Article

Evolving Matrix-Factorization-Based Collaborative Filtering Using Genetic Programming

Raúl Lara-Cabrera ^{1,*} , Ángel González-Prieto ^{1,2,†} , Fernando Ortega ^{1,†} and Jesús Bobadilla ^{1,†}

¹ Departamento de Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, 28031 Madrid, Spain; angel.gonzalez.prieto@upm.es (Á.G.-P.); fernando.ortega@upm.es (F.O.); jesus.bobadilla@upm.es (J.B.).

² Instituto de Ciencias Matemáticas, CSIC-UAM-UC3M-UCM, 28049 Madrid, Spain.

* Correspondence: raul.lara@upm.es.

† These authors contributed equally to this work.

Received: 24 December 2019; Accepted: 13 January 2020; Published: 18 January 2020

Abstract: Recommender systems aim to estimate the judgment or opinion that a user might offer to an item. Matrix-factorization-based collaborative filtering typifies both users and items as vectors of factors inferred from item rating patterns. This method finds latent structure in the data, assuming that observations lie close to a low-dimensional latent space. However, matrix factorizations have been traditionally designed by hand. Here, we present Evolutionary Matrix Factorization (EMF), an evolutionary approach that automatically generates matrix factorizations aimed at improving the performance of recommender systems. Initial experiments using this approach show that EMF generally outperforms baseline methods when applied to MovieLens and FilmTrust datasets, having a similar performance to those baselines on the worst cases. These results serve as an incentive to continue improving and studying the application of an evolutionary approach to collaborative filtering based on Matrix Factorization.

Keywords: genetic programming; recommender systems; collaborative filtering; matrix factorization

1. Introduction

In the digital world in which we currently live, we are constantly flooded with large amounts of information that we are unable to assimilate. Faced with such a large amount of information, those systems capable of filtering and customizing it for each user have become increasingly important in recent years. It was in this context that the Recommender Systems (RSs) were born, which have become popular in the scientific community if we look at the wide range of publications related to them [1–4].

A RS is a subcategory of the information filtering process that aims to estimate the judgment or opinion that a customer might offer to an item. Whether it is generating playlists for Spotify, recommending products to buy on Amazon, or suggesting the next series you will see on Netflix, RSs are a part of our daily lives. In other words, a RS should be defined as an intelligent system capable of providing each user with a personalized list of products or services that may be of interest [5,6].

These systems base their functionality on the information they acquire from their users [7]. A RS might gather information from the user explicitly, i.e., collecting users' ratings, or implicitly, i.e., analyzing the users' behavior. Moreover, additional data such as demographic information and social network relationships might be included to improve the performance of the system.

Out of all the dynamics that rule the functioning of a RS, such as the type of data it feeds on, the approach used—like probabilistic, Bayesian or bio-inspired—and whether the system is memory-

or model-based. The process that has the highest impact on the performance is the filtering and, therefore, the algorithm used for it.

One of the most widely used filtering techniques is Collaborative Filtering (CF) [8–10]. The basis of such algorithms is matching people based on similar interests. To do so, these techniques collect human ratings (also known as judgments) for items in a given domain and then match together those people who share the same preferences or inclinations. Users of a CF system distribute their opinions and ratings on each item they use in a way that other users of the system are able to make an informed decision on which items to buy, consume, or use.

There are several implementations of CF, with Matrix Factorization (MF) being one of the most popular families [11,12]. This family of implementations, which achieves both predictive accuracy and good scalability, models both users and items as vectors of factors inferred from item rating patterns. This way, high coincidence between item and user factors brings the system to a recommendation.

In the last years, the scientific community has focused its efforts on investigating new ways of improving MF to achieve good performance of the RS. In this spirit, several new MF methods have been proposed recently. Each of these new proposals postulate hypotheses about how the users of the RS interact with the items. However, these hypotheses are formulated based on the previous knowledge that the researchers may have on the particular domain of applicability, such as consumption habits or dynamics of the interest of users. Hence, in order to tackle the linearity hypothesis presented in the MF methods, the researchers modified the underlying probabilistic assumptions to add nonlinear adjustments introduced ad hoc. Therefore, the newly built recommendation models, based on the expertise of their creator, are intrinsically biased.

In order to overcome this problem, in this paper, we take the first steps towards the automatic generation of MF models through the use of Genetic Programming (GP), an optimization algorithm belonging to the family of evolutionary computing. Instead of introducing our previous knowledge to the MF model, we let GP extract the required knowledge from the data to build the MF model. In this way, the main contribution of this paper is to stress this idea by showing the feasibility of using GP to provide an efficient search method on the space of MFs. This gives rise to a better RSs that outperform the previous methods presented in the literature. This technique has been named after Evolutionary Matrix Factorization (EMF).

2. Materials and Methods

This section is devoted to describe techniques, algorithms, metrics, and processes used by EMF to automatically generate varied aggregation functions to improve the performance of MF models.

2.1. Matrix Factorization

MF, as in the traditional meaning of the term, consists of splitting a matrix $A \in \mathbb{R}^{n \times m}$ into the product of two other matrices $A = P \times Q$, $P \in \mathbb{R}^{n \times k}$, $Q \in \mathbb{R}^{k \times m}$. In other words, matrix factorization is a way of reducing a matrix into its constituent parts. This transformation is particularly convenient because it typically gives a more compact representation, hence making it easier to calculate more complex matrix operations.

When applied to CF, MF is an embedding model. Given the sparse rating matrix $R \in \mathbb{V}^{n \times m}$ —with n being the number of users (rows), m being the number of items (columns), and $\mathbb{V}$ being the feasible ratings that a user is able to give to an item—the model learns both a user projection matrix $P \in \mathbb{R}^{n \times k}$, where each row is a projection for a user; and an item projecting matrix $Q \in \mathbb{R}^{m \times k}$, with each row representing an projection for an item. The projections are learned such that the product $P \cdot Q^T$ is a good approximation of the ratings matrix R .

