Forecasting movie ratings with recommender systems

A research in collaborative filtering

Oscar Schyns
375237
July 2 2017

Abstract

In this paper I construct different recommender systems to predict movie ratings and compare there performances. Recommender systems can be used to classify movie ratings for users as interesting or not worth watching and with a few simple statistics the classification performance can be compared for different models. The relevance of this topic is growing every year with the exponential increase in data available. A lot of study has already been done about this topic but there is still work to be done in increasing the forecasting power of different recommender systems. The most important recommender system in this topic is a neural network. Different models are added to compare the networks performance or try to improve it. The network uses singular value decomposition to extract less dimensional information for a large and sparse data matrix. I conclude that all models constructed in this paper do have a good in-sample fit but the out-of-sample classification power is relatively modest. With many ratings near the classification threshold out-of-sample classifying is harder.

Erasmus university Rotterdam
Erasmus school of Economics
Bachelor Thesis [FEB23100-16]
Supervisor: Castelein, A
Second assessor: Frasincar, F

Contents

1 Introduction

With the Internet as a digital database the amount of data is expect to double every two years in the next decade. Just like a library orders books by topic and alphabetical order to make it easier to find the items you are looking for in such a huge pool, the Internet needs it's algorithms. The most famous examples of these algorithms are search engines, websites who use a string as input and gives you matching sites or video's or pictures as output. Recommender systems on the other hand order items on the probability of you liking it. Recommender systems try to find correlations between user ratings and recommends items similar users liked. Recommender systems fall in two distinct categories, content-based methods require textual description of item's and use them also to recommend items and collaborative methods forecast with product ratings only. Since textual descriptions are often not present most papers focus on collaborative methods. In my paper I construct different collaborative recommender systems and compare their performances. Compare them with each other but also compare them with random benchmarks such as the average of a persons ratings to see how strong the forecasting powers of the models are, and if they are significantly better than the benchmarks. This paper is an addition to the work of Billsus and Pazzani (1998) who constructed collaborative recommender systems to forecast movie ratings.
Given the amount of data and consumer products this is highly relevant. Consumers spend a lot of time finding certain products and using a recommender system could save many hours. One of the advantages of a recommender system over a search engine is that it can suggest items you will probably like even though you do not know what to look for. Websites using systems like this can sell more goods and if companies like Netflix can find items customers like these customers would be more satisfied with the company which can also lead to more customers. Netflix offered a million dollars to the person who could maximize the forecasting power of their recommender system and given the volume of Internet sales you can understand one thing: recommender systems are big business!

2 Literature review

For the construction of the recommender system I will follow approximately the same procedure as in Billsus and Pazzani (1998). In their paper they construct collaborative recommender systems to forecast movie ratings of individuals. They start with a simple correlation based system researched in other papers that predict a rating based on the rating of other users and the correlation between user ratings of individuals Shardanand and Maes (1995). They predict the movie rating of a user by taking the weighted average (weigh them by correlation) of the difference of a users rating for that movie with its average rating and add the average rating of the user they predict for. Tuzhilin and Adomavicius (2005) explain that taking the difference with the average is important since some people scale their ratings differently. This model has a few problems, most notably is that the correlation is determined only by items rated by two users, just because two

