An Elastic Net Regularized Matrix Factorization Technique for Recommender Systems

Bianca Mitroi
Erasmus University Rotterdam
Rotterdam, the Netherlands
ibmitroi@yahoo.com

Flavius Frasincar
Erasmus University Rotterdam
Rotterdam, the Netherlands
frasincar@ese.eur.nl

ABSTRACT

Matrix factorization models are becoming increasingly popular in the field of collaborative filtering recommender systems. Recent developments in this area of research use a penalization method, such as the L_2 penalty, to restrict overfitting and reduce sparseness. We propose an alternative way of regularizing matrix factorization for recommender systems, i.e., the elastic net. A compromise between the L_1 and L_2 penalties, the elastic net can be implemented in any coefficient estimation scenario. We evaluate the performance of our model on real-world data, namely the MovieLens 100K dataset. Comparison with two more restrictive models shows that our proposed regularization provides superior accuracy in predictions, as measured by the mean absolute error. Moreover, prediction errors for individual users occur less often, and we are able to accurately predict 95.02% of the ratings with an error of at most two points from the real ratings, given on a scale from 1 to 5. Finally, sensitivity analysis shows the stability of the proposed solution.

CCS CONCEPTS

• **Information systems → Recommender systems**; *Personalization*; Retrieval tasks and goals;

KEYWORDS

recommender systems, regularization, elastic net, MovieLens

ACM Reference Format:
Bianca Mitroi and Flavius Frasincar. 2020. An Elastic Net Regularized Matrix Factorization Technique for Recommender Systems. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3341105.3373847

1 INTRODUCTION

The World Wide Web is constantly experiencing a massive increase in the amount of information it provides. A 2013 study approximates that as much as 90% of the data available at that time had been created in the previous two years alone [18]. The amount of data available online, be it movies, music, news, scientific articles, etc., guarantees that users can find virtually any piece of information. However, with the rapid increase in accessible information, the ease of finding the exact piece of information that a user needs is increasingly hindered; all the information a user needs is available online, but finding it may be extremely challenging.

The previously described scenario is where the need for effective and efficient recommendation systems stems from; recommender systems are becoming more and more popular in today's abundance of information available with just the click of a button or the tap of a screen. They are used to filter a stream of information according to the needs and preferences of a certain user. In particular, recommenders are tools that try to predict the rating or preference that a user would give to an item. Thus, these systems make it possible to prompt a user only with items that he/she might find interesting, instead of letting the user browse through all available items. They have numerous applications, in particular since the advent of the Internet and the enormous increase in information available online. Given this benefit, it is not surprising that recommender systems are widely employed in fields like movies, music, news, and e-commerce. In particular, movies providers like Netflix, Hulu, and iTunes owe their existence to a certain extent to their recommender engine. If the system is able to understand the latent features of movies that appeal to a user, this enables the system to rank the movies which possess the same features that the user values, according to the user's preferences. These movies are then recommended to the user, since it is very likely that they match the user's tastes.

We propose a new approach to model-based collaborative filtering recommenders, and focus on latent factor models. Latent factor models are an effective solution to recommendations [2, 3, 16]. They try to reduce the dimensionality of the problem by finding features extracted from the users' stated preferences for a set of items. These are features that characterize the items and towards which users have certain affinities. Building on the state-of-the-art algorithms that exploit matrix factorization techniques, we propose a new approach by generalizing two existing algorithms [5, 9]. The goal of our research project is to fit a matrix factorization model to a set of movie ratings used as a training set in such a way that the model is able to accurately predict ratings that users would give to movies they have not seen.

2 RELATED WORK

The Netflix Prize[1] competition has shown that matrix factorization techniques outperform their classic nearest neighbor counterparts when applied to recommender systems [9]. The prize was won in 2009 by the team "BellKor's Pragmatic Chaos", whose algorithm

[1]http://www.netflixprize.com/

outperformed the algorithm that Netflix was using at that time for movie recommendations, namely Cinematch, by 10.05%[2] (with respect to RMSE). The winning algorithm includes a regularized matrix factorization approach, proving the power of such techniques for collaborative filtering recommendations.

A popular approach to collaborative recommender systems is represented by latent factor models. These models try to reduce the dimensionality of a given user-item ratings matrix by finding latent factors that are able to characterize the items and for which users have certain preferences. Due to their ability to reduce the dimensionality of datasets, the popularity of latent factor models has increased recently, especially since the tremendous increase in the amount of information available online, through which users have to browse before finding their favorite items [10, 13, 22].

Finding the latent factors and decomposing the user-item ratings matrix into the product of two smaller matrices, i.e., the user factor matrix and the item factor matrix, seem like tasks for singular value decomposition (SVD) methods. However, missing values are not admissible inside a matrix to be taken as input by an SVD method; a typical user-item ratings matrix, like the one we will analyze in this research, has a huge amount of missing values, since each user provides ratings for only a small subset of the movie items. This makes the SVD method unfit for our purpose.

Koren, Bell, and Volinsky [9] propose a regularized matrix factorization technique for recommender systems by imposing a ridge type of regularization on the user factor loadings and the item factor loadings. They estimate the model coefficients using two methods: alternating least squares (ALS) and stochastic gradient descent (SGD). As the authors report, SGD tends to be easier to implement and faster than ALS; furthermore, Aberger [1] also defends the SGD method's runtime advantage over ALS. On the other hand, ALS scales well to very large, sparse datasets [23]. In our project we choose to focus on SGD.

The literature on matrix factorization techniques for recommender systems also talks about other methods used to estimate such models. Rodrigues, Jorge, and Dutra [15] describe the cyclic coordinate descent (CCD) method, and show how a graphics processing unit (GPU) implementation of this algorithm can help accelerate recommender systems. Furthermore, Gogna and Majumdar [5] opt for a majorization-minimization (MM) approach to solve their matrix factorization model; MM is an iterative procedure that exploits the convexity of a function and is popular for avoiding intense computations, such as elaborate matrix inversions and matrix multiplications [14]. However, algorithms that implement MM procedures are usually slower than gradient descent methods [11].