Note that MF generally provides a more compact representation than learning the full matrix, as the projection dimension k is usually much smaller than m and n . Hence, MF finds latent structure in the data, assuming that observations lie close to a low-dimensional latent space.

To learn the embeddings, MF becomes an optimization problem, whose target is to minimize the following objective function:

$$\min_{p_{u,1}, \dots, p_{u,k}, q_{i,1}, \dots, q_{i,k}} \sum_{(u,i) \in R} (r_{u,i} - h(p_{u,1}, \dots, p_{u,k}, q_{i,1}, \dots, q_{i,k}))^2 + \lambda \left(\sum_{\alpha=1}^k p_{u,\alpha}^2 + q_{i,\alpha}^2 \right). \quad (1)$$

Here, R represents the set of (u, i) pairs for which the rating of the user u to the item i , denoted $r_{u,i}$, is known. The function $h(p_{u,1}, \dots, p_{u,k}, q_{i,1}, \dots, q_{i,k})$ is a mathematical function that aggregates user and item latent factors to compute model predictions, and λ is a regularization hyperparameter to avoid overfitting.

Stochastic Gradient Descent (SGD) [13], an iterative method for optimizing an objective function with suitable smoothness properties such as differentiability, is usually applied as the optimization algorithm to minimize the aforementioned function and learn the embeddings. SGD's iterative process updates $p_{u,\alpha}$ and $q_{i,\alpha}$ according to Equations (2)–(4).

$$e_{u,i}^t = r_{u,i} - h(p_{u,1}^t, \dots, p_{u,k}^t, q_{i,1}^t, \dots, q_{i,k}^t), \quad (2)$$

$$p_{u,\alpha}^{t+1} = p_{u,\alpha}^t + \gamma \left(e_{u,i}^t \cdot \frac{\partial h}{\partial p_{u,\alpha}} - \lambda p_{u,\alpha}^t \right), \quad (3)$$

$$q_{i,\alpha}^{t+1} = q_{i,\alpha}^t + \gamma \left(e_{u,i}^t \cdot \frac{\partial h}{\partial q_{i,\alpha}} - \lambda q_{i,\alpha}^t \right), \quad (4)$$

where γ is a hyperparameter to control the learning rate. It is customary to take, as aggregation function h , the usual inner product of the latent factors:

$$h(p_{u,1}, \dots, p_{u,k}, q_{i,1}, \dots, q_{i,k}) = \sum_{\alpha=0}^k p_{u,\alpha} \cdot q_{i,\alpha}. \quad (5)$$

In this paper, we break from with the traditional assumption that the users' and items' factors must be combined in a linear fashion (i.e., using the inner product) and propose EMF, an evolutionary framework to generate varied aggregation functions h in order to improve the quality of the predictions made by a CF RS based on MF. How EMF explores and generates different h functions is described in Section 2.2.

2.2. Genetic Programming

Evolutionary Algorithms (EAs) [14] are a method of optimization and search based on populations of individuals, which represent candidate solutions to a given problem. By means of an iterative process inspired by the natural selection mechanism, new solutions are generated from the existing ones and solutions (i.e., individuals) that show a bad performance are discarded.

The performance that guides the selection process is known as the fitness value of the individual. Selecting an adequate fitness function is crucial to success when designing an EA because it assigns a performance value to an individual which also guides the selection process. Hence, a bad fitness function makes it harder for the EA to find good solutions.

Another important step in the design of an EA is choosing a suitable set of variation operators. These functions are responsible for taking several individuals as inputs and producing new ones that are slightly different. The main idea of this process is to introduce diversity within the population of solutions.

GP [15] is a kind of algorithm belonging to the family of evolutionary computing. The distinctive feature of GP is the internal representation of the candidate solutions, which are coded with a tree structure. The internal nodes represent operations and functions of various kinds, while the leaf nodes correspond to input variables and constant values. Therefore, a candidate solution represents

a computer program, since the nodes correspond to functions and operators, whose results serve as inputs for other functions that also operate on certain inputs.

One of the features of GP algorithms is their ability to search and find novel solutions to the problem, subject to optimization. CF based on MF has exhibited a very good performance, so we planned to study whether we would be able to find aggregation functions of the latent factors h other than the usual inner product, that achieves a better performance.

EMF, our proposed method, is a GP algorithm in which the individuals of the population are aggregated functions h , as described in Section 2.1. To do so, leaf nodes of an individual are the projections of the users' ratings in the k -dimensional latent space: $p_{u,\alpha}, q_{i,\alpha}$. In addition to these, we included two constant values that might be considered as leaf nodes: 0.0 (Zero) and 1.0 (One), in order to increase the expressiveness in the aggregation functions h generated by the algorithm.

Regarding the internal nodes, we decided to include both linear and nonlinear mathematical operators. The operations are shown in Table 1.

Table 1. Mathematical operators included in the evolutionary process.

Operator	Arity	Function	Symbol
Sine	1	$\sin(x)$	<code>sin</code>
Cosine	1	$\cos(x)$	<code>cos</code>
Arctangent	1	$\arctan(x)$	<code>atan</code>
Exponential	1	$\exp(x)$	<code>exp</code>
Logarithm	1	$\log(x)$	<code>log</code>
Inverse	1	$\frac{1}{x}$	<code>inv</code>
Sign	1	$-x$	<code>-</code>
Addition	2	$x + y$	<code>+</code>
Subtraction	2	$x - y$	<code>-</code>
Multiplication	2	$x \times y$	<code>*</code>
Power	2	x^y	<code>pow</code>

To measure the quality of the solutions (i.e., the fitness function), we used the Mean Absolute Error (MAE) Equation (6) metric, defined as the averaged absolute difference between the real rating ($r_{u,i}$) and the prediction performed by the aggregation function h :

$$MAE = \frac{1}{\#R^{test}} \sum_{(u,i) \in R^{test}} |r_{u,i} - h(p_{u,1}, \dots, p_{u,k}, q_{i,1}, \dots, q_{i,k})|, \quad (6)$$

with R^{test} made up of those ratings from user u to item i ($r_{u,i}$) that were selected as test.