people rated different items doesn't mean they are not alike. Another problem is that if the overlap of rated items between two users is small, the correlation becomes a unreliable predictor of similarity.
Another way to use collaborative filtering to recommend items is to create a classification task. In a classification problem data is divided in certain disjoint classes and the recommender system is trained to forecast the class of a given object. Billsus and Pazzani divide all user ratings in two classes, low or high ratings. All data is stored in a matrix where the columns represent movies and the ratings of users are split up in two rows, the first row is a boolean row-vector where a one means that the user rated the movie low and a zero that he did not dislike it. The second row-vector contains a one if he liked it and a zero if he did not like it. This set-up is made because a lot of movies are unrated by a single user so the fact that a user did not rate a movie low doesn't mean he rated it high.
To reduce dimension of the large data matrix, and to compare users even if they did not rate the same items Billsus and Pazzani conducted singular value decomposition (SVD). Since SVD only works for a feature if they appear twice all rows with less than two ones are removed. After the decomposition of the matrix the lowest singular values and matching singular rows are discarded until the number of singular values equals k $= 0.9 \cdot \text{rank(A)}$, this amount was chosen since it resulted in the best performance using a tuning set. They scale their left over singular vectors by their matching singular values to represent a movie in k dimensions. With this information Billsus and Pazzani train a neural network to predict a movie rating of an individual. After experiments on a tunning set they conclude that the network performed best with two hidden units and one output unit. The hidden units use sigmoid functions as activation function and the output unit uses a linear function. With the model they forecast the difference of the users rating of a movie compared to the average rating. All weights are determined with backpropagation. Before they forecast they must first change the user rating vector of an item to a feature vector in k dimensions to match the format of the training data. After they feed the vector to the model they add the average rating of that movie to create the predicted rating convert it to a binary rating using the threshold. The binary variable represents the predicted class. The same neural network is used to construct their third model. In the third model they use a different information extraction algorithm instead of SVD. Here they use the expected information game method (Quinlan, 1986). This method selects the k variables that maximize the expected information gain. With movie ratings it selects the k variables which effect the expected rating the most after those variables where added. Billsus and Pazzani select the same number of variables with both methods so the topology of the network is not effected.
The forecasts of all 3 models are transformed to binary ratings and they compare the classification accuracy of all 3 models. Apart from comparing the models with each other they also compare the model performance over the amount of training data. They conclude that both neural networks perform significantly better than the correlation based approach. The performance of all models increases with more training data. The results show empirical evidence that a neural network can be trained to create a accurate

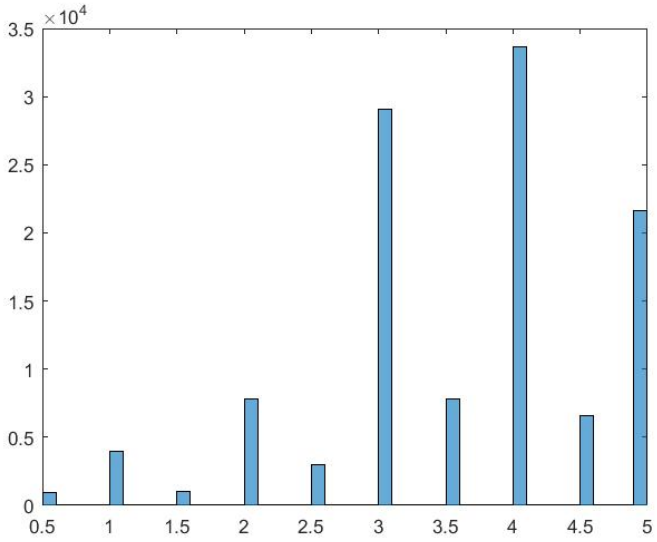
recommender system.

3 Data

In this research I focus on forecasting user ratings of movies. I use data from the MovieLens database[1], this database consists of 20 million ratings from 138,000 users for 27,000 movies. Since this database is a bit large I only use the ratings of the first 2,000 users and 1,410 movies. The same amount of data Billsus and Pazzani used. All ratings are discrete from 0.5 to 5.0 with an increment step of 0.5, with an average rating of 3.6006 and a standard deviation of 1.005. The most commonly given rating is 4.0 and the least commonly is 0.5, the median rating is 4.0. Since the average rating is quite high I only classify a rating of a movie as high if the user rated it 4.0 of higher. The data matrix is quite sparse, most users only rated a small part of the movies in the database. In the subset of 2,000 users and 1,410 movies the average user only rated about 4% of the movies in the database. I use the first 90% of all movies seen by a user as the in-sample set and the other 10% is the out-of-sample set and will only be used for forecasting. With this data I create recommender systems for 20 random users who saw at least 200 movies. In the histogram below it looks like users prefer to give integer ratings since all integer ratings are more common than the 2 nearest non integer ratings.