Gogna and Majumdar [5] offer a different point of view. The motivation they offer for their chosen regularization approach sets them apart from Koren, Bell, and Volinsky [9]. More specifically, they argue that any given user has a certain inclination towards any type of movie, be it a drama, a comedy, a thriller, etc. Therefore, they impose ridge regularization on the user factor matrix, whereby the factor loadings are optimized and scaled, but they can never become zero. A factor of zero in this context would mean that the user has no attraction towards a certain type of movie; the authors argue that this assumption is not likely to hold. On the other hand, as far as the item factor matrix is concerned, the authors impose lasso regularization, since, they argue, a certain movie cannot contain characteristics of all genres at once; the lasso is able to meet this constraint by selecting only those features that the movie has.

[2]https://en.wikipedia.org/wiki/Netflix_Prize#2009

The elastic net regularization has been shown to outperform the lasso and has been introduced in 2005 [24]. It compromises between the lasso and ridge regularization methods, i.e., it is a linear combination of the L_1 penalty (corresponding to the lasso) and the L_2 penalty (corresponding to ridge). An important advantage of the elastic net over ridge is the former's ability to perform variable selection, which is a limitation of ridge regularization. Moreover, the authors prove that another great advantage of the elastic net is that it manages to overcome two limitations of the lasso. One of the limitations occurs in cases with high-dimensional data (the number of variables/predictors is usually denoted by p) but few observations (the number of observations is typically denoted by n); often called "large p, small n" situations, they create problems for the lasso regularization, which is only able to select at most n variables before it saturates. The second limitation of the lasso, that is successfully fixed by the elastic net, is the lasso's tendency to select one variable from a group of strongly correlated variables and ignore the others; the elastic net manages to resolve this drawback by means of its group selection power, whereby it identifies groups of highly correlated variables and treats all variables in a group alike. The mathematical manifestation of this treatment is revealed by the values of the coefficients corresponding to highly correlated variables, which tend to be equal, up to a change of sign for negatively correlated variables. This implies that all variables in a group are either selected altogether, or none of them is selected.

What motivates us to conduct this research is our aspiration to develop a more general regularization method that embeds both the method of Koren, Bell, and Volinsky [9], as well as the model proposed by Gogna and Majumdar [5]. Additionally, we identified two reasons for which we doubt the validity of the assumptions made by Gogna and Majumdar [5] on the user factor matrix and the item factor matrix. The first reason is that the factors that a matrix factorization method finds do not coincide with the typical movie genres, as the authors seem to suggest. The factors found by this method are complex and difficult to interpret; analyzing in more detail the movies that score high on a certain factor gives an idea about what the factor might be. To illustrate the complexity of such factors, consider two examples of factors discovered by Koren, Bell, and Volinsky [9], which the authors name "critically acclaimed" and "mainstream crowd-pleaser"; clearly, movies belonging to possibly (almost) all movie genres have the potential of becoming "critically acclaimed" or a "mainstream crowd-pleaser". For example, a "critically acclaimed" movie can be a drama, a thriller, a documentary, a musical, a sci-fi, or even a comedy. We believe that, indeed, only certain factors can be associated with a movie (our first hypothesis); however, in contrast to Gogna and Majumdar [5], who use the lasso on the item factor matrix, we intend to use the (superior) regularization power of the elastic net to discover which factors best characterize each movie.

The second reason that motivates our deviation from the model of Gogna and Majumdar [5] builds on the first reason and concerns the fact that users might actually have no preference for certain factors (our second hypothesis). If we consider the "mainstream crowd-pleaser" category of movies, there might be users that have such a strong preference for suspense over predictability that they would never watch such a movie. Thus, we doubt the legitimacy of the ridge regularization imposed on the user factor matrix, which is unable to perform factor selection. In order to verify the validity of our second hypothesis we impose elastic net regularization on the user factor matrix.

We address the two issues discussed above by introducing a more general matrix factorization model that regularizes both the user and the item factor matrices using elastic net regularization. This raises the following research question: *How to exploit an elastic net regularization for matrix factorization in recommender systems?* Our proposed model generalizes two models, one proposed by Koren, Bell, and Volinsky [9], which we will refer to as the KBV method, and the other one coined by Gogna and Majumdar [5], which we will refer to as the GM method. The KBV model can be seen as a special case of our approach, since ridge regularization can be regarded as a special case of the elastic net. Similarly, the GM model can also be seen as a special case of the model we develop in this research, since both ridge and the lasso are special cases of the elastic net.

We will refer to our proposed approach using the acronym ENetMF, as an abbreviation for elastic net matrix factorization. To our knowledge, this is the first work that aims to achieve this. Since our proposed method is more general and has more degrees of freedom, we expect it to be able to better describe the latent factor space and thus to perform better when compared to more restrictive methods. More specifically, we expect our model to achieve a lower prediction error due to a better attuning of the parameters. However, the runtime of our model will presumably be higher, since it is more complex than the models described by Koren, Bell, and Volinsky [9], and Gogna and Majumdar [5].

3 METHODOLOGY

In this section we describe the model we develop in our research and provide the mathematical details of an estimation method, the stochastic gradient descent, which can be summarized in two update rules.

3.1 Matrix Factorization

Matrix factorization methods have previously been shown to be superior to classic nearest-neighbor techniques for the task of generating product recommendations [9]. For example, matrix factorization models are important tools that help reduce the dimensionality of a $U \times I$ user-movie ratings matrix R by mapping users and movies to an F-dimensional latent factor space. In this space each user u is represented by a vector $p_u \in \mathbb{R}^F$, which quantifies the preference the user has for movies that score high on the corresponding factors. Similarly, item i is represented in the same F-dimensional space by vector $q_i \in \mathbb{R}^F$, whose elements quantify the extent to which movie i possesses each of the F latent factors. The goal of matrix factorization is to estimate vectors p_u and q_i for all users $u = 1, \dots, U$ and items $i = 1, \dots, I$. Then $p_u^{\mathrm{T}} q_i$ is an approximation of user u's real rating of item i, denoted by r_{ui}, so that $p_u^{\mathrm{T}} q_i = \hat{r}_{ui}$. Matrix factorization decomposes R into two matrices $P \in \mathbb{R}^{U \times F}$ and $Q \in \mathbb{R}^{I \times F}$ such that $P \times Q^{\mathrm{T}}$ best approximates R. More specifically, $P \times Q^{\mathrm{T}}$ approximates the known elements of R. Note that R often contains missing values, since users do not provide ratings for all movies. Consequently, $P \times Q^{\mathrm{T}}$ makes predictions about the missing values of R. The matrix P describes the users' preferences for the F discovered movie features, called factors. Similarly, the matrix Q depicts the salience of the F factors in each of the movie items.