Although the fitness function has been set up, there is a following explanation about how this fitness value is assigned to each individual. First, the aggregation function encoded by the individual is derived, as required by the SGD algorithm. To find the derivative of each individual, we built a Java library for symbolic derivation [16]. Then, the MF model is trained following the procedure described in Section 2.1, taking into account that the individual being evaluated is the aggregation function h of Equation (1). Finally, the quality of the trained model is evaluated against a test split of the dataset, computing the MAE and assigning this value as the fitness of the individual.

On top of the aforementioned elements of the algorithm is the so-called breeding pipeline, that is, the precise process the individuals of the population follow during each generational step within the algorithm. The breeding pipeline of EMF is devised in Figure 1.

Each generational step starts with the selection of those individuals that would generate offspring (i.e., new solutions). The tournament selection randomly chooses two pairs of individuals and selects the one with the best fitness out of the pairs. Each selected individual becomes a parent. Then, new individuals are created with a probability of $p_R = 0.5$ by mating the parents. In our approach, we used a single-node crossover, which randomly selects one node of each parent and then exchanges the subtrees whose roots are the selected nodes. During the variation stage, the offspring is mutated

with a probability $p_V = 0.1$. The mutation of an individual consists of randomly selecting a node within the tree and then replacing it with another randomly selected operator that suits the former node's arity. New individuals are then evaluated and assigned a fitness value. Finally, the algorithm selects which of the individuals should survive to the next generational step. EMF always selects the top two individuals sorted by their fitness values and then completes the size of the new population by selecting the remaining individuals in a random manner. Note that the only stop criterion of our proposal is to reach a maximum number of generations $MAX_{iter} = 150$.

Figure 1. Breeding pipeline of Evolutionary Matrix Factorization (EMF) algorithm.

The use of evolutionary algorithms along RSs is not new, as it is possible to find many papers that use this kind of algorithm to enhance RSs [17,18]. However, according to the surveys on that topic, most of the papers from the state of the art are devoted to treat the weights of both the features and the latent models as the individual of the evolutionary process. Those blind optimization methods are very different to EMF, in which the subject of optimization is the aggregated function h used to guide the matrix factorization method.

On the other hand, there are a few papers that use GP to function learning, which is what EMF does. Belém et al. [19] used GP to find ranking functions based on a list of attributes related to metrics of tag relevance. Although the proposal shows promising results, the algorithm subject to optimization is a tag RS, which differs from our approach in that EMF optimizes collaborative filtering RSs.

Another use of GP within the RS context can be found in [20]. Anand and Bharadwaj proposed the use of GP to derive optimal transformation functions to map ratings to preferences scores while avoiding biases. The main difference, with respect our approach, is that the optimized functions in the former are used as some kind of preprocessing step. Moreover, although they used CF, the underlying technique is not MF.

Guimarães et al. [21] presented an evolutionary approach based on GP that “evolves a ranking function that is a combination of measures or (sub-)components of measures commonly used for collaborative filtering, such as the average rating of an item, item popularity, distance of user/item to the k nearest neighbors, among others”. Their approach differs from ours in that the former builds optimized functions from previously aggregated metrics, while ours lets the algorithm to search for these aggregations directly from the features.

As a final remark, EMF is based on the Jenetics framework [22], an evolutionary computation library written in Java that has no runtime dependencies to other libraries, except the Java 8 runtime itself.

2.3. Experimental Setup

In order to evaluate the performance of the proposed method, we have designed the experiments described in this subsection. All these experiments have been carried out using two of the most popular datasets in the CF field: MovieLens [23] and FilmTrust [24]. The main properties of these datasets can be found in Table 2. We have chosen these datasets due their reduced number of ratings, since the proposed method's computational cost is too expensive to be evaluated with larger datasets.

Table 2. Main properties of the datasets used in experiments.

Dataset	#users	#items	#ratings	Rating Scale
MovieLens	943	1682	100,000	1–5
FilmTrust	1508	2071	35,497	0.5–4.0

The main objective of our experimental evaluation is to compare the performance of EMF with respect to the standard MF techniques used for CF. We have selected the following MF techniques as baselines: PMF [25], the most popular factorization model for CF-based RS; BiasedMF [11], a factorization model that incorporate biases to represent the variations in the rating behavior of users and items; NMF [26], a model that factorizes the rating matrix into two new non-negative factors matrices; and BNMF [27], a model that factorizes the rating matrix into two non-negative matrices with an understandable probabilistic meaning.

Selected baselines contain hyperparameters that must be tuned for each dataset. In order to maximize the accuracy of the predictions computed for each baseline, we have tuned these hyperparameters by performing a grid search approach. As a result of this search, the hyperparameters has been fixed to the following: for PMF, $k = 6$, $\lambda = 0.085$, and $\gamma = 0.01$ for both MovieLens and FilmTrust; for BiasedMF, $k = 6$, $\lambda = 0.06$, and $\gamma = 0.01$ for MovieLens and $k = 12$, $\lambda = 0.095$, and $\gamma = 0.03$ for FilmTrust; for NMF, $k = 6$ for MovieLens and $k = 4$ for FilmTrust; and for BNMF, $k = 6$, $\alpha = 0.9$, and $\beta = 5$ for MovieLens and $k = 4$, $\alpha = 0.8$, and $\beta = 5$ for FilmTrust.

The proposed method, EMF, also contains two hyperparameters to control the SGD process. These hyperparameters are set to $\gamma = 0.001$ and $\lambda = 0.095$ for MovieLens and $\gamma = 0.035$ and $\lambda = 0.095$ for FilmTrust. These values have been chosen empirically. Furthermore, MF requires to set the number of latent factors (k). In the proposed approach, this hyperparameter must be interpreted as the maximum number of latent factors to be used during the GP optimization, since the GP algorithm may discard some of these latent factors to optimize the fitness function. We have set $k = 6$ for MovieLens and $k = 10$ for FilmTrust.