[1] *https://grouplens.org/datasets/movielens/20m/*

Figure 1: Histogram of all movie ratings given by users in the dataset



4 Methodology

4.1 Correlation model

The first model to classify movies is the correlation model. Predictions are based on the ratings of other users and the similarity between the users. A rating is computed by the following formula:

$$U_x = \overline{U} + \frac{\sum_{i=1}^{N}(J_x - \overline{J})r_{UJ}}{\sum_{i=1}^{N}|r_{UJ}|} \tag{1}$$

Where $\overline{U}$ is the average rating for user U and r_{UJ} is the correlation coefficient of the ratings of users U and J. To calculate the correlation coefficient and thus the model for a given user I divide the data in an in-sample and an out of sample part. For every user in the database I use approximately 90% of the movies he rated to construct the correlation coefficients. The estimation sample includes the first 90% of movies seen by the user who I create the model for (from now on referred to as user 1) and all movies with lower indexes, the movies not rated by user 1 still have ratings with valuable information because those ratings can be used to compute the average of users used in (1). I create a matrix with the first 90% of movies the user rated as columns and on the rows users who rated at least 2 movies on the columns, user 1 included. The matrix elements equal the ratings the user gave for that movie, if there is no rating provided it equals 0. With this matrix I calculate the correlation coefficients of the ratings of user 1 with all other users in the matrix while only using information of movies both users have rated. Since the correlation coefficient between 2 users only uses the ratings of movies both rated in the estimation sample it is possible one of the 2 users rated all those movies the same. In that case the variance of that user is 0 and the correlation coefficient cannot be computed. Using average ratings or the variance of a users rating over all ratings in an estimation sample could lead to a correlation coefficient larger than 1 or smaller than -1. I ignore all users who don't have a correlation coefficient with user 1 will be ignored for the calculations. To forecast a movie out-of-sample or in-sample I take the weighted average according to (1) of a users rating subtracted by its average rating for all users who rated that movie and have a computed correlation coefficient with user 1. After this I add the average of user 1's ratings in the estimation sample and the predicted value is calculated. Note that in (1) all averages of a user are taken from all movies in the estimation sample seen by that user, not just item's rated by both user 1 and the user in the estimation sample. After all missing ratings of user 1 have been estimated I create the same model for the other 19 users randomly selected from the database.

4.2 Eigenvectors and eigenvalues

An eigenvector of a matrix is a column vector whose direction does not change if multiplied with the initial matrix. Eigenvectors are linearly independent and if multiplied with the matrix equal the scalar product of the eigenvector with it's corresponding eigenvalue. Eigenvectors and eigenvalues of matrices come in pairs.

$$Av = \lambda v \tag{2}$$

In equation 2 v is an eigenvector with one column and the same amount of rows as the number of columns of matrix A, λ is an eigenvalue.

$$det(A - \lambda I) = 0 \tag{3}$$

Equation 3 holds for every eigenvalue so by calculating the determinant all λ's can be computed. The number of λ's equals the rank of matrix A. The eigenvectors can be calculated by combining the results of (2) and (3).

$$(A - \lambda I)v = 0 \tag{4}$$

4.3 Singular value decomposition

Singular value decomposition (SVD) is a dimension reduction method that decomposes a matrix with r rows and c columns into the three matrices in the following way:

$$A = U\Sigma V^T \tag{5}$$

U(r·m) and V(c·m) are a collection of orthonormal vectors called the singular vectors with m dimensions m equals to the rank of matrix A. Σ(m·m) is a diagonal matrix that has the singular values of A descending on its diagonal. SVD splits the data in matrix A up in m hidden concepts who form the underlying driving factor between the correlation of rows and columns in A. Every singular value belongs to concept, the higher the value, the more variance of the data in A can be explained by that concept. Matrix U consists of the correspondence of the rows of A to the m concepts, hence it is a $c \cdot m$ matrix. Matrix V does the same only for the correspondence of the columns of A to the m concepts and Σ is used to scale all dimensions in matrices U and V. In the case of movies concepts can be seen as genres such as horror or science-fiction. So the SVD for a matrix A with rows corresponding to users columns corresponding to movies, and ratings as items splits both users and movies up in hidden concepts as genres. Matrix U is a matrix of user vector where each user is split in in correspondence to m genres, and matrix V is the movie to user matrix.
SVD uses the eigenvectors and eigenvalues to create the matrices U, Σ and V. The eigenvectors of AA^T make up the columns of matrix U and the eigenvectors of A^TA make up the columns of matrix V. The singular value's of matrix Σ equal the root of the eigenvalue's of AA^T.
In my recommender system I use SVD to extract less dimensional information from the large and sparse data matrix. First I create a rating matrix where each row represents a user and each column a movie. Most users rated only a few movies so a lot of matrix elements are unknown, filling in a 0 at these blank spaces is not an option since that gives the impression everybody dislikes all movies not seen. To fix this issue I transform the rating vector of a individual to 2 boolean vectors like in Billsus and Pazzani (1998).