3.2 Regularization Methods

We need to estimate the elements of vectors p_u and q_i for all $u = 1, \dots, U$ and $i = 1, \dots, I$; in total there are $U+I$ vectors of dimension F, which means there are $F(U + I)$ coefficients to be estimated. The optimal number of factors needs to be large enough to capture all important characteristics of the ratings matrix R, and small enough to avoid overfitting; it is common practice to experiment with different values of F and empirically find the optimal one [16]. In Section 4.3 we describe the experiments that provide empirical support for fixing $F = 8$. With $U = 943$ users and $I = 1{,}682$ movie items, we need to estimate in total $8 \cdot (943 + 1,682) = 8 \cdot 2,625 = 21,000$ coefficients.

Complex systems with many coefficients to be estimated suffer from the problem of overfitting. Overfitting is synonymous to overestimating the effect of a set of predictors on the response variable. This overstating of the impact of certain variables translates into large estimated coefficients on the respective predictors. A solution is to incorporate a selection and shrinkage procedure, also called regularization method, into the model, which penalizes the magnitude of coefficients and selects those variables that have predictive power [9]. Next we review three regularization methods.

3.2.1 *Ridge.*

Ridge regularization is synonymous with L_2 penalization on the coefficients, which is done by adding a term equal to the square of the magnitude of the coefficients [20]. This leads to the following model:

$$\min_{(p_u, q_i)} \sum_{(u,i) \in K} \left(r_{ui} - p_u^{\mathrm{T}} q_i \right)^2 + \rho_1 \|p_u\|_2^2 + \rho_2 \|q_i\|_2^2 \qquad (1)$$

where K is the set of pairs (u, i) for which the user-item rating r_{ui} is known; ρ_1 and ρ_2 are positive constants, also called shrinkage parameters, or tuning parameters, and are determined via cross-validation. The term $\rho_1 \|p_u\|_2^2 + \rho_2 \|q_i\|_2^2$ is called the penalty term; without this term, the expression in Equation 1 amounts to least squares regression. The larger the coefficients p_u and q_i are, the larger the penalty will be for making them non-zero. Thus, the penalty term in Equation 1 has the effect of shrinking the values of the coefficients p_u and q_i, as compared to the values they would get without using a penalty term. However, the coefficients cannot become zero, which means that ridge regularization is not able to perform factor selection. We obtain the KBV model [9] by imposing that $\rho_1 = \rho_2$ in Equation 1.

3.2.2 *Lasso.*

The least absolute shrinkage and selection operator (lasso) is equivalent to L_1 penalization on the coefficients to be estimated [19]. This is done by adding a penalty term proportional to the absolute value of the magnitude of coefficients. In our case, lasso regularization leads to the following model:

$$\min_{(p_u,q_i)} \sum_{(u,i)\in K} \left(r_{ui} - p_u^{\mathrm{T}} q_i\right)^2 + \lambda_1 \|p_u\|_1 + \lambda_2 \|q_i\|_1 \tag{2}$$

where K is the set of pairs (u, i) for which the user-item rating r_{ui} is known; λ_1 and λ_2 are positive constants determined via cross-validation. They control the size of the coefficients, and therefore control the amount of regularization; the higher λ_1 and λ_2 are, the smaller the estimated coefficients p_u and q_i will be. Moreover, higher values of λ_1 and λ_2 produce a greater amount of null coefficients; conversely, small values of the parameters λ_1 and λ_2 imply that the lasso penalty is less powerful in performing variable selection and producing sparse solutions.

3.2.3 Elastic Net. The elastic net is a regularization method that combines ridge and lasso regularization methods in a linear way. We propose to use the elastic net in a matrix factorization algorithm in order to learn the factor vectors p_u and q_i, which amounts to minimizing the following non-convex regularized squared error on the set of known ratings:

$$\min_{(p_u,q_i)} \sum_{(u,i)\in K} \left(r_{ui} - p_u^{\mathrm{T}} q_i\right)^2 + \rho_1 \|p_u\|_2^2 + \rho_2 \|q_i\|_2^2 + \lambda_1 \|p_u\|_1 + \lambda_2 \|q_i\|_1 \tag{3}$$

where K is the set of pairs (u, i) for which the user-item rating r_{ui} is known. The constants ρ_1, ρ_2, λ_1 and λ_2 are determined by cross-validation and quantify the extent of regularization.

Note that the KBV and the GM models are embedded in our proposed model. More specifically, the KBV model is a special case of our model if we impose the restrictions:

$$\rho_1 = \rho_2 \tag{4}$$

$$\lambda_1 = \lambda_2 = 0 \tag{5}$$

On the other hand, the GM model can be obtained by imposing the following restrictions on the parameters in our proposed model:

$$\rho_2 = 0 \tag{6}$$

$$\lambda_1 = 0 \tag{7}$$

The novelty of the GM model lies in the argumentation Gogna and Majumdar [5] offer for their choice of parameters. The authors argue that any given user has a certain preference for any given type of movie (thriller, comedy, drama, etc.), therefore it makes sense to estimate the preferences of users for all movie genres. Given this argument, they use a ridge type of regularization on the user feature matrix. On the other hand, they argue, movies cannot possibly pertain to all genres, therefore they impose a lasso type of regularization on the item feature matrix, whereby the method selects only the features that characterize a certain movie.

We find it hard to believe that a user has a certain preference for any given movie genre; for example, some users may dislike violent scenes so much that they would never watch a horror movie. This makes ridge regularization unsuitable for the user feature matrix, since it is not able to select those features that appeal to each user. Moreover, the movie features found by a latent factor modeling approach are not as obvious as the difference between comedy and drama is, but tend to be rather ambiguous, such as "critically-acclaimed" or "standard, mainstream crowd-pleaser"; a critically-acclaimed movie can be a drama, or a thriller, or a psychological movie. Undoubtedly, only certain factors characterize each movie; however, we propose to enforce an elastic net regularization on the item feature matrix, since it has been shown to overcome the limitations of the lasso. More precisely, the lasso's performance is hindered by two types of limitations. One of the limitations occurs in "large p, small n" situations, i.e., high-dimensional data but few observations; the second limitation is the lasso's inability to carry out group selection, a drawback that prevents it from treating highly correlated variables similarly [24].