All experiments has been carried out by splitting datasets into training and test sets. In both datasets, we have randomly selected 80% of the users and 80% of the items as training users and items, respectively. Remaining users and items has been labeled as test.

The main goal of the proposed method is to improve the accuracy of the predictions performed with MF-based CF by evolving the aggregation function h that combines the user and item latent factors using GP. To quantify this improvement, we measured the accuracy of the predictions computed by EMF and selected baselines using MAE and Mean Squared Error (MSE) quality measures.

MAE has been defined previously in Section 2.2. The MSE (Equation (7)) is defined as the averaged squared difference between the real rating ($r_{u,i}$) and the one computed by the aggregation function h :

$$MSE = \frac{1}{\#R^{test}} \sum_{\langle u,i \rangle \in R^{test}} (r_{u,i} - h(p_{u,1}, \dots, p_{u,k}, q_{i,1}, \dots, q_{i,k}))^2, \quad (7)$$

where R^{test} represents the set of (u, i) pairs for which the rating of the user u to the item i ($r_{u,i}$) has been selected as test.

In order to ensure the convergence of the evolutionary method towards a definite solution, we have screened the time evolution of the sequence of best aggregate functions. Recall, as a result

of the EMF algorithm, we get a sequence of real valued aggregate functions h_n , for $1 \leq n \leq MAX_{iter}$. We have tested whether this sequence of functions converges in a functional sense.

For this purpose, we have fixed a functional space in which we can test such a convergence. A standard choice is to consider the L^p -spaces for $1 \leq p < \infty$. For the convenience of the reader, let us recall some basic properties of these spaces (for further information, see [28], Chapter 3). Given a measurable space Ω , and a measurable function $f : \Omega \rightarrow \mathbb{R}$, its L^p -norm is given as

$$\|f\|_p = \left(\int_{\Omega} |f|^p dx \right)^{1/p}. \quad (8)$$

In that case, $L^p(\Omega)$ is formed by such functions with finite L^p -norm. Given $f, g \in L^p(\Omega)$, its distance is defined as

$$d_p(f, g) = \|f - g\|_p = \left(\int_{\Omega} |f - g|^p dx \right)^{1/p}. \quad (9)$$

In particular, observe that if Ω is a finite set with the counting measure, then the L^1 -norm is the MAE distance and the L^2 -norm is the (square root of the) MSE distance.

In this sense, a sequence of functions f_n is said to converge to a limit function f if $d_p(f, f_n) \rightarrow 0$ when $n \rightarrow \infty$. An equivalent test of convergence that requires no knowledge of the limiting function f is the Cauchy test. It states that the sequence f_n is convergent if and only if $d(f_n, f_m) \rightarrow 0$ when $n, m \rightarrow \infty$. In other words, f_n is convergent if the values of the sequence become closer and closer with time.

In our case, as the results of the MF method are expected to be normalized due to the regularization hyperparameter, we have taken $\Omega = [-1/2, 1/2]^{2k}$, with the number of hidden factors of the matrix factorization k , endowed with the usual Lebesgue (i.e., arc-length) measure. In this case, it is a well-known fact that the distances d_p are nested, i.e., $d_q \leq d_p$ for $p \leq q$ [28]. Hence, a standard choice is to fix $p = 2$. Motivated by the previous discussion, for the sequence of aggregation functions h_n generated by the evolved grammar, we have computed the set of distances

$$d_2(h_n, h_{n-w}) = \left(\int_{[0,1]^{2k}} (h_n - h_{n-w})^2 dx_1 \dots dx_{2k} \right)^{1/2} \quad (10)$$

for $1 \leq w \leq W$, where W is the width of window of screening considered. With these quantities, we have computed the moving-window average

$$D_W(n) = \frac{1}{W} \sum_{w=1}^W d_2(h_n, h_{n-w}). \quad (11)$$

By the Cauchy test, we have tested the convergence of the sequence of best individuals of the population along the generations by checking if $D_W(n) \rightarrow 0$ when $n \rightarrow \infty$.

As we previously mentioned, all experiments has been performed using the CF4J [29] and Jenetics [22] Java frameworks. CF4J has been used to encode the CF algorithms used in the experiments, and Jenetics has been used to configure and perform GP-based optimization. Furthermore, we have developed a Java library that performs symbolic derivation of the individuals generated by the GP algorithm [16]. This library allows us to train EMF using an SGD approach, as mentioned in Section 2.1. Source code used to perform all the experiments can be accessed through GitHub's project page (<https://github.com/ferortega/evolutionary-matrix-factorization>).

3. Results

In this section, we discuss the experimental results obtained after fulfilling the experiments described in Section 2.3.

3.1. Quality of the Recommender System Predictions

Tables 3 and 4 contain the MAE and MSE of the predictions computed for tested datasets. As we can observe, the proposed method (EMF) has been executed 10 times in order to avoid fluctuations caused by random initialization of the optimization process. To sum up all these executions, we have added three additional rows to the table: one that contains the lowest value achieved (best), one that contains the highest value achieved (worst) and a last one that contains the average of all executions (avg).

On one side, Table 3 contains the quality of the predictions performed for MovieLens dataset. On the one hand, the method that obtains better results for the MAE is BiasedMF. However, in 6 of the 10 executions made for EMF (EMF-2, EMF-4, EMF-5, EMF-8, EMF-9, and EMF-10), the difference between EMF and BiasedMF is negligible, so we can consider that the quality of the predictions is equivalent. On the other hand, the method that gets the best results for MSE is EMF-4. In addition, EMF outperforms the best baseline (BiasedMF) in 4 of the 10 executions made (EMF-2, EMF-4, EMF-6, and EMF-10). These results demonstrate that the proposed method tends to produce fewer large differences between the real rating and the predicted ones.

On the other side, Table 4 contains the quality of the predictions performed for FilmTrust dataset. We can observe that EMF-10 provides the best results for MAE and MSE. Furthermore, the proposed method overcomes all tested baselines in all executions performed for both quality measures.

