UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

One-class Support Vector Machines Approach for Trust-Aware Recommendation Systems

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the Degree

of

MASTER OF SCIENCE

By

THOMAS WELBORN

Norman, Oklahoma 2021

One-class Support Vector Machines Approach for Trust-Aware Recommendation Systems

A THESIS APPROVED FOR THE

GALLOGLY COLLEGE OF ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Talayeh Razzaghi, Chair

Dr. Charles D. Nicholson

Dr. Dean F. Hougen

Abstract

As the amount of information users interact with every day continue to grow, filtering it for useful information is increasingly important. One of the most useful tools for this task are recommender systems (RS). These look at past products the user has interacted with and recommends similar products. However, these suffer from a major issue, cold-start, in which there is difficulty in producing recommendations for new users. One of the suggested techniques for mitigating the cold-start issue is the use of trust data. By using the relationships between users such as friendships on social media or following reviewers of movies the recommender system can recommend products that the user's friend would rate highly as well.

We extend previous trust models by applying a One-Class Support Vector Machine model to the known trust relations and predicting distrust relations among users. This is shown to improve the predictions movie ratings in some circumstances.

Table of Contents

Chapter 1: Introduction

Recommender systems suggest content to users that they are likely to rate highly. They are built on data gathered from previous interactions with a user and suggest content similar to past content. Recommender systems can also use data on other users to find similar users and recommend content popular with those users. These two approaches are called content-based filtering and collaborative filtering, respectively. Recommender systems provide better results as more data are gathered on both the website overall as well as on the specific user. This leaves a gap in the ability to suggest content to new users, on wyhom little data has been collected. This is referred to as the cold-start problem and is our focus in this paper.

Motivation

While many consumers may not realize it, recommender systems have become a common part of everyday life. A large portion of the interactions between consumers and modern technology involve content shown by a recommender system. One example is Amazon.com, which offers products to users based on past products purchased, or even simply viewed [1]. Netflix recommends movies similar to those that the users have already watched, especially those which a user has "thumbed up" (the thumb system replaces the previous one-to-five-star rating system). The company gained a large amount of attention with regards to recommender systems when they held a contest in which participants could create a recommender system to predict what ratings users would give to various movie titles in the Netflix catalogue [2]. By creating more powerful recommender

systems, owners of large catalogues can more accurately suggest content to users which results in a more streamlined user experience and more user engagement with the system. There are numerous challenges in making these predictions which, if addressed, can lead to more accurate content suggestions.

Problem Definition

Recent advances have begun to leverage the large amounts of data which modern sensors and websites are able to collect. These models can improve the predictive power as the amount of data provided to them increases. This has the potential to help users navigate the increasingly large catalogues of content and products offered by modern services. However, recommender systems do not always produce relevant recommendations.

One of the most prevalent issues in creating recommender systems is the "cold-start" problem, which results in poor recommendation performance [3]. The cold-start problem refers to the situation in which there is very little information on a new user, such as when a customer of an online retailer has only made a single purchase. One proposed solution to this problem is to use the concept of "trust" in the recommendation systems. Many social websites allow users to be connected to or follow other users. These users presumably have similar interests, so they can build trust relationship with other users. By including this data in the model, cold-start users, if they trust any users who have rated products, can be assumed to behave similar to more established users. Trust data could prove beneficial in both improving the accuracy of a model overall, as well as improving predictions for new users on which very little is known, and therefore improve their retention rates.

Ratings data, especially for use cases with many items, tend to be incredibly sparse. This is because each individual user is unlikely to review a large portion of the items available. For example, Netflix has a catalogue that is too large for the average consumer to watch most of it. Additionally, if a user consumes content or makes a purchase, they are not obligated to leave a review, which leads to even greater data sparsity. In order to overcome these challenges, several methods have been developed, including trust-aware recommender systems.

Objectives

In this study, we aim to identify sparse trust relationships among users using a predictive model. We do this rather than determining them directly by the human expert which is a tedious task and may be subjective to error or bias. We will demonstrate the use of one-class support vector machines (OC-SVM) in the identification of sparse trust data which represents the trust relationships between users. We hypothesize that the use of OC-SVM model for trust mining would improve the performance of recommender systems which predict the ratings of movies by users. In addition, because only positive relationship (trust rather than distrust) data are provided, OC-SVM can be used effectively to identify the trust relationships with similar sparse attributes to those with known social trust, so the trust relationships will be labeled as inliers, while distrust relationships will be flagged as outliers.

Research Contributions

Our work is inspired by Hu et al. (2020) which proposed a semi-supervised learning–based sparse trust recommendation (SSL-SVD) method [4]. Instead, our

method employs an unsupervised algorithm to predict sparse trust and improve recommendations. We create a model which can generate sparse trust data more easily. Our contribution is to employ OC-SVM to automatically learn the sparse trust rather than by using cutoffs (or thresholds) set by human experts, as in SSL-SVD [4]. Setting these thresholds is not easy and might be prone to bias or error. To our knowledge, there is no study that applies an outlier-detection algorithm in order to predict sparse trust among users. Our trust-aware recommendation system is equipped with an automated trust mining model using OC-SVM, an improvement over the SSL-SVD model which predicted trust using expert-guided manual cutoffs for the values of the similarity measures.

This thesis is structured as follows: in Chapter 2, we will discuss the state-of-the-art recommender system methods, particularly for trust-based recommendation systems. Furthermore, we will introduce four sparse trust relationship measures used in our model. In Chapter 3, the OC-SVM and trust-aware recommendation system methodologies are described along with the performance evaluation measures. In Chapter 4 we will describe how we carried out our experiments and compare our results to other methods. In Chapter 5 we will conclude and summarize our work and recommend future research directions.

Chapter 2: Literature Review

Matrix Factorization

Matrix factorization (MF) is a broad category of mathematical techniques which allows a large matrix, for example a matrix of size *users* by *items*, to be represented

by two much smaller matrices of sizes *users* by *r* and *r* by *items*, where *r* is smaller than either *users* or *items*. The interpretation of these new matrices depends on the technique used to perform the factorization. These new matrices can create denser representations of otherwise sparse data such as ratings matrices [5]. Early recommender systems often used a form matrix factorization to make predictions.

This new representation is known as a latent factor (LF) representation of the data. The latent factors represent underlying characteristics in the items represented. In the case of movies, the LFs could have a broad range of interpretations. For example, LFs could represent genre, movies with a strong female lead, a slow or fast soundtrack, or any other characteristic of the movie, including a combination of these. The LFs are not set manually but are instead the result of factoring a matrix into two or three (depending on the type of factorization) matrices. The interpretation of these latent variables is a topic of much study, but their interpretation is not important to the task of suggesting new content to users.