The first vector has a 1 as rating if the user saw it and liked it, and the second vector has a 1 as rating if the user saw it and did not like it, in all other cases the rating on that vector is 0. Placing these vectors on the rows of my new matrix gives us as 4000 by 1410 rating matrix. Because SVD uses the correlation between users and movies to compute the matrices all users who rated less than 2 movies, or movies with less than 2 ratings add no information about other users or movies and can be deleted from the matrix. This reduces the number of rows and columns in the matrix. For every user SVD is conducted separately. SVD transforms movies in m dimension vectors, those vectors can be used to train the model. This works in-sample but SVD cannot compute the movie to concept matrix V for movies out-of-sample. The-out-of sample m dimensional vectors for movies can be computed with the information we have of users in-sample U and them liking or disliking the movie. This way we can transform movies both in and out of sample with the following formula.

$$A^T U \Sigma^{-1} \tag{6}$$

It is also possible to ignore the Σ matrix in formula 6 since all the last term does is scaling the concepts in each observation. The method I use leads to relative smaller values for important concepts and will increase the coefficients for those concepts.
For every user SVD has to be conducted separately. First I remove all movies not seen by the user I conduct SVD for. Secondly I remove 30% of the movies he saw and add them to the out-of-sample-set. And thirdly I remove all users who saw less than 30 of the movies the user I conduct SVD for as well as the user itself. This gives a unique matrix A for each user. Not all m concepts are useful as an input for my second recommender system. Some concepts explain barely any variance of the movie ratings and can be seen as errors in matrix A rather than useful variables. Only the k largest singular values will be used, reducing the number of variables in a movie vector from m to k. I use the performance measures in section 4.5 to choose the number of dimensions in the training data. The k dimensional movie vectors will be used as input for the neural network.

4.4 Neural networks

The second recommender system I construct is a neural network. I train a neural network to predict the movie rating of a given movie and user minus the average rating for that movie Y_k. The network uses the k dimensional feature vectors of movies as input, the number of variables used from a movie vector is determined with the performance measures in the next section. A neural networks is a regression model with two or more stages. A neural network can be represented as a three layer network diagram where the first layer of nodes models the input X_p, the second layer the hidden units Z_m and the third layer the output Y_k. The value of each node depends on a linear combination of the values under it.

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), \qquad m = 1, ..., M \tag{7}$$

$$T_k = \beta_{0k} + \beta_k^T Z, \qquad k = 1, ..., K \tag{8}$$

$$f_k(X) = g_k^T(T), \qquad k = 1, ..., K \tag{9}$$

σ is the activation function and most often non-linear, $g(T)$ will be referred to as the transfer function. The coefficients are estimated by the process of backpropagation. Backpropagation consists of 2 steps, the first step calculates the the sum-of-squared residuals of the model with its current coefficients shown in equation 10 and is called the forward pass, the second step minimizes those errors with respects to the weights by gradient descent and is called the backward pass. Both steps executed once equal 1 training epoch. The first epoch uses random coefficients to forecast.

$$R_i = (y_{ik} - f_k(x_i))^2 \tag{10}$$

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \tag{11}$$

The above function shows how a parameter is updated after a training epoch. γ_r is the learning rate of the model at the r^{th} iteration and is set to either a constant or a decreasing function depending on the learning method. In my models I change the parameters after I back-propagated all observations, this is called batch learning and models who have batch learning use constant learning rates.

