# Collaborative Filtering For Recommendations

Alexander Barriga and Jared Thompson†

***Abstract*—A Recommendation System for Wine.com was built using Spark's MLlib implementation of matrix factorization with alternating least squares. A novel transform was used to map wine purchase frequency to a customer interest rating. The model was trained on a controlled subset of data provided by Wine.com and is validated using stratified leave-one-out cross validation. The collaborative filter appears to successfully learn and predict user wine preferences.**

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

## I. Introduction

E-commerce sites have long faced the same problem: users become overwhelmed by the vast number of choices when shopping for a product. Sites that didn't address this issue required that the user spend an undesirable amount of time searching through a large number of alternative choices, risking that the user leave their site for a competitor capable of providing a better shopping experience.

Recommendation systems provide e-commerce sites with a competitive edge because it addresses the above issue for the benefit of the user and the business alike. A recommendation system 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 populated with products that have a high probability of being purchased.

The 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. Companies like Amazon and Netflix have created an incredibly disruptive and successful business model centered around recommendation systems.

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

A Collaborative Filter takes data that relates user's ratings to products and learns how to predict a user s rating on a product that they have not purchased yet. This is done by finding user's 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.

Corresponding author: jared.thompson@galvanize.com

Although this approach can be very effective, there are two issues that must be addressed. First, rating data is almost always sparse: users dont often rate the product they purchase. Collaborative filtering overcomes this problem by using matrix factorization. The missing values are imputed by learning a small set of latent factors. These latent factors describe hidden relationships between products and users that are not obvious from simple inspection of the data. These factors are then later used to predict user preferences.

Second, there is the "cold start" problem: it is only reasonable to provide predictions for users that have previously rated a product, thus approximations have to be made for new users for a prediction to be made. Additionally, prediction robustness is dependent on sufficient user participation. In Wine.com's case, only 2.5% of users have rated a given bottle of wine.

The paucity of rating data was addressed in this work by using more abundant purchase data as a proxy for official rating data. A novel transform was used to convert relative purchase frequency per user into a score, which was then discretized into categorical values ranging from 1 to 10, providing a pseudo-rating matrix $R$.

There are many choices to consider when deciding on an approach to build a collaborative filter recommendation system. Several high-performance, 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 recommender engines. FMs scale linearly with rows, find latent features, handle sparsity well, and have the ability to include higher-order correlations among their features. In this work, we chose to work with low-rank matrix factorization (LRMF), optimized by a high-performance variant of alternating least squares (ALS).

## II. Theory

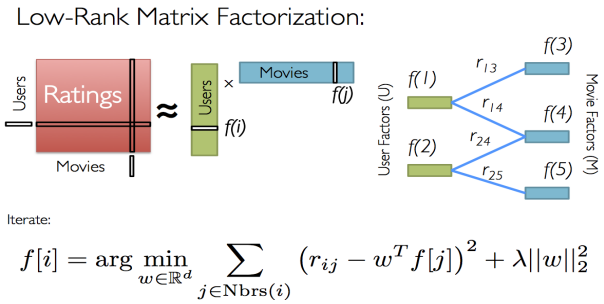


Figure 1: Spark's ALS

There are many implementations of Collaborative Filtering. An increasingly popular choice is to use the implementation from Spark's 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. Spark's scalability across computers, robust protection against data loss, and fast computation time makes it is a great choice for big data analytics and production models. Spark uses LRMF with ALS (figure 2) in order to provide high performance and robustness over large datasets.

Sparks LRMF caches two copies of $R$ in memory: one partitioned by rows and one partitioned by columns. Spark LRMF also keeps the $U$ and $V$ matrices partitioned in corresponding ways, and operates using preaddressed blocks to lower communication between processors.

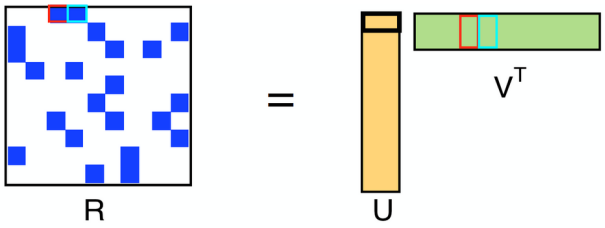


Figure 2: ALS cost function

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

## III. Methodology

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

### A. Data Preparation

All non-red or -white wine products were removed from the purchase data.The motivations for a user purchasing wines are, presumably, different from the motivations for purchasing whisky or champagne. Customers with less than 5 purchases within this category or that made only one purchase were removed, ensuring a minimum statistically sound sample of recurring users. Users that only purchase in bulk (likely wedding or party planners) were also removed. These bulk purchases rarely reflect deep wine preferences of individuals.