3.3 Estimation Method

We investigate one approach to fitting the model described in Equation 3, namely stochastic gradient descent. The main idea behind gradient descent methods is to make small adjustments to the objective function to be minimized; in each step, the adjustment is made in the opposite direction of the gradient of the objective function. Gradient descent methods require the method to run through all the observations in the training dataset before deciding to do a single update for the coefficients in a particular iteration. In contrast, stochastic gradient descent (SGD) methods use only one observation from the training set to update the coefficients in an iteration. Thus, SGD has a clear advantage in terms of speed over standard gradient descent methods.

We estimate the coefficients p_u and q_i using a stochastic gradient descent (SGD) optimization procedure, for all $u = 1, \ldots, U$ and $i = 1, \ldots, I$. The algorithm loops through all ratings from the training set K and iteratively optimizes the coefficients by adjusting them in the opposite direction of the gradient of the objective function from Equation 3.

The general update rule of SGD methods is:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta; r_{ui}) \tag{8}$$

where θ is the vector of coefficients to be estimated, $J(\theta)$ is the objective function to be minimized, and r_{ui} is an element from the training set. The gradient operator is denoted by ∇; α is called the learning rate and needs to be chosen in such a way that it leads to stable convergence. The *soft-thresholding* operator is defined as:

$$S(x, \gamma) = \operatorname{sgn}(x)(|x| - \gamma)^+ = \begin{cases} x - \gamma, & \text{if } x > 0 \text{ and } |x| > \gamma \\ x + \gamma, & \text{if } x < 0 \text{ and } |x| > \gamma \\ 0, & \text{if } |x| \leq \gamma \end{cases} \tag{9}$$

Using Equations 8 and 9, and insights from [9] and [17], we obtain the following update rules:

$$p_{uf} \leftarrow S\left(p_{uf} + \alpha\left(e_{ui} q_{if} - \rho_1 p_{uf}\right), \alpha\frac{\lambda_1}{2}\right) \tag{10}$$

$$q_{if} \leftarrow S\left(q_{if} + \alpha\left(e_{ui} p_{uf} - \rho_2 q_{if}\right), \alpha\frac{\lambda_2}{2}\right) \tag{11}$$

The algorithm loops through all ratings r_{ui} in the training set, and at each step it computes the corresponding p_{uf} and q_{if}, where f indexes the factors, $f = 1, \ldots, F$, and where F is the dimension of the factor space. Furthermore, $e_{ui} = r_{ui} - p_u^{\mathrm{T}} q_i$ is the estimation error between the actual rating r_{ui} given by user u to movie item i and its most recent estimation $\hat{r}_{ui} = p_u^{\mathrm{T}} q_i$. Applying the update rule from Equation 10 for all factors $f = 1, \ldots, F$ leads to an updated vector p_u; updating all vectors p_u for $u = 1, \ldots, U$ leads to an updated matrix P. The same procedure is used to update matrix Q. There are five unknown parameters: the learning rate of the SGD method, α, the two elastic net tuning parameters that regularize the user factor matrix, ρ_1 and λ_1, and the two elastic net tuning parameters that regularize the item factor matrix, ρ_2 and λ_2; we describe how we estimate these parameters in Section 4.3.

4 EVALUATION

We evaluate our model by assessing its performance in comparison with two more restrictive recommenders, namely the KBV recommender [9] and the GM recommender [5].

4.1 Dataset

We use the MovieLens 100K dataset [6] to empirically validate our research. The dataset contains 100,000 ratings (1-5) from 943 users on 1,682 movies, and is freely available online[3]. For each user we dispose of user demographics like age, gender, and occupation. The average user is roughly 34 years old, and there are approximately two times more male users than female users included. Out of the 19 occupations, the most well-represented category is that of students, comprising over a fifth of the 943 users. For each movie item we have information concerning the release date and the genre; there are 18 possible genres and movies can be in several genres at once. Indeed, 833 movies belong to one genre, while the rest belong to two, three, or up to six genres.

On average, the dataset contains about 100 movie ratings per user, with users having rated from at least 20 movies to at most 737 movies. On average, each movie received about 60 ratings, while the most popular movie in the dataset received almost 600 ratings from the users. A closer look at the ratings indicates that the average rating is 3.53, with a variance of 1.27. Moreover, the mode of the ratings distribution is 4, with around 34% of the ratings taking this value. The fact that each user rated, on average, about 100 movies gives an idea about the degree of sparsity of the user-item ratings matrix. The dimensions of the matrix are 943 users × 1,682 movies, which amounts to 1,586,126 potential ratings, of which we are only given 100,000.

4.2 Experimental Setup

In order to assess the prediction performance of our recommender, we use 90% of the dataset for training and validation and keep the remaining 10% for testing the recommender. Testing is done using the parameter configuration we find optimal in the training and validation phase. Training and validation of the recommender is performed through 10-fold cross-validation, whereby we randomly partition the selected 90% of the full dataset into ten equally sized subsets. We keep one of the ten subsets as the validation set and use the other nine subsets for training the recommender. We repeat this training and validation process ten times, using each of the ten subsets exactly once as the validation set. We run the recommender using each pair of training and validation sets, and record each time the desired measures of model quality. At the end of the ten runs we average the ten results for each model quality indicator, to obtain overall measures of model quality. As measures of model quality we consider the mean absolute error, the spread of prediction error, and the runtime. We consider the parameter configuration that provides the lowest mean absolute error to be optimal. Finally, we use the optimal parameter configuration and run the recommender on the test set (i.e., the remaining 10% of the full dataset) and we report the measures of model quality we obtain.

We compare the results we obtain following our procedure with the results the KBV and the GM methods obtain on the same dataset. We consider the performance indicators achieved by *our* implementation of the two algorithms we compare against, and we run all three algorithms on the same machine. By doing so, any differences in outcomes are solely a result of the regularization technique applied.