Neural network models have a lot of weights, especially models with a lot of hidden units. This can lead to overfitting, to combat this problem 2 methods have been developed. The first method is to add a function of the coefficients as a penalty to the error function 10 and minimizes this combined function with equation 11.

$$R_i = (y_{ik} - f_k(x_i))^2 + \lambda \sum_{km} \beta_{km}^2 + \lambda \sum_{ml} \alpha_{ml}^2 \tag{12}$$

This shrinks the coefficients especially those who do not have a lot of forecasting power. This is called weight decay. The penalty function can have many forms but most often the sum-of-squared coefficients is used. The second option is to use a validation set, this set is not included when conducting SVD and is not used to train the network but at the end of every epoch the current coefficients are used to predict the outcomes of these observations. If at a given point of the training the sum-of-squared-residuals in-sample is decreasing but out of sample increasing all new improvements in-sample are probably not valid externally. The new improvements in that case probably do not describe the pattern of movie concepts and the ratings of that movie but rather the errors in-sample, the model is probably overfitting. If the sum-of-squared residuals of the validation set increases over a determined number of epochs the training is stopped and all coefficients are set to the coefficients when the sum-of-squared residuals started

decreasing. Because the input with SVD is scaled by equation 6, which leads to higher coefficients for important concepts I choose the validation set approach, the weight decay method would effect higher more import coefficients more compared to lower coefficients. If the sum-of-squared residuals of the validation set increases for 10 epochs the model stops training, if this does not occur the model stops automatically with training after 1200 epochs. The network topology will be determined using the performance measures in the next section.

4.5 Performance measures

All models generate discrete outcomes. These can be compared with non-boolean movie ratings using the MSE, the MSE calculates the mean of the squared errors of all ratings predicted. The bias of the entire model equals the average predicted rating minus the average rating.
The models also can be used to classify ratings in 2 distinct categories. Here I consider all ratings as high if they are above the average rating of 3.6006 or higher, and low otherwise. All high ratings will be labeled as hot and low ratings will be labeled as cold. To evaluate the predicting power of the classification models performance measures have to be used. The most simple performance measure would be the percentage of correctly classified movies. When using this measure models who classify every movie as hot could have an reasonable high performance measure as long as the user rates a lot of movies high. In order to combat this problem I use the same performance measures as Billsus and Pazzani(1998), *precision* is the percentage of movies classified as hot who are truly hot, and *recall* is the percentage of movies who are hot classified as hot for the given user. *precision* could be easily optimized separately by only classifying movies as hot who have a very high probability to be liked by the user and *recall* could be easily optimized by classifying all movies as hot. To neutralize the imperfections of the separate measures a joint measure has to be used. The $F-measure$ combines both measures in the following way.

$$F - measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{13}$$

The last measure is called the random model. The random model classifies each movie randomly as hot or could with the probability of it being hot equal the percentage of movies in the in-sample set who are hot of a user. This measure will be used to compare the performance of the neural network classification.

4.6 Neural network topology

I select a few users to see what network topology produces the best results. All users I create networks for have seen at-least 200 movies. I divide the data in a training set, a validation set, and 2 test sets. The first test set will be used to see what model produces the best results. The second test set is the out-of-sample set and this set is only used for forecasting. The training set of one user has the data of 70% of all movies seen by that

user, the other 3 sets all 10% each. After comparing topological different models I use the model with the highest $F-measure$ with the first test test to forecast out-of-sample.

My network of choice has 1 hidden layer with 2 hidden neurons. The activation function $\sigma()$ is a sigmoid function and the transfer function is linear. I use the strongest 20% of concepts as input k = 0.2m and both functions also have biases included. The learning rate of the model is constant and equals 0.005. I train every neural network 10 times and select the set of parameters which achieved the highest $F-measure$ in the first test set. If 2 sets have the exact same $F-measure$ I choose the set with the lowest MSE.

4.7 Combined regression model

The third model combines the results of the neural network and the correlation model. This model predicts a rating with a linear combination of the ratings of the previous models.

$$\hat{y} = \beta_0 + \beta_1 y_1 + \beta_2 y_2 \tag{14}$$

All coefficients are estimated with ordinary least squares in-sample. In the regression the validation set and the first test set are considered in-sample since they 2 where used to improve the model not to test the performance of it. Only predictions are made for movies both previous models have predictions for. The neural network has less observations since only movies with more than 2 ratings of users who rated 30 movies in the boolean matrix A. The predictions are made only with the variables who significantly effect the rating. The significance of the coefficients of the estimated ratings of both models also provides a good test to see if both models predict significantly better than a constant near the average rating of a user.

