDA and model testing can be found on my github repository at https://github.com/AlexanderPhysics/Wine$_{Recommender}$

Acknowledgment

The author would like to thank GalvanizeU for connecting him with Wine.com and Wine.com for sharing the last 10 years of their data. Lastly, the author wants to give a special thanks to his invaluable data science instructor and mentor Jared Thomson, PhD.

References

[1] P. O. Hoyer, *Non-negative Matrix Factorization with Sparseness Constraints*, University of Helsinki:2004.

[2] S. Rendle, *Factorization Machines*, The Institute of Scientific and Industrial Research,Osaka University, Japan:2012

[3] Spark, *Collaborative Filtering*, http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html

[4] Paolo Cremonesi et.al., *An evaluation Methodology for Collaborative Recommender Systems*, http://www.contentwise.tv/files/An-evaluation-Methodology-for-Collaborative-Recommender-Systems.pdf