One model quality indicator we evaluate is the mean absolute error (MAE), which measures the mean deviation of the predicted ratings from the real ratings and is an indicator of the overall accuracy of an algorithm [5]. Equation 12 shows how the MAE is computed in fold t of the 10-fold cross-validation. Moreover, K_t represents a subset of K; K is the set of pairs (u, i) for which the user-item rating r_{ui} is known - similarly, we define K_t as the set of pairs (u, i) for which the user-item rating r_{ui} is known and that are used in fold t of the cross-validation. Thus, $|K_t| = \frac{90,000}{10} = 9,000$, for all $t = 1, \ldots, 10$. The overall MAE is computed by averaging over all ten mean absolute errors found in the ten folds of the cross-validation (Equation 13).

$$MAE_t = \frac{\sum_{(u,i) \in K_t} |r_{ui} - \hat{r}_{ui}|}{|K_t|} \qquad (12)$$

$$MAE = \frac{1}{10}(MAE_1 + MAE_2 + \ldots + MAE_{10}) \qquad (13)$$

Another model quality indicator we examine is the prediction error: when talking about a single rating prediction, it is the absolute difference between predicted and real ratings. Because the ratings, true or predicted, can take one of the values 1 to 5, it follows that $|r_{ui} - \hat{r}_{ui}| \in \{0, 1, 2, 3, 4\}$. The spread of the prediction error looks at how many times the prediction error takes each value in $\{0, 1, 2, 3, 4\}$, and relates them to the number of ratings in the validation set, namely 9,000 in our case. Equation 14 shows how to compute the *share* of correctly predicted ratings (prediction error is 0) in fold t of the cross-validation, where $x_{t,0}$ is the number of predicted ratings that exactly match the actual ratings. In a similar fashion we record the share of prediction errors equal to 1, 2, 3, and 4 in fold t of the cross-validation via $n_{t,1}$ to $n_{t,4}$; in the computation of $n_{t,1}$ to $n_{t,4}$ we use $x_{t,1}$ to $x_{t,4}$, which represent the number of predicted ratings that differ from the actual ratings given by the users by 1 to 4, respectively. We use Equation 15 to average the shares over the ten folds and to obtain an overall value for the share of prediction error 0 via n_0. In a similar way we compute the overall values for the shares of prediction errors 1, 2, 3, and 4 via n_1 to n_4.

$$n_{t,0} = \frac{x_{t,0}}{|K_t|} \qquad (14)$$

[3]http://grouplens.org/datasets/movielens/100k/

$$n_0 = \frac{1}{10}\left(n_{1,0} + n_{2,0} + \ldots + n_{10,0}\right) \tag{15}$$

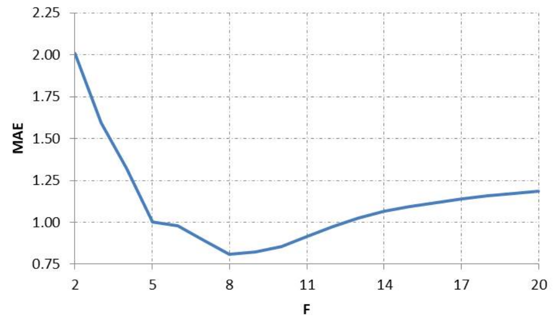
The final quality measure we use to assess the performance of our algorithm is the runtime. To this end, we record the number of seconds from the beginning of the algorithm, throughout the matrix factorization part, until the predictions are obtained and the recommender stops.

Compared to the KBV model, our model has three new variables; compared to the GM model, our model has two extra variables. Since our proposed method is more general and has more degrees of freedom, which enable it to better describe the latent factor space, we expect it to perform better when compared to the other two methods; we expect the MAE of our recommender to be lower and the prediction errors to occur less often. However, the runtime of our recommender will likely be higher, since we propose a more general model, with more tuning parameters.

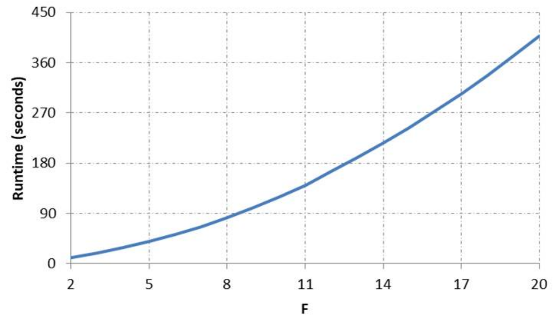
4.3 Parameter Setup

There are four tuning parameters that need to be estimated in such a way as to minimize the expression in Equation 3: the two elastic net parameters on the user factor matrix, ρ_1 and λ_1, and the two elastic net parameters on the movie factor matrix, ρ_2 and λ_2. Further, in order to approximate the solution to Equation 3 we use the stochastic gradient descent (SGD) method, which adds another tuning parameter, namely the learning rate α. Finally, the number of dimensions F of the latent factor space that explains the ratings given by users to movies is another tuning parameter, and needs to be estimated.

The learning rate α is a tuning parameter that determines the size of the steps the SGD algorithm takes to reach the minimum of the expression in Equation 3, thus, α influences the speed of convergence of the algorithm. Moreover, α controls the accuracy of the predictions: a learning rate α that is too large can cause the algorithm to miss/overlook the optimal solution, while an α that is too small can slow down the algorithm and prevent it from finding the optimal solution when there is a fixed number of iterations. Although different values of the learning rate α lead to different sets of predictions, and, thus, to different prediction accuracies, Koren [8] provides empirical evidence that using a fixed learning rate can be beneficial; the researcher finds that fixing $\alpha = 0.001$ works well for the Netflix problem. Moreover, working with a fixed learning rate is a computationally attractive alternative to optimizing it together with the other parameters of the model; this argues in favor of fixing α and attuning the remaining parameters to a fixed learning rate. To this end, we perform a line search for an adequate value of α under the following setup: we fix ρ_1, ρ_2, λ_1, and λ_2 to 0.25, weighting each of the four elastic net components in Equation 3 equally, and we fix F to 8. Under this setup, α was varied between 0.0001 and 0.005, in steps of 0.0001. Using the MAE as the evaluation criterion, we found that $\alpha = 0.001$ minimizes this assessment measure. Moreover, the literature [8] and various simulation studies performed on similar movies datasets also exploit this fixed value for the learning rate in the analyses. Thus, we decided to fix $\alpha = 0.001$ as well, and optimize the rest of the parameters accordingly.