Table 3. Mean Absolute Error (MAE) and Mean Squared Error (MSE) in predictions for MovieLens dataset.

Method	MAE	MSE
PMF	0.7225	0.8492
BiasedMF	0.7160	0.8406
NMF	0.7672	0.9867
BNMF	0.7500	0.8860
EMF-1	0.7256	0.8728
EMF-2	0.7195	0.8332
EMF-3	0.7210	0.8427
EMF-4	0.7197	0.8282
EMF-5	0.7195	0.8592
EMF-6	0.7220	0.8377
EMF-7	0.7255	0.8721
EMF-8	0.7193	0.8442
EMF-9	0.7161	0.8441
EMF-10	0.7163	0.8381
EMF (best)	0.7161	0.8282
EMF (worst)	0.7256	0.8728
EMF (avg)	0.7205	0.8472

3.2. Results of the Evolutionary Method

In order to analyze the final aggregation functions h generated by the GP algorithm, we have built graphical tree representations of the best individuals for each dataset. In Figure 2, we plot the best individual for the MovieLens dataset (EMF-4). On the other hand, in Figure 3, we plot the best individual for the FilmTrust dataset (EMF-10). Furthermore, for the sake of completeness, the rest of the individuals have been included in Appendix A. Recall that the correspondence between node symbols and operators are defined in Table 1.

Figure 2. Tree representation of the best individual (EMF-4) achieved for the MovieLens dataset. The correspondence between node symbols and operators are defined in Table 1.

Table 4. MAE and MSE in predictions for FilmTrust dataset.

Method	MAE	MSE
PMF	0.7514	1.1321
BiasedMF	0.6277	0.7050
NMF	0.7950	1.3710
BNMF	0.6598	0.6987
EMF-1	0.6046	0.6705
EMF-2	0.6114	0.6653
EMF-3	0.6013	0.6778
EMF-4	0.6303	0.6780
EMF-5	0.6109	0.6846
EMF-6	0.6087	0.6808
EMF-7	0.6108	0.6652
EMF-8	0.6075	0.6672
EMF-9	0.6209	0.7050
EMF-10	0.5993	0.6581
EMF (best)	0.5993	0.6581
EMF (worst)	0.6303	0.7050
EMF (avg)	0.6105	0.6752

Regarding the performance of the evolutionary process, Figure 4 shows the evolution of the 25th percentile of fitness values for each run of the experiments. As it can be seen, the fitness values of the population decrease as the evolutionary process goes on, pointing out the quality of the solutions within the population improves. This phenomenon can be observed for both MovieLens and FilmTrust datasets.

Concerning the execution time of our evolutionary approach, the average EMF execution time is 8.069 h, with a standard deviation of 5.029 when using the MovieLens dataset, while the median is 6.483. The average execution time for the Filmtrust dataset is 8.335 h, with a standard deviation of 11.231 and with a median of 4.099 h. Experiments have been run in a computer with an Intel i7 CPU, 16 Gb RAM, and Ubuntu 18.10 as the operating system.

Figure 3. Tree representation of the best individual achieved for the FilmTrust dataset (EMF-10). The correspondence between node symbols and operators are defined in Table 1.

Figure 4. Evolution of the 25th percentile (first quartile) of the population fitness for each run on both (a) MovieLens and (b) FilmTrust datasets.

3.3. Convergence of the Evolutionary Method

As described in Section 2.3, we have computed the moving average distances $D_W(n)$ in order to test the convergence of the GP method. To get a fair balance between locality of the compared best individuals and a global perspective, we have set the window width to $W = 5$. The results of these tests are shown in Figures 5 and 6.

In this way, in Figure 5, we show the evolution of the function $D_5(n)$ for the MovieLens dataset for each of the 10 executions of the EMF method. In this figure, we can observe the existence of wide variations between the best individuals along the first 80–100 generations. These variations correspond to the exploratory phase of the GP search algorithm. During this phase, the maximum of the fitness function has not achieved its stable regime so new individuals, potentially very different from the previous best solution, may obtain better scores, giving rise to the observed large fluctuations.

From the 100th generation on, the variations are tamer, corresponding to the stable phase of the algorithm. In this regime, the algorithm has approached to a local maximum and only very local changes towards the maximum point produce variations. Observe that, in all the cases, from the 130th generation on, the variations are negligible. Hence, from this analysis we conclude that the EMF method has a satisfactory convergence towards the proposed final solution in the MovieLens dataset.

In Figure 6, we show the evolution of the $D_5(n)$ function along the 10 executions on the FilmTrust dataset. In these plots, we also observe that the largest variations are produced during the first steps of the algorithm, corresponding to the exploratory phase. However, in this case, we observe that the fluctuations are less sharp than in the MovieLens dataset. Even more, the stable regime is achieved much sooner than in the MovieLens case, with no significant variations later than the 80th generation in 8 of the 10 executions (and the variations in the 2 left are much milder). Therefore, this analysis provides strong evidences of the convergence of the EMF method for the FilmTrust dataset, with a faster convergence rate than for MovieLens. This suggests that the search problem in the FilmTrust dataset is easier than in the MovieLens dataset so the GP method finds the local minimum sooner, shortening the exploratory phase.

In order to strengthen our conclusions, in Figure 7 we show a representation of the distances between the best individuals of each execution of the EMF method, both for MovieLens and FilmTrust. For that purpose, we depict a weighted undirected complete graph whose vertices are the 10 executions of the EMF method. The edges are weighted with the L^2 distance between the corresponding best individuals of the executions. The plotting has been generated with the Neato Graphviz library [30] in such a way that the planar plots preserve as much as possible the prescribed relative distances between the vertices.