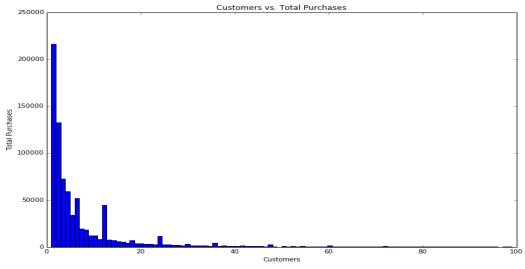


Figure 3: Number of purchases per user

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 three or less bottles in those ten 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.

The approach taken in this work was to extract a relative rating for each wine based on the user's frequency of purchases for red and white wines. A novel formula was used, approximating a metric similar to a z-score for relative wine preference:

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 to be validated is fitted on a training set $(x_{train}, y_{train})$, then its performance is tested on the previously unseen holdout set $(x_{test}, y_{test})$.

In the present work, hold-out validation is studied 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 ten categories of pseudo-ratings 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 curves, the non-binary predicted ratings of the model must 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 $\geq$ t then 1 else if r $>$ t then 0 ). Those 0 and 1 binary classifications enable us to build a confusion matrix that can be used to obtain true positive

and false positive rates. A more detailed explanation of this procedure can be found in *An Evaluation Methodology for Collaborative Recommender Systems*. [4]

## IV. Experimental results

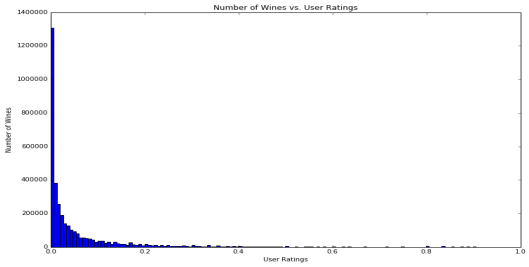


Figure 4: Purchases transformed into ratings

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.

The floating point pseudo-rating data was binned into categorical ratings ranging from 1 to 10. This was 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 (AUC) from figure 4.This process results in a balanced set of labels.

The results from model validation were encouraging. Figure 5 shows that, as the size of the training set increases, the error of the test set monotonically approaches the error of the training set. These results were obtained by setting the rank of the collaborative filter to 16.

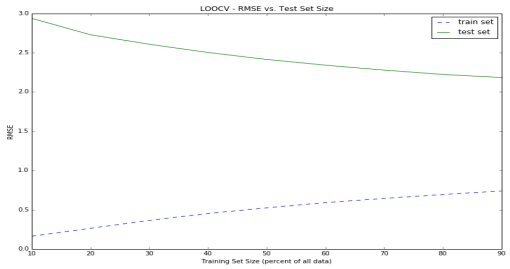


Figure 5: Model Error as Test Size Increases

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.

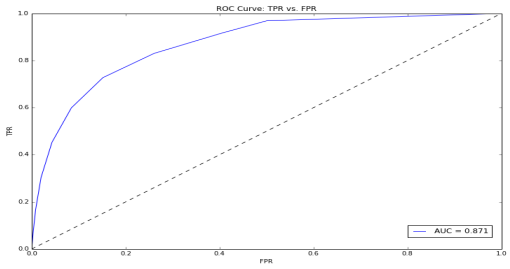


Figure 6: ROC Curve produced with all data

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 model derived in this work can function as a prototype that can now be used to provide recommendations for users with a purchase history of five or more wines with good confidence.

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 suggests that the model has reached its learning potential. It is likely that model's efficacy is limited by the availability of purchase data; additional purchase history will likely lead to improved accuracy.

Although the ROC curve indicates 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 test on users to determine if the recommendations produced by the model lead to significantly more purchases.

The cold start problem has not been addressed here. However one possible solution is to provide users with a short list of questions about previous purchases and wine preferences. This small survey could then be used to populate an initial factor vector for that user based on the item vectors of similar wines.

All work on Jupyter Notebooks displaying EDA and model testing can be found on a github repository at https://github.com/AlexanderPhysics/Wine_Recommender

## Acknowledgment

The author would like to thank the authors of Spark and Wine.com for sharing the last 10 years of their data.

## 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