SVD++

SVD++ is a model proposed by Koren [6] which is built on the mathematical concept of singular value decomposition (SVD). This technique has been used for decades to factor a single large matrix into two long, thin matrices as well as a central diagonal matrix with the dimensions of the original matrix. This can decrease the sparsity of the data. One improvement is the use of implicit data. For this model, that consists of combining the ratings data with data on which movies the users rated. This is achieved by one-hot-encoding the data as either rated or not rated, 1 or 0. Koren then uses a neighborhood-based model which considers

whether other users rated a given movie higher or lower than their average and adjusts the predicted rating accordingly. The final equation used as the prediction function for SVD++ is given below.

$$\widehat{r_{ui}} = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j\right) \quad \text{Equation 1}$$

$$+|R^k(i;u)|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |N^k(i;u)|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij}$$

Equation 1 is described as having three parts. The first is $\mu + b_u + b_i$, where μ is the overall mean of all movies, b_u is the tendency of the user to rate higher or lower than the mean, and b_i is the same for the given movie. The next portion is $q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j\right)$, where q_i^T is the items-factors vector, p_u is the items-factors vector, and y_j is a factor vector. $N(u)$ is the number of items where the user provided implicit preference. The remaining portion is the neighborhood portion of the model. This is where for each of the nearest *k* neighbors, the model considers the difference between the users rating and their bias for the movie. This difference is weighted by w_{ij}, which is theorized to weight movies that are similar to movie *m*. c_{ij} also is theorized to be higher in items *j* that are similar to *i* for all movies user *u* has provided implicit feedback on.

While the use of matrix factorization allows for the creation of recommender systems, it struggles in creating predictions for new users; this issue is known as the cold-start problem. In the next section, trust-aware recommender systems are introduced in order to address the issue of cold-start.

Trust-aware Recommender Systems

Trust-aware recommender systems (Trust-aware RS) were originally proposed by Massa and Avesani in 2007 in order to overcome the cold-start problem [7]. The recommender system they presented takes advantage of additional data collected on the trust between users. If users trust or "friend" other users, then those users are assumed to have similar tastes. This helps resolve the cold-start problem as those users who have limited reviews can be recommended products that are rated highly by the users they trust.

Trust data are not always available. Many services simply have not created a system that allows users to suggest trust relations. When such a system does exist, it is not always able to gather large amounts of trust data for each user. However, trust data are still quite useful as the trust-aware models are an extension on previous, ratings-only models. Therefore, when a user does not have any trust relations the recommender can fall back on traditional ratings-based recommendations.

Trust-SVD

Based on singular value decomposition (SVD), Trust-SVD was proposed as an algorithm which explores the concept of implicit trust. Implicit trust is mined by comparing the rating patterns of one user to another. This is contrasted with explicit trust which is obtained from users actively trusting another user. Implicit trust is less reliable than explicit trust, but it has the advantage of being more abundant, as it does not require additional action by the users, it is simply calculated from the ratings data. The final loss function that Trust-SVD attempts to minimize is given in Equation 2 [8].

$$loss = \frac{1}{2}\sum_u \sum_{j \in I_u} \left(\hat{r}_{u,j} - r_{u,j}\right)^2 + \frac{\lambda_t}{2}\sum_u \sum_{v \in T_u} \left(\hat{t}_{u,v} - t_{u,v}\right)^2 + \quad \text{Equation 2}$$

$$\frac{\Lambda}{2}\sum_j \left|U_j\right|^{-\frac{1}{2}} b_j^2 + \sum_u \left(\frac{\lambda}{2}|I_u|^{-\frac{1}{2}} + \frac{\lambda_t}{2}|T_u|^{-\frac{1}{2}}\right)\|p_u\|_F^2 + \frac{\lambda}{2}\sum_j \left|U_j\right|^{-\frac{1}{2}} \left\|q_j\right\|_F^2 +$$

$$\sum_j |U_i|^{-\frac{1}{2}} \left\|y_j\right\|_F^2 + \frac{\lambda}{2}|T_v^+|^{-\frac{1}{2}}\|w_v\|_F^2 + \frac{\lambda}{2}\sum_u |I_u|^{-\frac{1}{2}} b_u^2$$

In Equation 2, $\hat{r}_{u,j} - r_{u,j}$ and $\hat{t}_{u,v} - t_{u,v}$ are the errors in rating prediction for every user-movie combination and the trust error for every user-user combination. Λ and λ_t are regularization parameters which adjust the level of a role which trust plays vs the latent model. $|I_u|^{-\frac{1}{2}} b_u^2$ regulates the size of bias terms for each user while $\left|U_j\right|^{-\frac{1}{2}} b_j^2$ plays the same role for movies. Each of the following terms controls for model complexity as well. $\|\,.\,\|_F$ represents the Frobenius norm and grows as the members of the vector grow; therefore, it is useful in measuring complexity. T_u is the set of users trusted by u, U is the set of users who rated the item represented by the subscript while I is the set of items rated by the user in the subscript. The variables b, p, q, y, and w are latent matrices which are trained by gradient descent. This is hypothesized to create a model which accurately predicts ratings and trusts while minimizing the size of parameters to limit overfitting.

RoleTS

A model based on "Role-based Trust Strength" (RoleTS) [9] is an extension of Trust-SVD. The authors improve the predictive power of Trust-SVD by separating trust into two "roles." This is done by adding a second trust matrix T^- which represents the trustors in addition to the original trust matrix T which represents the trustees.

Pan et al. (2020) also improved the model by adding two regularizations. This is accomplished by the following Equations 3 and 4.

$$\frac{\beta}{2}\sum_u \sum_{v\in T_u} g\left(X_{u,v}\right)\|P_u - W_v\|_F^2 + \frac{\lambda}{2}\sum_u \sum_{v\in T_u} X_{u,v}^2 + \quad \text{Equation 3}$$

$$\frac{\gamma_M}{2}\sum_u \sum_{v\in T_u}(X_{u,v} - g(w_m P_u W_v^T + b_m))^2 +$$

$$\frac{\gamma_S}{2}\sum_u \sum_{v\in T_u}(X_{u,v} - g(w_s Sim(u,v) + b_s))^2$$

$$g\left(X_{u,v}\right) = \frac{1}{1+e^{-X_{u,v}}} \quad \text{Equation 4}$$

β is a hyperparameter that would balance the importance of trust for recommendations. $X_{u,v}$ is the latent trust between users u and v. $P_u - W_v$ is the difference between the latent trustor and trustee vectors. γ_M and γ_S are hyperparameters which weight the importance of each of the two parts of the regularization. The second term attempts to limit the values of $P_u W_v^T$ which gives back the latent trust matrix $X_{u,v}$. The final term utilizes the Pearson correlation coefficient to train the parameter w_s. This recommender achieves improved results over Trust-SVD; however, it is outperformed by the algorithm described in the following section.

SSL-SVD

A model based on semi-supervised learning is SSL-SVD [4], which extends other trust models by increasing the density of trust by using "sparse trust." Sparse trust is specific type of implicit trust. Sparse trust is considered weaker than social trust, however, any added trust relations discovered have the potential to improve predictive power. This type of trust is also mutually exclusive to social trust; It

displaces relations that would be considered unknown or distrust in other models. The authors proport to have increased the trust density of each dataset by at least sixty-five percent.