The mean of the distances between the best individuals in the MovieLens experiment is 2.178 while the mean of the distances in the FilmTrust case is 1.284. This means that the distances in the MovieLens case are approximately 70% bigger than in the FilmTrust case. Accordingly, Figure 7 shows that the distances between the best individuals of the executions on the FilmTrust dataset are significantly smaller than the distances between the best individuals on the MovieLens dataset. Moreover, the nodes in the FilmTrust case essentially form a single cluster (except maybe EMF-6 and EMF-10). On the other hand, the nodes in the MovieLens graph are sparser and cannot be grouped into a single cluster.

This supports the observation that the search problem for the FilmTrust dataset is easier than in the MovieLens case. For FilmTrust, the GP algorithm finds an essentially unique local minimum, and all the solutions are distributed around it. However, in the MovieLens dataset, several local minima are found, with no global attraction, corresponding to the different clusters that can be observed. This shows that the dynamics that rule the learning process of the GP method for the MovieLens dataset are much richer than for FilmTrust.

Figure 5. Convergence of the evolutionary method on the MovieLens dataset. The plots show the moving average distance $D_5(n)$ with window width $W = 5$ and $p = 2$ during the 150 generations of the 10 executions.

Figure 6. Convergence of the evolutive method on the FilmTrust dataset. The plots show the moving average distance $D_5(n)$ with window width $W = 5$ and $p = 2$ during the 150 generations of the 10 executions.

Figure 7. Graphical representation of the distances between the best individuals of each execution of EMF for the (a) MovieLens and (b) FilmTrust datasets. The length of the edges is proportional to the L^2 distance between the corresponding aggregate functions. The mean distance in the MovieLens graph is 70% bigger than the mean distance in FilmTrust.

4. Discussion

This paper presents a new approach to improve the quality of predictions provided by MF models applied to CF-based RS. Related work shows that MF models aggregate the learned factors of users and items through a linear combination of them, i.e., the inner product of the vector that represents the latent factors of the users and the vector that represents the latent factors of the items. This aggregation is based on the assumption that the ratings of a CF system must be modeled by means of a lineal factorization model. This restriction is enforced during the model design so it is not learned from the data. To impose apriorisms in the design of machine learning algorithms is a bad practice that should be avoided. This contribution breaks from this apriorism and lets the data to find the optimal mathematical function to aggregate the user and item latent factors without any restriction.

Experimental results show that there exists better approaches to aggregate latent factors spaces than the traditional lineal aggregation. This work must be analyzed as a preliminary work in this field since these conclusions are supported by two datasets with a low number of ratings due to the high computational cost of the proposed method.

One of the drawbacks of using evolutionary computation is, precisely, the high computational cost. As a population-based method that needs to evaluate the fitness of the individuals following a generational scheme, the critical point of the computational complexity of such methods is the evaluation function. In the case of EMF, the evaluation function performs the MF according to the evolved function represented by the individual and then computes the MAE. Due to the high computational cost of this process, which is executed several times on each generation, we had to use rather small datasets for the experimentation.

Although using GP in MF might be seen as counterproductive due to the high computational cost, that problem can be mitigated with a proper setup of the algorithm, that is, reducing the size of the population as well as the number of generations. In fact, according to our experimental results, the most significant improvements in the quality of the solutions happen within the first generations (see Figure 4). Hence, we can deal with the aforementioned issue keeping a low number of generations and population size while obtaining promising results compared to other MF methods.

Even though it is not possible to guarantee that EMF reaches a global optimum, as evolutionary algorithms are approximate optimization methods, we analyzed the best individuals of each EMF execution to study its exploration capabilities.

In this work, the fitness function proposed for the GP-based optimization method has been designed to increase the accuracy of the predictions performed by the CF method. However, there are other measures widely used to evaluate the quality of the output of a RS [31]. On one hand, to evaluate the quality of the recommendations, measures such as precision, recall, and F1 are commonly used. On the other hand, to evaluate the quality of a list of recommendations, nDCG is the most used measure. In future work, we propose to apply multiobjective GP-based optimization in such a way that the accuracy of predictions, the veracity of the recommendations, and the quality of the list of recommendations are boosted.

Another improvement to the evolutionary process of EMF is to include an additional step within the breeding pipeline to reduce the complexity of the generated trees by using rewriting tools. This way, it would be possible to reduce those operations that are redundant (e.g., $* \text{One } X \rightarrow X$, $+ \text{Zero } X \rightarrow X$, $\text{inv inv } X \rightarrow X$, etc.).

Author Contributions: Conceptualization, R.L.-C., and Á.G.-P., and F.O.; formal analysis, Á.G.-P.; methodology, R.L.-C. and F.O.; software, R.L.-C., Á.G.-P., and F.O.; writing—original draft preparation, R.L.-C., Á.G.-P., and J.B.; writing—review and editing, J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by Instituto de Ciencias Matemáticas through a Severo Ochoa Postdoctoral Fellowship SOLAUT_00030167, and Spanish Ministry of Science and Education and Competitiveness (MINECO) and European Regional Development Fund (FEDER) under grants TIN2017-85727-C4-3-P (DeepBio).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

This appendix contains the best individuals generated by the genetic-programming-based algorithms. On one hand, the proposed method has generated the following individuals for the MovieLens dataset:

- EMF-1: $* - \cos \cos \log \exp \operatorname{atan} p_{u4} - \operatorname{atan} - q_{i3} \exp - \operatorname{atan} \log \exp p_{u1} \exp \cos \log \cos \log \exp - \exp \cos p_{u2} \exp - \operatorname{atan} \log \operatorname{atan} \sin \cos \cos \operatorname{atan} \operatorname{atan} \exp - \log \exp \operatorname{atan} \cos \log \exp \operatorname{atan} p_{u4} p_{u2}$
- EMF-2: $- + \exp \sin q_{i3} \exp + \operatorname{atan} p_{u2} \cos \text{One } p_{u1} * + p_{u1} \sin + p_{u1} \sin + + p_{u1} + + \exp q_{i5} \exp + * p_{u1} p_{u0} \sin + + p_{u1} * + * * \text{One One } p_{u0} p_{u1} q_{i2} \text{One} + p_{u1} * + * * \text{One One } q_{i4} p_{u1} q_{i2} \text{One } q_{i2}$
- EMF-3: $- - \operatorname{inv} \operatorname{inv} + \operatorname{inv} \operatorname{inv} + \sin \text{One} \cos \text{Zero} - \operatorname{inv} \operatorname{inv} + \operatorname{inv} \operatorname{inv} + \sin \operatorname{atan} - + + * q_{i0} p_{u4} p_{u0} * q_{i0} p_{u4} \cos \text{Zero} \cos \text{Zero} + * \operatorname{inv} \operatorname{inv} q_{i3} p_{u4} - \sin \operatorname{atan} + + q_{i0} p_{u0} * q_{i0} q_{i0} + * \operatorname{inv} \operatorname{inv} q_{i3} p_{u4} - \sin \operatorname{atan} + + q_{i0} p_{u0} * \cos \text{Zero } p_{u4} \operatorname{atan} - + + q_{i0} p_{u0} * \operatorname{inv} + \operatorname{inv} \operatorname{inv} + \sin \text{One} \cos \text{Zero} \cos \text{Zero } p_{u4}$
- EMF-4: $- + - - q_{i0} + + + \text{One One} * p_{u3} q_{i0} \text{One} + \text{One } p_{u2} * p_{u0} q_{i0}$
- EMF-5: $- - + + \operatorname{atan} q_{i1} + + \operatorname{atan} p_{u5} \exp p_{u3} \operatorname{atan} + - - \exp q_{i2} \exp \operatorname{atan} p_{u4} q_{i5} \exp q_{i1}$
- EMF-6: $\log \exp + \exp \operatorname{atan} \exp \operatorname{atan} - \exp \operatorname{atan} - \exp p_{u0} q_{i5} q_{i5} \exp \operatorname{atan} - p_{u3} q_{i5}$
- EMF-7: $* - - - \cos - - p_{u1} \operatorname{atan} \operatorname{inv} - q_{i3} \exp \cos * \operatorname{atan} - \exp \operatorname{atan} p_{u1} - q_{i4}$
- EMF-8: $- + - q_{i4} p_{u2} \operatorname{inv} * \cos \sin - q_{i4} p_{u2} \operatorname{atan} \exp q_{i2} - \cos - q_{i4} p_{u4} \cos \cos * - - p_{u2} * q_{i4} \text{One } p_{u2} - \operatorname{atan} p_{u0}$
- EMF-9: $+ \exp q_{i0} - \exp \operatorname{atan} p_{u0} + - \text{Zero} - \exp q_{i0} q_{i5} + * p_{u5} q_{i5} - p_{u4} q_{i2}$
- EMF-10: $\exp \operatorname{atan} \operatorname{pow} \operatorname{pow} * \exp \exp p_{u5} \operatorname{inv} \exp q_{i0} \exp p_{u4} \exp \operatorname{inv} \exp p_{u5}$

On the other hand, the proposed method has generated the following individuals for the FilmTrust dataset:

- EMF-1: $\exp \operatorname{atan} + \cos p_{u5} \exp + + p_{u3} q_{i2} p_{u3}$
- EMF-2: $- - + \sin - - p_{u0} + \text{One} \exp \sin \cos \exp + p_{u6} - + \sin p_{u9} + \sin q_{i4} \exp \sin - + \text{One} \exp \sin \cos - + \sin \sin p_{u9} + \text{One} \exp \sin \cos \exp + p_{u6} - + \sin q_{i4} \exp \sin - \sin p_{u9}$

- EMF-3: $\text{inv exp atan} - p_{u2} \text{ exp exp atan} - \text{exp} - \text{exp} - p_{u2} \text{ exp atan} - p_{u2} p_{u7} \text{ exp} - - p_{u7} q_{i3} \text{ exp atan} - p_{u2} \text{ exp} - p_{u2} \text{ exp atan} - p_{u2} \text{ exp exp atan} - \text{exp} - p_{u7} \text{ exp} - - p_{u2} \text{ exp exp atan} - \text{exp} - p_{u2} \text{ exp} - - p_{u7} q_{i3} \text{ exp atan} - p_{u2} \text{ exp} - p_{u2} \text{ exp atan} - p_{u2} \text{ exp exp} - p_{u2} \text{ exp atan} - p_{u2} - p_{u2} p_{u7} - p_{u7} p_{u2} p_{u7} - p_{u7} p_{u2} - p_{u7} p_{u2}$
- EMF-4: $+ p_{u4} \text{ exp cos } q_{i6}$
- EMF-5: $\text{exp atan} - \text{exp exp } p_{u1} - q_{i9}$
- EMF-6: $\text{exp atan} + q_{i4} + + p_{u3} \text{ atan atan } p_{u2} \text{ inv } p_{u2}$
- EMF-7: $+ \text{atan exp } q_{i2} \text{ exp cos } p_{u2}$
- EMF-8: $+ \text{exp } p_{u3} \text{ exp cos sin exp} - \text{atan } q_{i0}$
- EMF-9: $\text{exp inv cos cos exp cos exp atan} - * \text{One exp} * q_{i6} \text{ cos cos exp} * p_{u7} q_{i2} \text{ exp atan} - * \text{One exp} * q_{i6} \text{ cos } p_{u4} p_{u0}$
- EMF-10: $\text{exp atan} + * \text{atan} + + \text{atan} + * \text{atan} + + q_{i5} \text{ atan} + q_{i5} \text{ atan} - \text{inv } p_{u8} * q_{i5} \text{ inv inv } p_{u8} \text{ inv } p_{u8} + q_{i5} \text{ atan} - \text{inv } p_{u8} * q_{i5} \text{ inv inv } p_{u8} \text{ inv } p_{u8} \text{ atan} + q_{i5} \text{ atan} - \text{inv } p_{u8} * q_{i5} \text{ inv} + q_{i5} \text{ atan } q_{i5} \text{ inv } p_{u8} + q_{i5} \text{ atan} - \text{inv } p_{u8} * q_{i5} \text{ inv inv } p_{u8} \text{ inv } p_{u8}$

Note that these individuals have been encoded using prefix notation. $p_{u\alpha}$ and $q_{i\alpha}$ denote the α -th latent factor of the user and item, respectively.