(a)



(b)

Figure 1: Influence of F on prediction accuracy, as measured by the mean absolute error (MAE), and on runtime (expressed in seconds). Plotted for values of F up to 20, Figure (a) shows how the MAE behaves with increasing F, while Figure (b) illustrates how the runtime responds to increasing values of F. The value of F that minimizes the mean absolute error (MAE) assessment criterion is 8.

Apart from fixing the learning rate α, we also fix the number of factors F. A similar argumentation invoking computational effort supports performing a line search for the appropriate number of factors and fixing F to that certain value throughout subsequent analyses. Keeping the value 0.25 for ρ_1, ρ_2, λ_1, and λ_2, and fixing α to 0.001, we performed a line search for a suitable value of F. Under this setup, the number of factors F was varied between 2 and 20. For values of F up to 8, we noticed that the prediction accuracy of our recommender system improves when the number of factors it estimates increases (Figure 1 (a)). For higher values of F, the MAE increases, compared to its minimum value attained for $F = 8$. Moreover, Figure 1 (b) shows that the runtime of the algorithm grows at such a high rate with increasing values of F, that it does not justify increasing the number of factors beyond 8. Additionally, from an intuitive point of view, in the context of movies and viewers' attitude towards movies, we hypothesize that eight key characteristics would allow us to find enough descriptive dimensions to understand and predict movie preferences, therefore we work with $F = 8$.

Thus, we restrict the parameter space of our model in two directions, by imposing $\alpha = 0.001$ and $F = 8$. We estimate the other parameters by four-dimensional grid search: we search for the optimal values of ρ_1, ρ_2, λ_1, and λ_2 in $[0, 0.5]$, using a 0.1 step size. Our four-dimensional grid search analyzes 1,296 4-tuples and chooses the one that leads to the lowest mean absolute error of the predicted ratings; each of the three algorithms we consider has its own optimal parameters. The algorithms were implemented in PHP[4] and run on a machine with Intel Xeon W3680 3.33GHz six core CPU, 12GB of RAM, and 64-bit Windows 10 operating system.

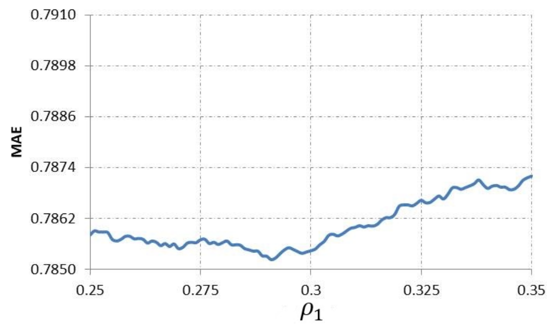
4.4 Results

The 4-tuples $(\rho_1, \rho_2, \lambda_1, \lambda_2)$ that give the lowest MAE for each of the three methods, respectively, are (0.3, 0.0, 0.0, 0.0) for ENetMF, (0.2, 0.2, 0.0, 0.0) for KBV, and (0.2, 0.0, 0.0, 0.1) for GM. Although our approach does not impose any restrictions on the parameters, the algorithm finds that the optimal parameter setup fixes $\rho_2 = 0$, $\lambda_1 = 0$, and $\lambda_2 = 0$. Since ρ_2 and λ_2 are the elastic net parameters acting on the item factor matrix, their optimal values found by our algorithm imply that our proposed approach makes the most accurate predictions when the item factor matrix is not regularized. Moreover, finding $\lambda_1 = 0$ indicates that there is no selection of factors performed on the user item matrix, which implies that users have a certain affinity towards all factors, as hypothesized by Gogna and Majumdar [5].

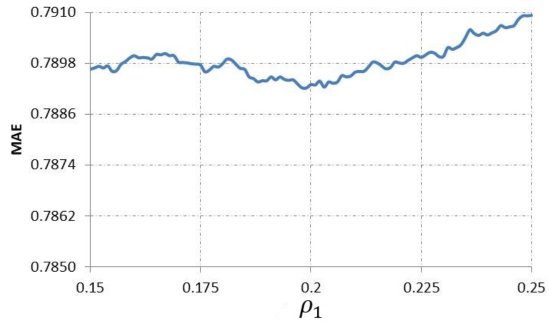
Measuring how close predicted ratings are to real ratings given by the users to the movie items, the MAE is a measure of accuracy used to assess the performance of various recommenders; the MAEs obtained by ENetMF (0.9279), KBV (0.9543), and GM (0.9523) attest the superior prediction accuracy of our proposed model and endorse it as a reliable recommendation tool. Our algorithm performs 2.77% better than KBV, and achieves 2.56% improvement in predictions with respect to GM, both in terms of mean absolute error.

As explained in Section 4.2, the optimal parameters corresponding to each of the three models are obtained following 10-fold cross-validation. This means that each algorithm returns a MAE in each of the ten folds; averaging the ten MAEs corresponding to each algorithm yields the overall MAE of each model in the training and validation phase, corresponding to a certain parameter setup $(\rho_1, \rho_2, \lambda_1, \lambda_2)$. Since the differences between these values are small, we run two separate two-sample paired t-tests to properly assess whether the MAEs differ significantly between the three methods. Compared to the ten MAEs achieved by the KBV recommendation method, the MAEs obtained following our approach tend to be smaller, although the statistical evidence is not very strong ($p = 0.051$). Further, compared to the ten MAEs achieved by the GM recommendation method, the MAEs obtained following our approach also tend to be smaller, however, the statistical evidence is not compelling ($p = 0.08$).

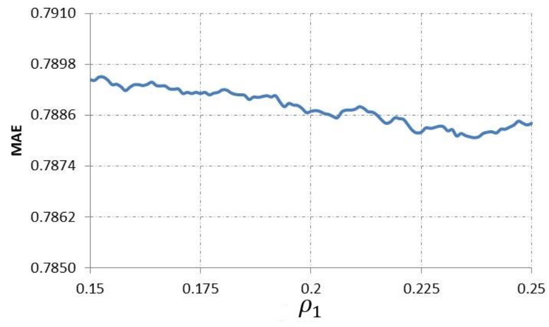
The mean absolute error provides an overview of the prediction performance and accuracy of a recommender system. To zoom in on the predicted ratings, we analyze how close individual predictions fall from the real ratings. Our proposed approach predicts 95.02% of the ratings with an error of at most two points (prediction error is either 0, 1, or 2), while KBV predicts 94.39% of the ratings with the same accuracy, and GM predicts only 94.41% of the ratings with the same precision. Although 0.01% of the ratings in our working dataset of size 100K amounts to ten ratings, the number rapidly increases with the size of the dataset under scrutiny; therefore, an increase of 0.01% in correctly predicted ratings is a considerable improvement for very large datasets.