5 Results

5.1 Correlation based model

The table below shows the results of the correlation based recommender system for the 20 users randomly selected.

Measures	results in-sample	results out-sample
MSE	0.5310	0.6982
precision	0.8306	0.7329
recall	0.5856	0.4068
$F - measure$	0.6869	0.5233
average rating data	3.3262	3.2550
average estimation	3.2505	3.1742
variance ratings data	1.1300	1.0663
variance estimations	0.4017	0.3398

In the results it is clearly visible that the model has a downward bias, not only is the average predicted rating lower than the average rating, from the *precision* and *recall* values we can see that more movies are hot than classified as hot. The out-of-sample perform worse than in-sample according to all measures. out-of-sample ratings are on average a bit lower than in-sample. The actual variance of the ratings are much higher than the variance of the estimations. The average rating of all 20 users both in-sample and out-of-sample is about 0.3 lower than the average rating in the data set.
The results are even worse for the users with the lowest in-sample average rating.

Out-of-sample results for the 3 users with the lowest in sample average.

Measures			
MSE	0.9963	0.9126	1.0803
precision	-	-	-
recall	0	0	0
$F - measure$	-	-	-
average rating data	2.8888	3.2414	3.1111
average estimation	2.4214	2.9428	2.9845

The models have not predicted any movie out-of-sample as hot for those 3 users, therefore the *precision* and $F - measure$ statistics can not be computed. The biases defined as the difference between the average rating in the data and the average estimated rating is also a lot larger.

5.2 Neural network

Neural network forecast results over the samples:

Measures	in-sample	validation set	test set	out-of-sample
MSE	0.5536	0.9834	0.6177	0.7489
$precision$	0.7541	0.7252	0.7290	0.6812
$recall$	0.6412	0.5080	0.7835	0.4719
$F-measure$	0.6931	0.5975	0.7553	0.5577
average rating data	3.2317	3.7077	3.7394	3.2921
average estimation	3.2156	3.5308	3.7105	3.2423
variance ratings data	1.1533	0.9469	0.7640	1.0663
variance estimations	0.6055	0.3308	0.2065	0.3901

Just like the correlation model in-samples performs better than out-of-sample. The results in sample are comparable with the correlation based model. Out-of-sample it has a smaller bias but a larger MSE. The neural networks labels less movies as hot than there are hot movies in all samples. The performance of the validation set is surprisingly poor, it has the highest MSE and bias, only the out-of-sample set has a slightly worse $F-Measure$. It is also worth nothing that the validation set and the test set have high average ratings compared with the other sets.

Results in-sample and out-of-sample of the random model

	in-sample	out-of-sample
$precision$	0.4756	0.4563
$recall$	0.4671	0.4545
$F-Measure$	0.4713	0.4554

The results of the random model in-sample and out-of-sample of the random model are almost the same. The proportion of hot movies in-sample and out-of-sample is therefore probably similar. The random model $F-Measure$ converges to the proportion of movies classified as hot in-sample. Out-of-sample it converges to a lower number because the proportion is not exactly the same as in the in-sample set. The neural network outperforms the random model out-of-sample. This shows that the model has external validity in the out-of-sample set.

Figure 2: The $F - Measure$ and MSE respectively in-sample of users over total movies seen