References

1. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl.-Based Syst.* **2013**, *46*, 109–132. [CrossRef]
2. Ricci, F.; Rokach, L.; Shapira, B. Introduction to recommender systems handbook. In *Recommender Systems Handbook*; Springer: Berlin, Germany, 2011; pp. 1–35.
3. Wan, X.; Zhang, B.; Zou, G.; Chang, F. Sparse Data Recommendation by Fusing Continuous Imputation Denoising Autoencoder and Neural Matrix Factorization. *Appl. Sci.* **2019**, *9*, 54. [CrossRef]
4. Melo, E. Improving Collaborative Filtering-Based Image Recommendation through Use of Eye Gaze Tracking. *Information* **2018**, *9*, 262. [CrossRef]
5. Resnick, P.; Varian, H.R. Recommender systems. *Commun. ACM* **1997**, *40*, 56–59. [CrossRef]
6. Ai, Q.; Azizi, V.; Chen, X.; Zhang, Y. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* **2018**, *11*, 137. [CrossRef]
7. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749. [CrossRef]
8. Herlocker, J.L.; Konstan, J.A.; Borchers, A.; Riedl, J. An algorithmic framework for performing collaborative filtering. In *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*; Association for Computing Machinery, Inc.: New York, NY, USA, 1999; pp. 230–237.
9. Schafer, J.B.; Frankowski, D.; Herlocker, J.; Sen, S. Collaborative filtering recommender systems. In *The Adaptive Web*; Springer: Berlin, Germany, 2007; pp. 291–324.
10. Su, X.; Khoshgoftaar, T.M. A survey of collaborative filtering techniques. *Adv. Artif. Intell.* **2009**, 2009. [CrossRef]
11. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
12. Han, H.; Huang, M.; Zhang, Y.; Bhatti, U. An Extended-Tag-Induced Matrix Factorization Technique for Recommender Systems. *Information* **2018**, *9*, 143. [CrossRef]
13. Bottou, L. Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT'2010*; Lechevallier, Y., Saporta, G., Eds.; Physica-Verlag HD: Heidelberg, Germany, 2010; pp. 177–186.
14. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Springer: Berlin, Germany, 2003; Volume 53.
15. Koza, J.R.; Koza, J.R. *Genetic Programming: On the Programming of Computers by Means Of Natural Selection*; MIT Press: Cambridge, MA, USA 1992; Volume 1.
16. González-Prieto, Á.; Lara-Cabrera, R.; Ortega, F. Symbolic Derivation. 2019. Available online: <https://github.com/AngelGonzalezPrieto/sym-derivation> (accessed on 15 January 2020). [CrossRef]

17. Horváth, T.; de Carvalho, A.C.P.L.F. Evolutionary computing in recommender systems: A review of recent research. *Nat. Comput.* **2017**, *16*, 441–462. [CrossRef]
18. Sadeghi, M.; Asghari, S.A. Recommender Systems Based on Evolutionary Computing: A Survey. *J. Softw. Eng. Appl.* **2017**, *10*, 407–421. [CrossRef]
19. Belém, F.M.; Martins, E.F.; Almeida, J.M.; Gonçalves, M.A. Personalized and object-centered tag recommendation methods for Web 2.0 applications. *Inf. Process. Manag.* **2014**, *50*, 524–553. [CrossRef]
20. Anand, D.; Bharadwaj, K.K. Adaptive user similarity measures for recommender systems: A genetic programming approach. In Proceedings of the 2010 3rd International Conference on Computer Science and Information Technology, Chengdu, China, 9–11 July 2010; Volume 8, pp. 121–125.
21. Guimarães, A.; Costa, T.F.; Lacerda, A.; Pappa, G.L.; Ziviani, N. Guard: A genetic unified approach for recommendation. *J. Inf. Data Manag.* **2013**, *4*, 295. Available online: <https://periodicos.ufmg.br/index.php/jidm/article/view/217> (accessed on 15 January 2020).
22. Wilhelmstötter, F. JENETICS: Java Genetic Algorithm Library. 2012 Available online: <http://jenetics.io/> (accessed on 15 January 2020).
23. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst. (TIIS)* **2016**, *5*, 19. [CrossRef]
24. Guo, G.; Zhang, J.; Yorke-Smith, N. A Novel Bayesian Similarity Measure for Recommender Systems. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), Beijing, China, 3–9 August 2013; pp. 2619–2625.
25. Mnih, A.; Salakhutdinov, R.R. Probabilistic matrix factorization. *Adv. Neural Inf. Process. Syst.* **2008**, 1257–1264. Available online: <http://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf> (accessed on 15 January 2020).
26. Lee, D.D.; Seung, H.S. Algorithms for non-negative matrix factorization. *Adv. Neural Inf. Process. Syst.* **2001**, *13*, 556–562.
27. Hernando, A.; Bobadilla, J.; Ortega, F. A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. *Knowl.-Based Syst.* **2016**, *97*, 188–202. [CrossRef]
28. Rudin, W. *Real and Complex Analysis*; Mathematics Series; McGraw-Hill: New York, NY, USA, 1987.
29. Ortega, F.; Zhu, B.; Bobadilla, J.; Hernando, A. CF4J: Collaborative filtering for Java. *Knowl.-Based Syst.* **2018**, *152*, 94–99. [CrossRef]
30. Ellson, J.; Gansner, E.; Koutsofios, L.; North, S.; Woodhull, G.; Description, S.; Technologies, L. *Graphviz—Open Source Graph Drawing Tools*; Lecture Notes in Computer Science; Springer-Verlag: Berlin, Germany, 2001; pp. 483–484.
31. Herlocker, J.L.; Konstan, J.A.; Terveen, L.G.; Riedl, J.T. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst. (TOIS)* **2004**, *22*, 5–53. [CrossRef]