(a)



(b)



(c)

Figure 2: MAE fluctuations corresponding to small changes in ρ_1 around its optimal value. Figure (a) illustrates MAE fluctuations caused by ENetMF (optimal $\rho_1 = 0.3$), Figure (b) exhibits MAE fluctuations caused by the KBV method (optimal $\rho_1 = 0.2$), while Figure (c) reveals MAE fluctuations caused by the GM method (optimal $\rho_1 = 0.2$).

Comparison of the algorithms across the runtime dimension is done by recording the time (in seconds) it takes each method to evaluate a single 4-tuple $(\rho_1, \rho_2, \lambda_1, \lambda_2)$. Runtime increases with the number of parameters to be estimated: KBV estimates one

[4]Code available at https://tinyurl.com/yb6ddww4.

parameter, since the model imposes that ρ_1 and ρ_2 should be equal, and has the lowest runtime, 61.49; GM estimates two parameters and requires 61.80 seconds; finally, ENetMF imposes no restrictions and estimates four parameters, and has the highest runtime, 62.27.

The only parameter that is not restricted by any of the three recommendation methods is ρ_1 (λ_1 and λ_2 are restricted to 0 in the KBV method, while ρ_2 and λ_1 are restricted to 0 in the GM method). A sensitivity analysis of how small changes in this parameter affect the predictive performance of the three algorithms would shed light on which recommendation method is the most sensitive to parameter fluctuations. Figure 2 shows how the MAE reacts to small changes in ρ_1 around its optimal value, for each of the three recommendation methods. Using a 0.001 step size, we consider changes of at most ± 0.05 around ρ_1's optimal values: they are 0.3 for our proposed method, 0.2 for the KBV algorithm, and 0.2 for the GM recommender. For each of the three methods we consider an interval of length 0.1, centered around ρ_1's corresponding optimal value, and evaluate how these changes affect the predictive performance of the recommenders, as measured by their respective MAE. Note that in Figure 2 the optimal value of ρ_1 that we found previously does not lead to the lowest MAE; the reason is that these plots zoom in extensively around a specific value of ρ_1, and can thus approximate the optimal value of ρ_1 more accurately, while the resolution of the grid search we performed to find the optimal parameters was inferior, for runtime and computational burden reasons. High resolution grid search is the only support we have in finding the best parameter estimates, as the minimization of an expression with an elastic net penalty has no closed form solution. The reason is that the elastic net penalty does not have partial derivatives at zero for any of the penalized coefficients, due to the L_1 component of the penalty [21]. However, the L_2 component of the elastic net penalty ensures strict convexity, which guarantees the existence of a minimum to the expression in Equation 3. Therefore, we optimize using the stochastic gradient descent procedure, whereby we alternate between updating the user factor matrix P and the item factor matrix Q, and we use the grid resolution as an indication of how accurate our estimates are. The plots in Figure 2 show that the range of the MAE of our proposed algorithm (Figure 2 (a)) is comparable to the ranges corresponding to the other two methods (Figure 2 (b) and (c)), attesting the stability of all three approaches; this kind of stability is a desirable characteristic of a recommender system, since it shows that the prediction performance of the underlying algorithm is less sensitive to possible inaccuracies in the estimated parameter values.

4.5 Discussion

In this section we revisit the two hypotheses we formulated in Section 2 and analyze to what extent they are confirmed/infirmed based on the results of Section 4.4.

An interesting result that our research revealed is that, although we do not impose restrictions on the elastic net parameters that regularize the item factor matrix, our algorithm finds that their optimal values are 0. Note that the elastic net parameter λ_2 accounts for the lasso contribution to the regularization, thus it is responsible for the selection of coefficients. Finding that $\lambda_2 = 0$ invalidates our first hypothesis stated in Section 2, namely that movies can only be characterized by certain factors. An interpretation that could be given to the fact that $\lambda_2 = 0$ is that there seems to be no need for a recommendation method to select a certain subset of factors to describe movies; instead, all factors found by ENetMF characterize movies to certain extents. Since factors are complex features that describe the content of movies, and not plain movie genres, it is plausible that each movie is a mix of (possibly) all factors, with different weights; the salience of a factor in a movie item translates into the magnitude of the corresponding coefficient in the item factor matrix.

In presenting our second hypothesis in Section 2, we deviated from the reasoning of Gogna and Majumdar [5], and stated that users might not have an inclination towards all factors, and that their preferences are better described by subsets of the latent factors revealed by a recommendation tool. Finding $\lambda_1 = 0$ invalidates our second hypothesis and suggests that users do indeed a certain affinity towards all underlying features describing movies.

We mentioned in Section 2 that one of the advantages of using the elastic net regularization over ridge or lasso is its ability to perform group variable selection. Our research allows us to make a surprising empirical observation in this direction, namely that the sparsity property and the grouping effect usually inflicted by the L_1 component of the elastic net are not that transparent in the case of our recommender system. More precisely, the coefficients of the variables of our model (the eight latent factors) do not make the inclusion/exclusion decision that would produce the sparse solutions generally induced by the elastic net; even under the influence of the elastic net, our model includes all eight variables. A closer look at the values of the optimal parameters gives us an idea why this is the case. Parameters λ_1 and λ_2 quantify the lasso contribution to the elastic net regularization on the user and the item factor matrices, respectively, and control the number of null coefficients. We found $\lambda_1 = 0$, whose value is responsible for the parameter's inability to perform variable selection. Additionally, we found $\lambda_2 = 0$, which effectively means there is no selection operator acting on the item factor matrix. Also, the variables cannot be split into groups according to comparable values of their corresponding coefficients, which prevents us from observing the grouping effect we expected.

