Recommendation System and Retail Trading

Yueyang Liu, SUNetID yueyl | Finance & Commerce

1 Introduction

"Attention is a scarce cognitive resource" − Daniel Kahneman. Literature on attention and its asset pricing implications suggests that people pay attention to stocks that appear in the media and people tend to buy attention-grabbing stocks. Stock recommendations from the TV show Mad Money and Wall $treet Week have proven to catch people's attention and create price pressure on the stocks recommended, triggering a positive overnight excess return followed by price reversal in 2-5 days as was found in Beltz and Jennings (1997) and Engelberg, Sasseville, and Williams (2012). This is the so-called "price pressure hypothesis." Barber and Odean (2008) shows that since investors face a formidable search problem in buying stocks but not in selling stocks, people buy stocks that are recommended in the media while there isn't a huge media effect on negative stock recommendations.

Since attention is limited, a natural question to ask is what is the list of stocks that an arbitrary investor pays attention to. We aim to uncover the attention span of individual traders by analyzing the historical trading data of household investment accounts. We conjecture that due to limited attention, individual traders will trade stocks on their attention list most of the time, though the list may be verge large. Our study of trading behavior will benefit policy making in detecting illegal trading. In the meanwhile, we may give new directions in exploring asset price movements.

2 Data

The dataset is the same dataset used in Barber and Odean (2008) and related research in the field of behavioral finance, provided by a large stock brokerage firm. The data is a collection of over 3M historical trading records of all accounts opened by 78,000 US households over the period 1991 −1996. Each trading record consists of account, date, buy/sell, security number, quantity and price. After removing incomplete trading records, the final data we use in our analysis consists of over 3M trading records of over 100K accounts. We performed analysis on accounts as we believe that each account serves a different purpose. Even for accounts opened by the same households, an 401k account and an brokerage account are most likely managed differently. In the rest of the paper, we refer to a single account as an individual trader.

Let N denote the number of accounts, M denote the number of securities, and K_{ij} the number of trading records of account i on security j. We represent each trading record as follows: $X_{ij}^{(k)}$ is the $k-$th trading record of account i on security j, with $i \in \{1, ..., N\}$, $j \in \{1, ...M\}$ and $k \in \{1, ..., K_{ij}\}$. A security is either common stock or mutual fund.

3 Problem and Feature

3.1 Reformulation of the Problem

To simplify the setting, we first ignore all selling because most individual traders don't short-sell and attention doesn't play a role when they only sell the stocks they already own. We also assume that trading preferences don't change over a short period of time of only 5 years, so we can infer an individual trader's general preference from a collection of their trading activities and predict their unobserved trading behaviors. Now each trader-security pair corresponds to a binary variable, with positive values indicating that the security is on the trader's attention list, and negative values indicating that the security is not on the trader's attention list. However, our dataset contains only information on positive trader-security pairs, because we can infer from a trader buying a security that the corresponding security is most likely on the trader's attention list, yet we cannot infer the opposite when a trader does not trade a target security. This increases the difficulty of testing any algorithm predicting binary values against our dataset.

To solve the technical difficulty, we reformulate the problem as a slightly more complicated problem similar to developing a generic recommender system: Given a list of records $\{X_{ij}^{(k)}\}$, we would like to uncover each individual trader i's rating R_{ij} of security j, with $R_{ij} \in [1, 5]$. We interpret an extreme rating of 5 as "the security j is the only security that trader i is likely to trade except on rare occasions," and the other extreme rating of 1 as "the security j is not on the trader i's attention list thus the trader is unlikely to trade the security unless in rare and interesting occasions." The rating we observe in a dataset is called the observed rating, which is the sum of the actual rating and an error, e.g.,

$$R_{ij}^{observed} = R_{ij} + \epsilon_{ij},$$

which we will discuss in more detail in data preprocessing. So we answer a slightly more informative question than merely determining the attention list of each individual trader. And in the meanwhile, we can apply the techniques in constructing recommender systems to our problem.

3.2 Data Preprocessing and Feature

Below is a list of steps to preprocess the data:

- Remove records with incomplete information and selling records as we have discussed.
- Divide the data randomly into training set (60%), development set (20%), and test set (20%). We put the test set aside for now. Although our data consists of more than 3 million records, we have as many as 126,399 accounts, so there's an average of 24.01 records per account. With limited data, we use the 60%/ 20%/ 20% split. Note that we will use k-fold cross-validation with $k = 4$.
- Calculate the observed rating $R_{ij}^{observed}$ for training set and development set separately according to the following formula:

$$R_{ij}^{observed} = 1 + 4\frac{\sum_k Q_{ij}^{(k)}}{\sum_{j,k} Q_{ij}^{(k)}},$$

  where $Q_{ij}^{(k)}$ denotes the total equity bought in the k-th transaction, $k \in \{1, 2, ...K_{ij}\}$, and the summation is defined over all values in the corresponding dataset, whether it is the training set or dev set. We only include ratings greater than 1 in the training set or dev set respectively, so now each rating R_{ij} in either training set or dev set is a real number in the range (1, 5].