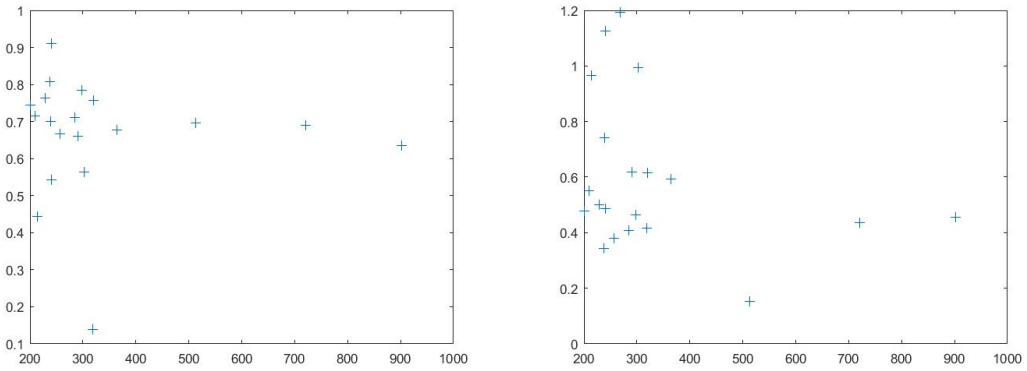
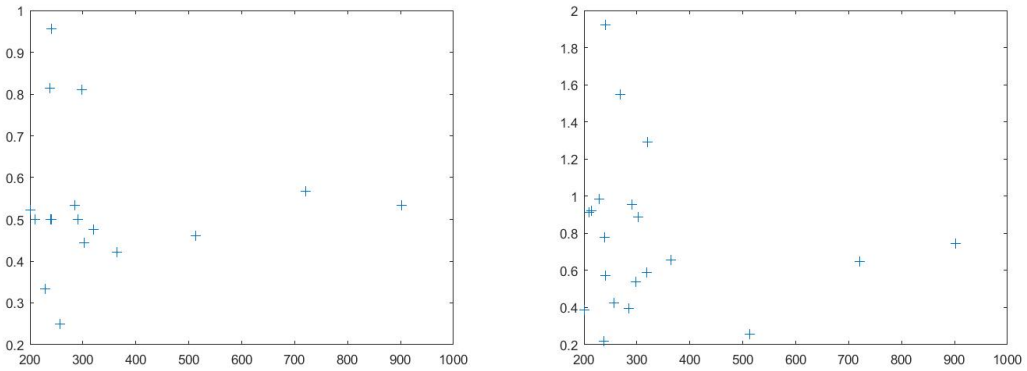


Figure 3: The $F - Measure$ and MSE respectively out-of-sample of users over total movies seen



The classification performance of the neural network does not seem to improve if a user rates more than 200 movies. The $F - measure$ is roughly the same for users who saw less than 300 movies and those who saw more than 300 movies. The MSE however does look like to decrease slightly over the number of movies seen by a user. Since the network is designed to minimize the sum-of-squared residuals it looks like users who saw more movies probably have slightly better fitting networks. For classification however there does not seem to be a difference.

5.3 Regression model

The table below is the results of the regression of the movie ratings in-sample on the forecasts of the correlation model x1 and the neural network x2.

	coefficient	p-value	R^2
constant	-0.6934	0.0000	0.5631
x1	0.8372	0.0000	-
x2	0.3927	0.0000	-

All coefficients are remarkably significant. Even-though the correlation based method does not perform much better or even better, the correlation based forecasts are the most important factor in this model. The coefficient of the correlation based forecast is more than twice the coefficient of the neural network forecast. When both coefficients are added the number is bigger than 1, this method is therefore no weighted average of both models. The coefficient of -0.69 is because of this not that surprising. The R^2 shows that a lot of the variance of in-sample ratings can be explained by the variables, but still a quite big part not explained by this model. With such a large part missing it is hard classifying accurately.

forecast results in-sample and out-sample:

Measures	results in-sample	results out-sample
MSE	0.4933	0.7289
$precision$	0.7701	0.6667
$recall$	0.7238	0.4621
$F - measure$	0.7462	0.5459
average rating data	3.3347	3.2808
average rating estimations	3.3347	3.2488

Although the regression model has more information to use than the neural network, it does not classify items better. the MSE is slightly better but definitely not significant. The bias in-sample is 0 which is a property of ordinary least-squares. Out-of-sample it has a comparable bias to the neural network model. Just like all models produced this model classifies less movies as hot than there are hot movies, but the difference is smaller than with other models.