Hu et al. (2020) have introduced four sparse trust relationship measures which can be calculated from the ratings dataset. These measures are similarity, consistency, credibility, and objectivity, which are granular representations of the preferences and behavior of users. Each measure is described below:

$$Similarity(u,v): \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \quad \text{Equation 5}$$

$$Consistency(u,v): \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})}{\sqrt{\sum i \in I_{u,v} (r_{u,i} - \overline{r_u})^2} \sqrt{\sum i \in I_{u,v} (r_{v,i} - \overline{r_v})^2}} \quad \text{Equation 6}$$

$$Credibility(v) := l\left(C_1 + \frac{1}{C_2}\right), \quad \text{Equation 7}$$

$$C_1 = \frac{\sum_{i \in I_v} \left(r_{v,i} - \overline{r_v}\right)^2}{N}, C_2 = \frac{\sum_{i \in I_v} \left(r_{v,i} - \overline{r_i}\right)^2}{N}, \quad \text{Equations 8, 9}$$

$$l = \begin{cases} \frac{N}{N^u} & N < N^u \\ 1 \; otherwise \end{cases} \quad \text{Equation 10}$$

$$Objectivity(v): \frac{\sum_{i \in I_v} (r_{v,i} - \overline{r_v})(r_i - \bar{r})}{\sqrt{\sum i \in I_v (r_{v,i} - \overline{r_v})^2} \sqrt{\sum i \in I_{u,v} (\overline{r_i} - \bar{r})^2}} \quad \text{Equation 11}$$

In Equations 5-11, I_u and I_v are the sets of movies users u and v have rated, respectively. $r_{u,i}$ and $r_{v,i}$ are the rating of item i by user u and v respectively. $\overline{r_u}$ and $\overline{r_v}$ is the mean ratings of user u and v respectively, $\overline{r_i}$ is the average rating on movie i, and $\bar{r}$ is the average rating across all ratings. N is the number of items

rated by user u and N^u is the minimum number of items a user needs to rate to be considered fully "lively."

Similarity is the ratio of movies rated by both users to the number of movies rated, by either user; it does not consider the ratings given. It is hypothesized that users who have a high similarity would be more likely to trust each other. Consistency is used to find users who have similar "grade inflations" which is the idea that some users may rate a liked movie 2-stars and a disliked movie 4-stars, being reluctant to give out 1- or 5- star ratings; while other may more freely give 1-star and 5-star rating to movies they find equally as good or bad as the first user. Credibility is predicted to be low in users who are being dishonest. This is predicted by finding users who rate many items significantly different than their own mean rating and different than the items mean rating, a behavior that is not expected to be found in honest users. It is also higher in users who have rated at least a minimum number of items. Objectivity is similar to credibility; however, it does not take into account how many items a user has rated, their liveliness, and it is not based on volatility, or whether a user rates items similar to their own mean. We adopt these same measures in the model used for this experiment. After the measures are calculated for each user-user pair they are used to predict the sparse trust between those two users.

Hu et al. (2020) then used each of these measures to find distrust among users by applying the transductive support vector machine (TSVM) to the data. However, before applying the TSVM the authors label relations below a certain threshold as

distrust. These predictions are then combined with the know social trust relations for use in predicting ratings.

$$loss = \frac{1}{2}\sum_u \sum_{i \in I_u} (\hat{r}_{u,i} - r_{u,i})^2 + \frac{\lambda_t}{2}\sum_u \sum_{v \in T_u} (\hat{t}_{u,v} - t_{u,v})^2 \qquad \text{Equation 12}$$

$$+ \frac{\lambda_s}{2}\sum_u \sum_{z \in S_u} (\hat{s}_{u,z} - s_{u,z})^2 + \frac{\lambda}{2}\sum_u |I_u|^{-\frac{1}{2}} b_u^2 + \frac{\lambda}{2}\sum_i |U_i|^{-\frac{1}{2}} b_i^2$$

$$+ \sum_u \left(\frac{\lambda}{2} |I_u|^{-\frac{1}{2}} + \frac{\lambda_t \alpha}{2} |T_u|^{-\frac{1}{2}} + \frac{\lambda_s (1-\alpha)}{2} |S_u|^{-\frac{1}{2}} \right) \|p_u\|_F^2$$

$$+ \frac{\lambda}{2}\sum_i |U_i|^{-\frac{1}{2}} \|q_i\|_F^2 + \frac{\lambda}{2}\sum_j |U_j|^{-\frac{1}{2}} \|y_j\|_F^2 + \frac{\lambda}{2} |T_v^+|^{-\frac{1}{2}} \|w_v\|_F^2 + \frac{\lambda}{2} |S_z^+|^{-\frac{1}{2}} \|f_z\|_F^2$$

Equation 12 is the loss function used by SSL-SVD. It will also be the loss function adopted by our model. The sparse trust mined by SSL-SVD will be replaced with the predicted sparse trust in our model. In equation 12, $\hat{r}_{u,i} - r_{u,i}$ represents the error in rating prediction while $\hat{t}_{u,v} - t_{u,v}$ and $\hat{s}_{u,z} - s_{u,z}$ serve the same purpose for social and sparse trust, respectively. $|I_u|^{-\frac{1}{2}} b_u^2$ and $|U_i|^{-\frac{1}{2}} b_i^2$ serve as regulatory terms on the user and item vectors, respectively. $\frac{\lambda}{2} |I_u|^{-\frac{1}{2}} + \frac{\lambda_t \alpha}{2} |T_u|^{-\frac{1}{2}} + \frac{\lambda_s (1-\alpha)}{2} |S_u|^{-\frac{1}{2}}$ provides a balance between social and sparse trust. If α is set to one, only social trust is considered, only sparse trust is considered when α is set to zero. Each of the remaining terms control for complexity. q_i is the latent item matrix, y_j represents the predicted trust values, w_v and f_z are the latent user vector of the social and spares followees of u, respectively.

While SSL-SVD was able to achieve better results than existing Trust-Aware RS, they rely on the use of expert guidance in setting the cutoffs for trust versus distrust

on each of the four sparse trust measures. In the next chapter we cover the methodology used to generate the predictions with our model.

TT-SVD

The authors of this paper utilize a two-pronged model based on two-way trust (TT-SVD) [10]. They run a separate gradient descent on using the data on both trustees and trustors. The authors then have a parameter β which controls the ratio between the two predictions as seen in Equation 13 below.

$$\hat{r}_{u,j} = \beta(TrusteeSVD) + (1 - \beta)(TrusterSVD) \quad \text{Equation 13}$$

In Equation 13 $\hat{r}_{u,j}$ is the predicted rating, β is a hyperparameter between 0 and 1, and TrusteeSVD and TrusterSVD are the rating prediction given by the trustee and truster models, respectively. This method resulted in improved results over previous methods which trained a single model using both the trustee and trustor information.

DLMF

Deep learning is a machine learning technique which has gained significant popularity recently. Deng et al. [11] apply deep learning to the problem of ratings prediction using trust data in a model they refer to as Deep Learning based Matrix Factorization (DLMF). They begin by pretraining an autoencoder in order to pick accurate starting values. This is intended to reduce the tendency of the algorithm to converge on local optima and achieve performance closer to the global optimum. The researchers used a Continuous Restricted Boltzmann Machine (CRBM) to create a latent representation of the user vectors into a lower

dimensional space. The latent vector is then decoded to attempt to reconstruct the original user matrix. The model is trained to minimize the error between the original user matrix and the reconstruction.