5 CONCLUSIONS

The aim of our research was to develop a recommender system that uses matrix factorization with an elastic net regularization, and to evaluate its performance on the MovieLens 100K dataset, a collection of 100,000 ratings (1-5) from 943 users on 1,682 movies, against the KBV model [9] and the GM model [5]. There are two main hypotheses that motivated our research. Our first hypothesis postulates that not all factors included in the model are necessary for describing movie characteristics. Therefore, the L_1 contribution of the elastic net should act accordingly, and select those underlying factors that are needed to explain movie features, through parameter λ_2. However, our model finds that $\lambda_2 = 0$, which invalidates our first hypothesis and suggests that each movie item is a mix of all latent factors, since the parameter that grants the elastic net its variable selection power is null. Our second hypothesis states that users might not have a preference for all factors. Our research infirms this statement by finding $\lambda_1 = 0$; the elastic net regularization acting on the user factor matrix is unable to perform factor selection, given that the value we found for λ_1 is null. The fact that $\lambda_1 = 0$ shows that user preferences do not need the selection ability of the elastic net in order to be correctly represented using subsets of the latent factors; in fact, our research indicates that users tend to have a certain preference towards all underlying factors that characterize movie items.

In terms of mean absolute error (MAE), our algorithm performs best, achieving 2.77% better prediction accuracy compared to the KBV model; however, the improvement is not statistically significant, based on a p-value of 0.051. Compared to the GM model, our approach obtains a 2.56% increase in accuracy of predictions, although the improvement is not statistically significant (p-value 0.08). Moreover, our method performs better in terms of the prediction errors it makes for individual users: we manage to predict 95.02% of the ratings with an error of at most two points from the real ratings, while KBV achieves the same accuracy for 94.39% of the ratings, and GM for 94.41%. Our method is thus able to describe the latent factor space better and thus outperform the other two methods with respect to prediction accuracy.

There are a number of possible future research directions that build up on the current state of our recommender and could potentially improve our prediction accuracy by adjusting some of the specifications of our model. First, allowing the parameter α to vary alongside the other four elastic net parameters (ρ_1, ρ_2, λ_1, and λ_2) could improve the performance of our recommender through a better optimization of our objective function. Alternatively, one could use a genetic algorithm to find the optimal five-tuple $(\alpha, \rho_1, \rho_2, \lambda_1, \lambda_2)$, as the efficiency of this class of algorithms has been studied and confirmed by previous research [4, 7, 12]. Secondly, increasing the number of latent factors to be discovered beyond eight could impact results in three ways: (1) it might be easier for the elastic net regularization to find groups of highly correlated variables when there are more explanatory variables included in the model; (2) a model with more variables has a better chance of forming a sparse representation under the influence of the elastic net penalty, since it might be more straightforward to find unnecessary factors and leave them out of the model when there are more variables to choose from; (3) an increased number of underlying factors has the potential of better represent user choices and movie features, and, thus, lead to more accurate predictions.

REFERENCES

[1] Christopher R. Aberger. 2016. Recommender: An Analysis of Collaborative Filtering Techniques. https://pdfs.semanticscholar.org/53bd/928ceaab4572594590970b877da1278b7a79.pdf.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734–749.

[3] Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based Latent Factor Models. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2009)*. ACM, 19–28.

[4] Jesus Bobadilla, Fernando Ortega, Antonio Hernando, and Javier Alcala. 2011. Improving Collaborative Filtering Recommender System Results and Performance Using Genetic Algorithms. *Knowledge-Based Systems* 24, 8 (2011), 1310–1316.

[5] Anupriya Gogna and Angshul Majumdar. 2014. Distributed Elastic Net Regularized Blind Compressive Sensing for Recommender System Design. In *Proceedings of the 20th International Conference on Management of Data (COMAD 2014)*. Computer Society of India, 29–37.

[6] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.

[7] Kyoung-jae Kim and Hyunchul Ahn. 2008. A Recommender System Using GA K-Means Clustering in an Online Shopping Market. *Expert Systems with Applications* 34, 2 (2008), 1200–1209.

[8] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2008)*. ACM, 426–434.

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer Society* 42, 8 (2009), 30–37.

[10] Bipul Kumar. 2016. A Novel Latent Factor Model for Recommender System. *Journal of Information Systems and Technology Management* 13 (2016), 497–514.

[11] Kenneth Lange. 2004. *The MM Algorithm*. Springer New York, 119–136.

[12] Hsiang-Hsi Liu and Chorng-Shyong Ong. 2008. Variable Selection in Clustering for Marketing Segmentation Using Genetic Algorithms. *Expert Systems with Applications* 34, 1 (2008), 502–510.

[13] Andriy Mnih. 2011. Taxonomy-Informed Latent Factor Models for Implicit Feedback. In *Proceedings of the 2011 International Conference on KDD Cup 2011*. JMLR.org, 169–181.

[14] James M. Ortega and Werner C. Rheinboldt. 1970. *Iterative Solution of Nonlinear Equations in Several Variables*. Vol. 30. Siam.

[15] Andre V. Rodrigues, Alipio Jorge, and Ines Dutra. 2015. Accelerating Recommender Systems Using GPUs. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC 2015)*. ACM, 879–884.

[16] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. 2000. Application of Dimensionality Reduction in Recommender System - A Case Study. In *Proceedings of the Web Mining for E-commerce Workshop (WebKDD 2000)*. ACM, 82–90.

[17] Shai Shalev-Shwartz and Ambuj Tewari. 2011. Stochastic Methods for L1-regularized Loss Minimization. *The Journal of Machine Learning Research* 12 (2011), 1865–1892.

[18] SINTEF. 2013. Big Data, for Better or Worse: 90% of World's Data Generated over Last Two Years. https://www.sciencedaily.com/releases/2013/05/130522085217.htm.

[19] Robert Tibshirani. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 267–288.

[20] Andrey N. Tikhonov. 1943. On the Stability of Inverse Problems. In *Doklady Akademii Nauk SSSR*, Vol. 39. 195–198.

[21] Michael J. Wurm, Paul J. Rathouz, and Bret M. Hanlon. 2017. Regularized Ordinal Regression and the ordinalNet R Package. (2017).

[22] Tong Zhao, Julian McAuley, and Irwin King. 2015. Improving Latent Factor Models via Personalized Feature Projection for One Class Recommendation. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015)*. ACM, 821–830.

[23] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. 2008. *Large-Scale Parallel Collaborative Filtering for the Netflix Prize*. Springer Berlin Heidelberg, Chapter 32, 337–348.

[24] Hui Zou and Trevor Hastie. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67, 2 (2005), 301–320.