6 Conclusion

In this paper I examine the classification performance and forecasting power of different recommender systems. Compare them with each other and across samples.
From the results I can conclude that the estimated values are correlated with the ratings and the models have a good in-sample fit. Out-of-sample the results outperform the random model benchmark significantly in a set of 614 observations but the difference is not that great. The reasons for the correlated models but relatively modest classification performance out-of-sample can be numerous. The users I randomly selected all saw over 200 movies, these people can be not very representative for the entire population. Movies with higher indexes can have higher average ratings than those with lower indexes, or there could be structural break in another way. Users seem to prefer giving integer rates which could mean they rate certain movies 4.0 when a 3.5 makes more sense. This could really hurt the classification performance and modeling such a process would require more non-linear relations or coefficients which can lead to a early overfitting of the dataset according to the validation set, especially with a relative small and sparse dataset. With a lot of the ratings near the threshold of hot classifying becomes more difficult.
The correlation based model seems to perform comparable to the other models, but it has a terrible classification performance for users who rated movies in-sample low. The model has the smallest out-sample MSE but this could be because of the small variance of the model. The variance of the estimated rating is much smaller than the variance of the actual ratings. The correlation based approach assumes that the variance of ratings of different users is similar. People who rated a lot of movies can have structural higher variances than the average user because they rate all movies they saw, not only the ones they really liked. A small variance combined with a user who has a low average rating results in a model which barely classifies any movies as hot.
Of all models the neural network has the best out-of-sample classification performance. The difference however is marginal. With a lot of the ratings near the hot threshold the small downward bias can have really big effects of the number of movies classified as hot. The performance of the network in the validation set is remarkably poor. The classification performance is better than in the out-of-sample set, but the MSE is much higher. This is an unexpected result given the fact that the model stops training after the sum-of-squared residuals in the validation set increases for 10 epochs. The result is a sign that the validation set is not very representative for the entire data. Just like the correlation model the neural network produces forecasts with a much lower variance than the actual data. A low variance can lead to a lower MSE in a model but can also hurt the classification accuracy, especially for users who have a very low average rating. The problem with using a neural network to classify items is that the network tries to minimize the sum-of-squared residuals of a sample while training, not the $F - measure$ or another classification accuracy measure. The neural network does not classify better for users who rated significantly more movies than 200, 200 rated items provide enough information for the SVD to extract information from the matrix. The small downward

bias can be a results from the early stopping method and the fact that the model does not optimize a parameter separately but changes all coefficients in 1 epoch. The optimal constant added to the outputs of the hidden unites changes therefore every epoch and with the early stopping method it can never be optimized.
The regression model has extremely significant coefficients. The sum of the coefficients of the forecasts of both ratings is greater than 1 meaning that it does not take a weighted average of both forecasts. The negative constant makes sure the forecasts don't become to high. The model has the smallest in-sample MSE which is exactly what ordinary least squares tries to minimize. The neural network minimizes the MSE also but with the early stopping method the model can't always choose the exact parameters that result in the lowest in-sample MSE. Even though the model uses all predictions of the neural network and the correlation model the regression model does not classify better out-of-sample than the neural network. In-sample it has the best $F - Measure$ but with so many ratings and estimated ratings near the threshold of hot it does not take much to effect the $F - Measure$ severely.

7 Discussion

The models constructed do have explanatory and forecasting power but the results where slightly worse than I expected. A less sparse matrix could give the models better information to work with. A larger dataset also reduces the probability that a validation set is not representative for all data. More continuous ratings from 1 to 100 for example could give more information especially for ratings close to the hot threshold. Perhaps with a percentage based rating users do not round ratings upwards that frequently. The correlation based approach could be corrected for variance differences between user rating, some sort of users can have a structurally higher or lower variance. The neural network can be altered to minimize the number of incorrectly classified objects instead of minimizing the sum-of-squared residuals. Neural networks can also combined content based filtering algorithms with collaborative filtering. The singular value decomposition is a good dimension reducing method but the same models can be constructed with input from other input techniques. Most neural networks do no incorporate significance test for parameters and rely on other methods for overfitting. A neural network which also tests all its parameters for significance could have a better out-of-sample performance. The topic already occupied the minds of many great scholars but given the importance the interest in finding the perfect recommender system will only grow.

8 References

Billsus and Pazzani (1998), Learning collaborative information filters

Tuzhilin en Adomavicius (2005), Towards the next generation of recommender systems: a survey of the state-of the art and possible extensions

Shardanand and Maes (1995), Social information filtering: algorithms for automating "word of mouth"

Quinlan, J.R. (1986). Induction of decision trees. Machine Learning, 1:81–106