The authors use trust propagation to increase the trust density. This method is based on the principle that if user 1 trusts User 2, and User 2 trusts User 3 then User 1 will tend to trust User 3 as well. They used a concept of "cliques" to predict which users would be most likely to share interests with each other. A clique is a subgraph of the whole trust graph in which the users all trust one another. The algorithm then attempts to minimize the difference in the users rating on an item from that of the cliques average rating. This is achieved by minimizing the regularization term given below.

$$\left\| P_u - \frac{1}{|N(u)|} \sum_{v \in N(u)} T_{u,v} P_v \right\|_F^2 \qquad \text{Equation 14}$$

In Equation 14, P_u is the latent matrix of user u, N(u) is the neighborhood of user u (those in the same clique) and $T_{u,v}$ is the trust value between user u and v.

Chapter 3: Methodology

Sparse Trust Mining

Before OC-SVM could be performed, the data were mined for internal trust using Equations 6-11. This gives additional features to each relationship between every user in the dataset which allows for the prediction of the trust relation between users where no explicit trust is given. These features attempt to describe the rating behavior of each user and show how they compare. This is predicated on the assumption that the characteristics of the relations between users with social trust will be similar to those who have sparse trust.

SVM

Support vector machine (SVM) is an algorithm which divides data into two or more classes. SVM relies on the concept of a separating hyperplane to select the class for a given data point. This hyperplane is a function which takes as input all the dimensions provided about a point and returns the class to which the point belongs.

However, for many datasets there would be many or no such hyperplanes that divide the data into the correct classes based on the input variables. Both have issues been addressed for SVM [12].

First, the issue of multiply separable data. When there are multiple hyperplanes that divide the data, the SVM algorithm attempts to select the maximum-margin hyperplane. This is the hyperplane where the distance between the hyperplane and the nearest points are minimized. This will theoretically reduce the number of

misclassified points that are predicted when those points lie between the known location of points in each class for the train data.

The second issue is when the data are not linearly separable with the given dimensions. There are two solutions to this issue. One is the use of soft margins, which allow a small portion of the training points to be misclassified. By allowing a few outlying points to be misclassified the distance between the hyperplane and the next nearest points can be increased. The number of such points must be limited however, or else the hyperplane will no longer separate the classes sufficiently to be useful.

When soft margins are not sufficient to make the data separable, the issue can be further treated using "kernels." Kernels are transformations applied to the dataset which create different features than linear space which can therefore lead to separable data. It has been shown that any dataset can be made separable using kernels, however, it is not always clear which kernel is correct. In most cases researchers simply try several kernels and select the one which provides the highest separability of the data [13].

Equation 15 below describes the hyperplane created by an SVM model. $\frac{1}{2}\|\omega\|_2^2$ is a term designed to limit overfitting of the model; larger coefficient values are discouraged. ξ_i is the error for training point $\vec{x}_i$; the coefficients of the equation which describes the hyperplanes is given as $\vec{\omega}$, while the bias is given by b. C is a hyperparameter which balances the penalty between errors and overfitting prevention.

$$\min_{\vec{\omega},b,\xi_i \geq 0} \quad \frac{1}{2}\|\omega\|_2^2 + C\sum_{i=1}^{m}\xi_i \quad s.t.\ \mathrm{y}_i(\vec{\omega}\cdot\vec{\mathrm{x}}_i - \mathrm{b}) + \xi_i \geq 1 \text{ Equation 15.}$$

One-Class SVM

One-class support vector machines (OC-SVM) are frequently used in the field of anomaly detection when many typical data have been collected. These models are a specific adaptation of the broader class of models described above known as SVM. The model attempts to estimate a density which contains most of the typical data. When new data are passed into the model, those which fall out of this density can be considered outliers and are often further tested to determine whether they are errors or defects. The largest benefit of OC-SVM is that it does not require any labeled data of the second class.

$$\left\{\begin{matrix} \min \frac{1}{2}\|\omega\|^2 + \frac{1}{Nv}\sum_{i=1}^{N}\xi_i - \rho \\ s.t.\left(w * \phi(x_i)\right) \geq \rho - \xi_i, i \ldots N \\ w \in Z, \xi \in \mathbb{R}_+^N, \rho \in \mathbb{R}, v \in (0,1] \end{matrix}\right\} \quad \text{Equation 16}$$

Equation 16 is used to define the boundary of the decision boundary for whether points are considered outliers [14]. $\omega\ and\ \rho$ fully define the boundary. $\frac{1}{2}\|\omega\|^2$ is used as a regularization coefficient, N is the number of training samples (in the positive class), ξ_i is used as a slack variable, v controls the number of training samples allowed outside the boundary by raising or lowering the penalty on such point. Different kernels can be applied to achieve better separability of the data while minimizing errors. This is represented by transforming the input vector using $\phi(x_i)$.

OC-SVM is applied to trust data with known trusts, but no known distrusts, in order to predict the relationships that are distrustful. This is useful as most organizations that collect data on trust, for example using friending and following, do not collect data on unfriends or other such inverses of friendship. That means that researchers examining these data must attempt to find the cases most similar to the known trust relations and use them to contrast the rest of relations where trust is unknown. The OC-SVM applied to my recommender system was from the Sci-Kit Learn package.

Performance Measures

Performance measures serve two purposes in the experiment. The first is for hyperparameter tuning of both the OC-SVM algorithm and the recommender system and the second is for evaluation of final results. The metrics used are "mean absolute error" (MAE) and "root mean square error" (RMSE), which both increase as the number and scale of difference increase between the predicted ratings and the actual ratings.

$$MAE = \sum_{i=1}^{n} \frac{|r_i - y_i|}{n} \qquad \text{Equations 17}$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(r_i - y_i)^2}{n}} \qquad \text{Equations 18}$$

Equations 17 and 18 describe how to calculate MAE and RMSE respectively. For both equations r_i represents the actual rating of a movie by a user, y_i represents the rating predicted by the recommender, and n is the number of ratings in the relevant dataset. RMSE penalizes larger errors significantly more than small errors, while MAE penalizes errors directly proportional to their size.

The OC-SVM and the recommender were tuned separately. OC-SVM is difficult to evaluate using traditional methods as all the labeled data belong to the train set; there is no test set available. To overcome this, we tuned the OC-SVM by running the entire model including the recommender and selecting the OC-SVM hyperparameters which achieved the lowest test RMSE on the ratings data.

To select hyperparameters, grid search was used. We tested all combinations of kernel, degree, and ν (not all of these parameters apply to each kernel). This is an improvement over tuning each individually as hyperparameters can interact with each other and should not be assumed to be independent of each other [15]. The recommender was then tuned in a similar way while holding the OC-SVM hyperparameters fixed. The values tuned for the recommender were the values for learning rate and α, the ratio of the weight of sparse versus social trust.