4 Methods

4.1 Baseline Algorithm

The baseline algorithm we consider is a naive algorithm to recover the rating of traders. Let R_{ij}^t denote an observed rating in training set, the algorithm gives estimated rating $\hat{R}_{ij}$:

$$\hat{R}_{ij} = R_{ij}^t = 1 + 4\frac{\sum_k Q_{ij}^{(k)}}{\sum_{j,k} Q_{ij}^{(k)}}.$$

In words, the estimated rating is defined to be the observed rating. And the estimated rating is 1 if trader i never trades j in training set.

4.2 SVD Algorithm

There are several algorithms in constructing recommendation systems using dimension reduction, and the SVD algorithm is one of those. The estimated rating is given by:

$$\hat{R}_{ij} = \mu + \alpha_i + \beta_j + u_j^T v_i,$$

and we minimize a squared loss with regularization

$$\sum(\hat{R}_{ij} - R_{ij}^t)^2 + \lambda(\alpha_i^2 + \beta_j^2 + ||v_i||^2 + ||u_j||^2)$$

using gradient descent.

4.3 Slope One Collaborative Filtering

The Slope One algorithms models the ratings as follows:

$$\hat{R}_{ij} = \mu_i + \frac{1}{|R_j(i)|} \sum_{k \in R_j(i)} d(j,k),$$

where $R_j(i)$ is a collection of securities rated by i that has common buyer(s) with j, and $d(j,k)$ is the average difference in ratings.

4.4 Co-clustering

The co-clustering algorithm simultaneously cluster both traders and securities and gives predictions as follows:

$$\hat{R}_{ij} = \bar{C_{ij}} + \mu_i - \bar{C}_i + \mu_j - \bar{C}_j,$$

where $C\bar{(ij)}$ is the average rating within the co-cluster, and $\bar{C}_i$ is the average rating in the trader's cluster, while $\bar{C}_j$ is the average rating in the security's cluster.

4.5 K Nearest Neighbors

K-nearest neighbors algorithms calculates the similarity score between individual investors and between securities, and predicts the trading behavior accordingly. We present the formula for user-based algorithm:

$$\hat{R}_{ij} = \frac{\sum_x sim(x,i) R_{xj}}{\sum_x sim(x,i)},$$

where similarity is defined to be the correlation, mean squared difference, or the cosine similarity.

5 Results

We have implemented the baseline algorithm, the SVD algorithm, co-clustering, and Slope One in estimating the security ratings.

5.1 Metrics to Evaluate Performance

We use the following four metrics to evaluate the performance of each model on a target dataset (development set). The first metric is the mean squared error:

$$E_1 = \frac{1}{S} \sum(\hat{R}_{ij} - R_{ij}^d)^2,$$

where S is the total number of records in the target dataset (development set), R_{ij}^d the observed rating on the target set (dev set), and the summation is taken over all records in the target set (dev set).

The second metric is the average absolute deviation

$$E_2 = \frac{1}{S} \sum |\hat{R}_{ij} - R^d_{ij}|,$$

where S is the total number of records in the target dataset (development set), R^d_{ij} the observed rating on the target set (dev set), and the summation is taken over all records in the target set (dev set).

The third metric is the sample Pearson correlation:

$$E_3 = corr(\hat{R}, R^d),$$

where $\hat{R}$ and R^d are ratings of length S.

The fourth metric is the proportion of off-list securities bought:

$$E_4 = \frac{1}{S} \sum \mathbb{1}_{\{R_{ij}=1\}},$$

where the summation again is taken over all records in the target set. We are interested in the third metric as well, since it measures the rate at which the traders buy securities not on their attention list. We conjecture that the rate should be relatively small.