Recommender System

We applied OC-SVM to estimate trust relationship recommendation system techniques used in Hu et al. (2020) [4] including SVD++ [6], PMF [16], SoRec [17], RSTE [18], SocialMF [19], TrustMF [20], TrustSVD [3]. In addition, we have vectorized many of the components of the model in order to significantly improve the speed of the implementation. Algorithms 1 and 2 below demonstrate the pseudo-code of our model.

Pseudocode

Algorithm 1: Algorithm for prediciton of sparse trust using OC-SVM

```
Input: ratings of movies by users, trust relations between users;
Output: sparse trust predictions;
for user u in all users do
    Calculate Objectivity(u);
    Calculate Credibility(u);
    for user v in all users do
        Calculate Consistency(u, v);
        Calculate Similarity(u, v);
    end
end
Train OC-SVM(Labeled Trust Data);
Predict OC-SVM(Unlabeled Trust Data);
```

Algorithm 2: Algorithm for trust-aware recommender based on OC-SVM predictions of sparse trust.

```
Input: ratings, trusts, sparse trust predictions;
Output: predictions of all users, cold users;
for i in recommender iterations do
    for user, movie in train ratings do
        predicted rating = predict rating(user, movie);
        error = actual rating - predicted rating;
        loss += error ** 2;
        update bias matrices(error);
        update item matrix(error);
        update spar matrix(followee latent vector);
        for f in user's followees do
            loss += dot(sparse matrix, user matrix) ** 2;
            loss += dot(social matrix, user matrix) ** 2;
        end
        update social matrix based on followees;
        update user matrix based on error;
        update ratings matrix based on error;
        loss += shrinkage factor;
        if loss > previousloss then
            exit program;
        end
    end
end
```

Algorithm 1 shows the steps used to first generate the similarity measures which are then used as the features to train the OC-SVM model on the trust relations labeled as trust. It then outputs the sparse trust predictions which are used in

Algorithm 2 as inputs to produce predictions used to determine the performance of the model.

In chapters 4 and 5 we describe the results of our implementation using the performance metrics described and compare them to the results found in previous work.

Chapter 4: Results

Data

We first describe the dataset and cross validation used to test the model. Table 1 describes the characteristics of both datasets tested in this study, CiaoDVD and FilmTrust [21].

Table 1. Summary statistics of each dataset examined.

Data Set	Users	Movies	Ratings	Ratings Density	Trusts Relations	Trust Density	% Cold Users
CiaoDVD	7,375	99,746	278,483	0.0379%	111,781	0.23%	18.63
FilmTrust	1,508	2,071	35,497	1.14%	1,853	0.42%	0.08

Table 1 shows that the FilmTrust dataset has fewer ratings and users than the CiaoDVD dataset, but it has higher ratings and trust densities. Trust density is defined as the number of trust relationships divided by the possible trust relations. We describe the best hyperparameters experimentally determined, and finally we compare the model to other state-of-the-art techniques.

CiaoDVD has significantly more trust relations; however, it has a much lower percentage of cold-start users. This could explain the better performance of our model, as there are more non-cold-start users on which to train the model and improve the predictions on the cold-start users. To train the model, an 80/20 train/test split is used alongside 5-fold cross-validation.

OC-SVM Results

Below we present the results of the performance of different OC-SVM hyperparameters. We judge the optimal value for each hyperparameter by

evaluating the performance of the overall recommender. We determine the optimal hyperparameters for our model and then compare the results of our model to the state-of-the-art techniques based on RMSE and MAE.

FilmTrust

Table 2. The best performing hyperparameters for each kernel on FilmTrust dataset.

Kernel	Degree	ν	Train RMSE	Train MAE	Test RMSE	Test MAE
Polynomial	3	0.125	**0.5558**	0.5751	**0.7893**	**0.6731**
Linear	-	1	0.5555	**0.5748**	0.7925	0.6739
RBF	-	0.05	0.5554	0.5752	0.7926	0.6748

CiaoDVD

Table 3. The best performing hyperparameters for each kernel on CiaoDVD dataset.

Kernel	Degree	ν	Train RMSE	Train MAE	Test RMSE	Test MAE
Polynomial	3	0.05	**0.6842**	**0.5005**	**0.9811**	**0.7298**
Linear	-	0.01	0.6873	0.5030	0.9817	0.7306
RBF	-	0.1	0.6864	0.5018	0.9831	0.7306

Next, we use the results shown in Tables 2 and 3 which show the hyperparameter combinations that give the lowest test RMSE for each kernel tested. We use the recommender system hyperparameters suggested in the Hu et al. study [4] while adjusting the OC-SVM hyperparameters. Finally, for each dataset, the best performing of the related hyperparameters are then used by the recommendation system as shown in Table 4.

Table 4. Hyperparameters used by each method.

Approaches	Parameter Settings
PMF	$\lambda = 0.001$
SVD++	Recommended in Reference [6]
SoRec	Recommended in Reference [17]
RSTE	$\alpha = 1.0, \lambda = 0.001, \lambda_T = 1$
SocialMF	$\lambda = 0.001, \lambda_T = 1$
TrustMF	$\lambda = 0.001, \lambda_T = 1$
TrustSVD	$\lambda = 0.001, \lambda_T = 1$
SSL-SVD	$\alpha = 0.3, \lambda = 0.001, \lambda_T = 1$
Our Approach	$\alpha = 0.0, \lambda = 0.001, \lambda_T = 1, \nu = 0.05$

Recommender Results

In this section we present figures which show the performance of the recommender with different hyperparameter configurations. We then describe the process used to select the best hyperparameters. For each hyperparameter we show the results for both RMSE and MAE as well as all users and cold-start users only.

FilmTrust

Alpha

Below we present the tuning results for α. α is the hyperparameter which controls the ratio between the sparse trust and social trust in the loss function (Equation 12). The learning rate affects how quickly the gradient descent changes with each iteration. Too large of a learning rate can cause the descent to overshoot the minimum, while too small of a learning rate may cause the algorithm to be stuck in a local minimum that is higher than the global minimum, or to fail to converge.

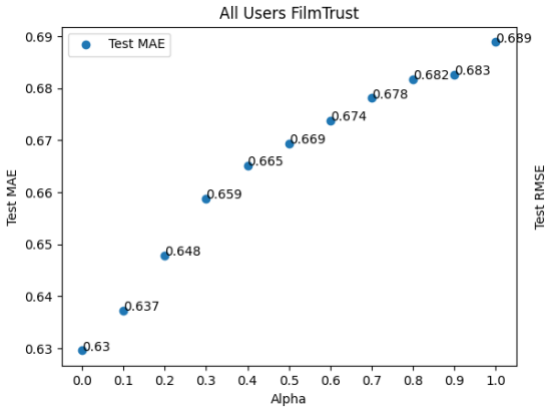


Figure 1. MAE vs α All

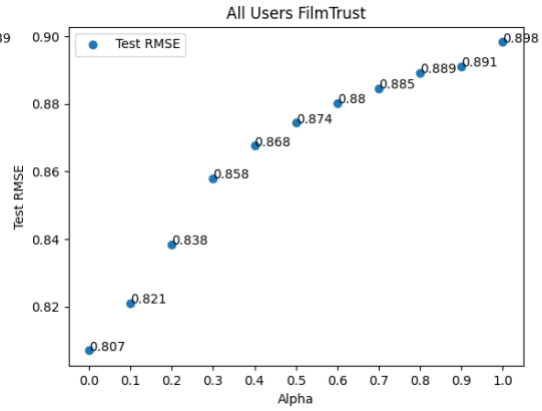


Figure 2. RMSE vs α All

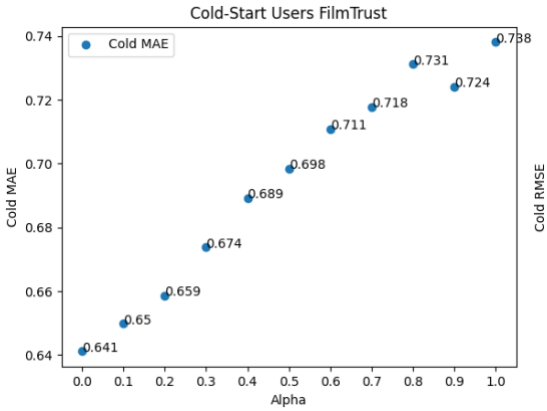


Figure 3. MAE vs α Cold

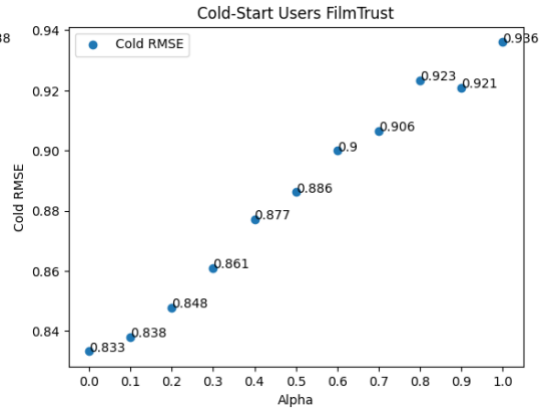


Figure 4. RMSE vs α Cold

Learning Rate

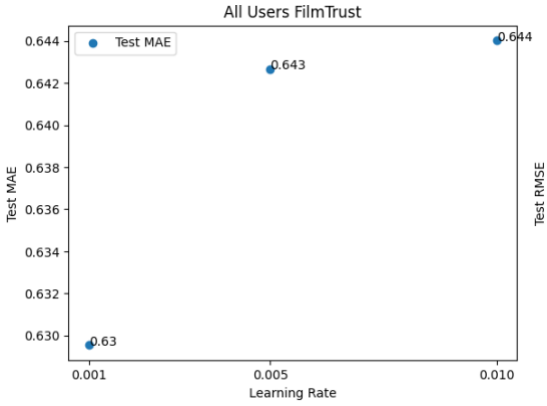


Figure 5. MAE vs Learning Rate All

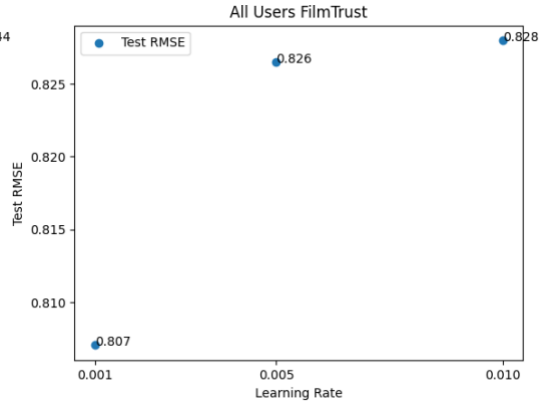
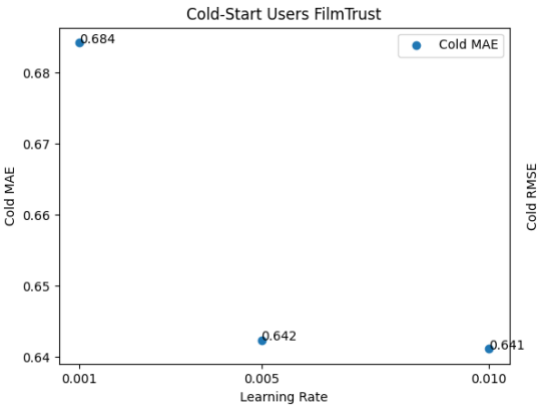


Figure 6. RMSE vs Learning Rate All



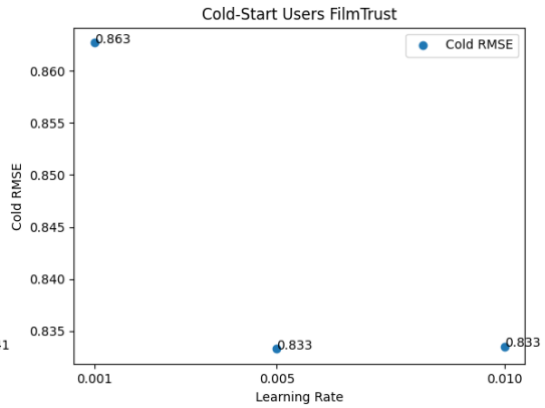
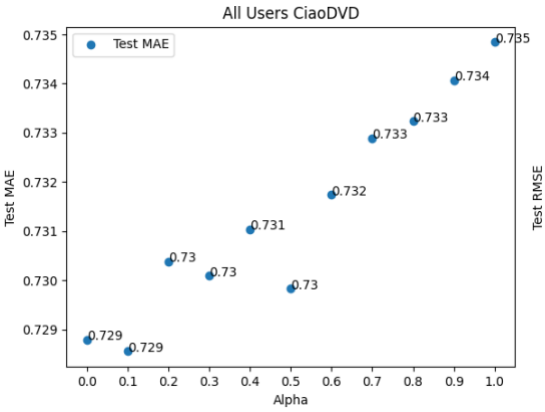


Figure 7. MAE vs Learning Rate Cold **Figure 8.** RMSE vs Learning Rate Cold

Based on Figures 1-4 we selected 0 as the optimal value for α because the lowest error was universally found with this value. This suggests that the sparse trust predictions made by the OC-SVM model were the most predictive. Our learning rate was set to 0.001 based on the results on all users, however the cold-start users performed better with a higher learning rate as seen in figures 5-8.