5.2 Results from Experiments

We first plot the number of records per account in a histogram. We observe that most of the accounts trades less frequently:

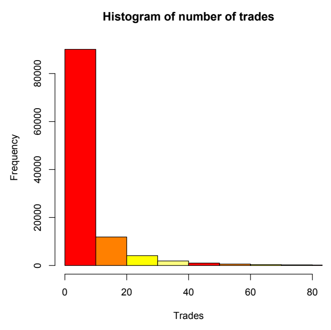


Figure 1: Data Visualization

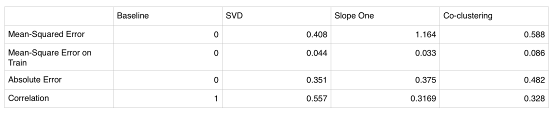
We then pre-define a set of cutoffs k range between 0 and 20, and test each algorithm on data with accounts with more than k trading records. A snapshot of how the performance of algorithm varies with k is shown below:

Mean-Squared Error / Cutoffs	0	5	10	15	20
Dev Error	4.82	1.93	0.71	0.408	0.288

Figure 2: Optimal Cutoff

We picked $k = 15$ to be the optimal k since the error is small enough. Note that the optimal k actually varies with models and even parameters of the models, yet we use a single value for k for simplicity.

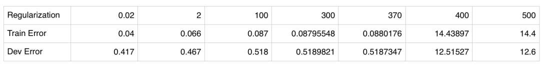
We next run each algorithm and performs error analysis, as shown on the table.



	Baseline	SVD	Slope One	Co-clustering
Mean-Squared Error	0	0.408	1.164	0.588
Mean-Square Error on Train	0	0.044	0.033	0.086
Absolute Error	0	0.351	0.375	0.482
Correlation	1	0.557	0.3169	0.328

Figure 3: Error Analysis I

In order to alleviate the concerns on high variance and overfitting, we tried regularization with a large set of different parameters and concluded that regularization wouldn't improve the performance. Below we take SVD algorithm as an example. We observe that as the parameter for regularization increases, the gap between training error and dev error shrinks, yet the dev error always increases as the regularization parameter increases.



Regularization	0.02	2	100	300	370	400	500
Train Error	0.04	0.066	0.087	0.08795548	0.0880176	14.43897	14.4
Dev Error	0.417	0.467	0.518	0.5189821	0.5187347	12.51527	12.6

Figure 4: Error Analysis II

We concluded that the SVD performs the best, and re-run the model on training set and dev set, and predict on test set. The mean squared error is 0.546. The predictions is shown below:

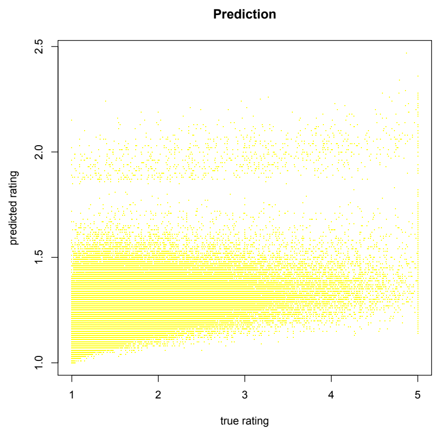


Figure 5: Prediction

6 Discussion and Future Work

Our main contributions include reformulating the problem of uncovering attention lists of retail traders into a problem of trading volume prediction, and applying algorithms in constructing recommendation systems to tackle the problem. We then removed accounts with number of records below a certain threshold such that the data represent the correct attention distribution. In order to alleviate the concerns of high variance, we tried a set of possible values for regularization with no luck. We conclude that regularization wouldn't improve performance. We've also shown that the recommendation algorithms perform better than the baseline algorithm in predicting trades.

To improve the algorithms and performance, we plan to construct models that extracts information from accounts with less than a certain number of trading records. We will also try neural networks in making predictions. The future directions of our research include detecting illegal trading based on unexpected trades, and predict stock price movements. We can make predictions based on either unexpected trades as it may contain information as a result of illegal trading, or based on anticipated trades as those are made by traders with more experience trading the security in question.

References

[1] Jess Beltz and Robert Jennings, (1997), "Wall street week with Louis Rukheyser" recommendations:Trading activity and performance, Review of Financial Economics, 6, (1), 15-27

[2] Joseph Engelberg; Caroline Sasseville and Jared Williams, (2012), Market Madness? The Case of Mad Money, Management Science, 58, (2), 351-364

[3] Barber, Brad M., and Terrance Odean, 2008, All that glitters: The effect of attention and news on the buying behavior of individual and institutional investors, Review of Financial Studies 21, 785, 818

[4] Online Notes on Netflix Prize, Simon Funk, 2006. http://sifter.org/ simon/journal/20061211.html

[5] Package Documentation of surprise, Nicolas Hug, 2015.

Acknowledgements

We would like to thank Professor Terrance Odean for sharing the data and discussing the behavioral finance aspects with us to make our project possible. We would also like to thank Professor Dan Boneh and Professor Andrew Ng for the wonderful lectures, Professor Dan Boneh , Ziang Xie, Ben Zhang, Lucio Dery, and an TA stopped by at the poster session for advice on the project, and the rest of the course stuff for making CS 229 a great learning experience.