CiaoDVD

Alpha



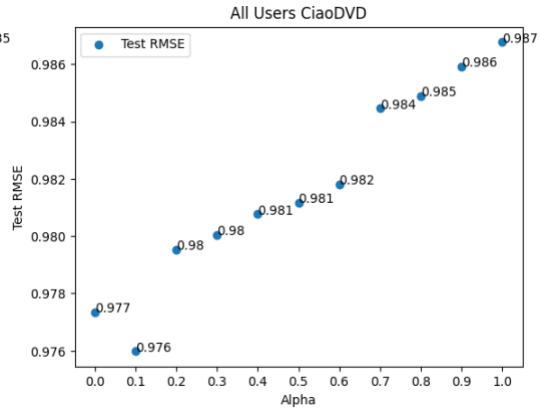


Figure 9. MAE vs Alpha All **Figure 10.** RMSE vs Alpha All

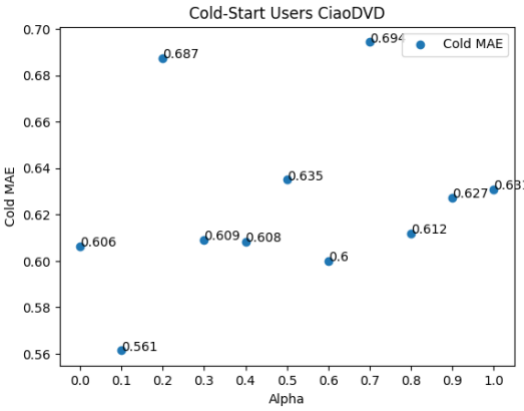


Figure 11. MAE vs Alpha Cold

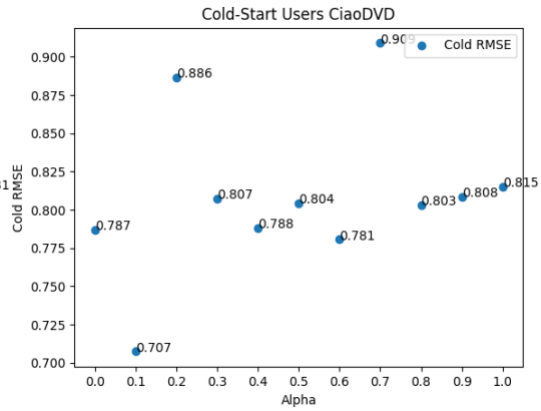


Figure 12. RMSE vs Alpha Cold

Learning Rate

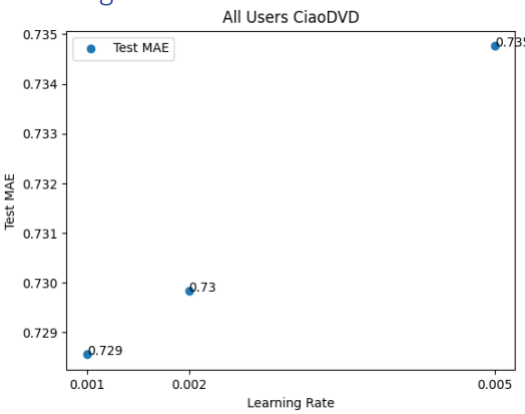


Figure 13. MAE vs Learning Rate All

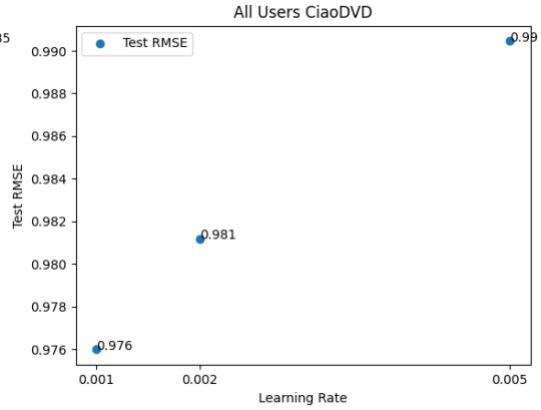


Figure 14. RMSE vs Learning Rate All

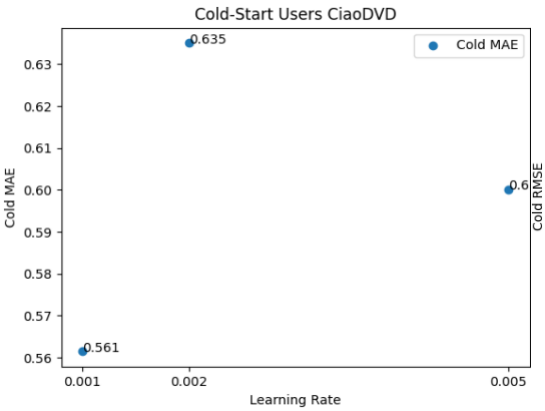


Figure 15. MAE vs Learning Rate Cold.

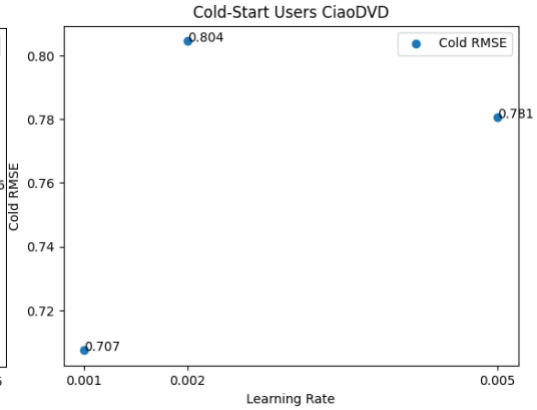


Figure 16. RMSE vs Learning Rate Cold

The results for CiaoDVD seen in figures 9-16 showed similar best-performing hyperparameters, also suggesting that an α of zero performed best.

Based on Table 4, the best performance is achieved by using almost entirely sparse trust. This suggests that the use of OC-SVM for prediction of sparse trust is effective at improving predictions, when compared to the use of social trust alone.

Model Comparisons

Table 5. The performance of each model on both datasets for all users.

Dataset	Metrics	PMF	SVD++	SoRec	RSTE	SocialMF	TrustMF	TrustSVD	SSL-SVD	Our Method
	MAE	0.66	0.642	0.657	0.656	0.668	0.637	0.593	**0.584**	0.632
Film	(Improve)	4.28%	1.60%	3.84%	3.70%	5.43%	0.83%	-6.53%	**-8.18%**	-
Trust	RMSE	0.863	0.87	0.872	0.852	0.875	0.815	0.745	**0.737**	0.810
	(Improve)	6.20%	6.95%	7.17%	4.99%	7.48%	0.67%	-8.66%	**-9.84%**	-
	MAE	0.862	0.755	0.767	0.767	0.76	0.747	0.76	0.747	**0.731**
Ciao	(Improve)	15.23%	3.21%	4.73%	4.73%	3.85%	2.18%	3.85%	2.18%	**-**
DVD	RMSE	1.209	1.078	1.148	1.145	1.107	1.028	1.008	0.989	**0.980**
	(Improve)	18.96%	9.11%	14.65%	14.43%	11.49%	4.69%	2.80%	0.93%	**-**

Table 6 The performance of each model on both datasets for cold-start users.

Dataset	Metrics	PMF	SVD++	SoRec	RSTE	SocialMF	TrustMF	TrustSVD	SSL-SVD	Our Method
	MAE	0.772	0.677	0.670	0.745	0.733	0.675	0.667	**0.649**	0.693
Film	(Improve)	10.23%	-2.36%	-3.43%	6.98%	5.46%	-2.67%	-3.90%	**-6.78%**	-
Trust	RMSE	0.973	0.875	0.902	0.910	0.914	0.870	0.868	**0.864**	0.870
	(Improve)	6.20%	6.95%	7.17%	4.99%	7.48%	0.67%	-8.66%	**-9.84%**	-
	MAE	1.021	0.775	0.796	0.895	0.770	0.764	0.735	0.730	**0.580**
Ciao	(Improve)	43.18%	25.15%	27.13%	35.19%	24.66%	24.07%	21.08%	20.54%	**-**
DVD	RMSE	1.204	1.033	1.002	1.108	0.997	0.963	0.961	0.957	**0.732**
	(Improve)	39.21%	29.14%	26.95%	33.94%	26.58%	23.99%	32.83%	23.51%	**-**

Based on Tables 5 and 6 we show that the model improved performance on the CiaoDVD dataset significantly, while underperforming the most recent state-of-the-art-models on the FilmTrust dataset. Our method stood out in its ability to predict ratings for the cold-start users on the CiaoDVD. This is especially important as cold-start users are the primary target for trust-based recommenders. While the model was not able to outperform every model on both datasets, the large increase in performance on the CiaoDVD dataset cold-start users suggests that our model could be very useful for providing recommendations for cold-start users.

Chapter 5: Discussion

Our proposed algorithm performed significantly better on one of the two datasets. Based on this, we believe that OC-SVM has the potential to improve recommendations where trust data is available. We believe these results will improve what can be achieved using trust-based recommenders and will be useful in providing predictions, especially for cold-start users. We think that the application of OC-SVM has improved the predictions obtained by penalizing those relations identified as distrust.

The recommender system was programmed in Python. Python was chosen due to its wide variety of packages which support machine learning and vectorized programming. Numpy was used primarily for its Array object and vectorized methods. Pandas was also used to represent the tabular data as data frames. These packages can significantly speed up the calculations compared to native python lists and dictionaries. For the CiaoDVD dataset the recommendation took $26{,}548 \pm 7527$s and FilmTrust took 778.8 ± 24.6s.

Chapter 6: Conclusion and Future Work

Future work to expand on this project could include testing alternatives to OC-SVM as a one class classification algorithm. There are other promising alternatives which employ neural networks, which are becoming increasingly powerful and popular as computing speed continuously improves. The neural networks can address the weaknesses of OC-SVM as datasets become increasingly large. Chalapathy et. al found these to be quite effective in comparison with OC-SVM [22]. The separation of the model into two algorithms will allow future practitioners to apply new algorithms to replace OC-SVM without needing to modify Algorithm 2.

Further improvements could also be sought by attempting to incorporate additional data into the model as was done with trust, for example labels of genre or lead actors. These types of data are often used in content-based recommenders [23]. Another widely discussed extension of trust data is a temporal component. It has been theorized that trust between parties evolves over time and therefore trust can be made or broken as time goes on. This could be studied by including the time a trust relation was formed into the relative weight given to it.

References

[1] B. Smith and G. Linden, "Two Decades of Recommender Systems at Amazon.com," *IEEE Internet Computing,* pp. 12-18, 2017.

[2] J. Bennett, S. Lannign and N. Netflix, "The Netflix Prize," in *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[3] J. Z. D. T. Guibing Guo, "A Simple but Effective Method to Incorporate Trusted Neighbors in Recommender Systems," in *User Modeling, Adaption and Personalization*, 2012.

[4] Z. Hu, G. Xu, X. Zheng, J. Liu, Z. Li, Q. Z. Shang, W. Lian and H. Xian, "Ssl-Svd: Semi-Supervised Learning--Based Sparse Trust Recommendation," *ACM Transaction on Internet Technology,* vol. 20, no. 1, pp. 4:1-4:20, 2020.

[5] D. Lee and H. Seung, "Learning the Parts of Objects by Non-Negative Matrix Factorization," *Nature,* vol. 401, pp. 788-91, 1999.

[6] Y. Koren, "Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model," *ACM,* 2008.

[7] P. Massa and P. Avesani, "Trust-aware Recommender Systems," in *Proceedings of the 2007 ACM conference on Recommender systems*, New York, NY, USA, 2007.

[8] G. Guo, J. Zhang and N. Yorke-Smith, "Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence,* pp. 123-129, 2015.

[9] Y. Pan, F. He, H. Yu and H. Li, "Learning adaptive trust strength with user roles of truster and trustee for trust-aware recommender systems," *Applied Intelligence,* vol. 50, no. 2, pp. 314-327, 2020.

[10] Y. Z. L. J. M. F. Guangquan Xu, "TT-SVD: an Efficient Sparse Decision Making Model with Two-way Trust Recommendation in the AI Enabled IoT Systems," *IEEE Internet of Things Journal,* vol. 8, no. 12, pp. 9559 - 9567, 2020.

[11] L. H. G. X. X. W. Shuiguang Deng, "On Deep Learning for Trust-Aware Recommendations in Social Networks Publisher: IEEE Cite This," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 28, no. 5, pp. 1164 - 1177, 2017.

[12] D. Barbella, S. Benzaid, C. Janara, B. Jackson, X. Qin and D. Musicant, "Understanding Support Vector Machine Classifications via a Recommender System-Like Approach," in *DMIN*, 2009.

[13] W. Noble, "What is a Support Vector Machine?," *Nature Biotechnology,* vol. 24, pp. 1565-7, 2007.

[14] R. C. W. A. J. S. J. S.-T. J. C. P. Bernhard Schölkopf, "Support vector method for novelty detection.," *NIPS*, vol. 12, pp. 582-588, 1999.

[15] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," 2019.

[16] A. M. Ruslan Salakhutdinov, "Probabilistic Matrix Factorization," in *Proceedings of the 20th Interna- tional Conference on Neural Information Processing Systems*, 2007.

[17] H. Y. M. R. L. I. K. Hao Ma, "SoRec: social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008.

[18] I. K. M. R. L. Hao Ma, "Learning to recommend with social trust ensemble," in *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2009.

[19] M. E. Mohsen Jamali, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of the 4th ACM Conference on Recommender Systems*, 2010.

[20] Y. L. D. J. L. B. Yang, "Social collaborative filtering by trust," in *IJCAI International Joint Conference on Artificial Intelligence*, 2013.

[21] J. Z. N. Y.-S. Guibing Guo, "A Novel Bayesian Similarity Measure for Recommender Systems," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[22] R. Chalapathy, A. K. Menon and S. Chawla, "Anomaly Detection using One-Class Neural Networks," 2019.

[23] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems," in *The Adaptive Web*, Berlin, Heidelberg, Springer Berlin Heidelberg, 2007, pp. 325-341.

[24] T. Horiuchi, B. Bishofberger and C. Koch, "Support Vector Method for Novelty Detection," in *Advances in Neural Information Processing Systems*, 1994.

[25] L. Rozonoer, B. Mirkin, I. Muchnik, E. Bauman and K. Bauman, "One-Class Semi-supervised Learning," in *Braverman Readings in Machine Learning. Key Ideas from Inception to Current State*, Springer International Publishing, 2018, pp. 189-200.

[26] S. Deng, L. Huang, G. Xu, X. Wu and Z. Wu, "On Deep Learning for Trust-Aware Recommendations in Social Networks," *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, vol. 28, no. 5, pp. 1164-1177, 2017.