RECOMMENDER SYSTEMS BASED ON MATRIX FACTORIZATION FOR AGRICULTURAL DATASETS

Gábor Farkas, Katalin Lőrincz (Budapest, Hungary)
Péter Magyar (Szombathely, Hungary)
András Molnár (Budapest, Hungary)

Communicated by Imre Kátai
(Received March 27, 2019; accepted July 6, 2019)

Abstract. As hardware devices became increasingly sophisticated and cheaper, and fast and broadband wireless internet connection became available not just in towns but also in remote rural areas, using sensors to collect various kinds of data became common in agriculture and applying these sensors to a wide range of locations using huge databases. However, despite the numerous highly qualified agricultural specialists and this huge amount of data collected, lot of useful information remains hidden. A demand for software naturally arises which is able to handle these enormous databases, and can derive latent information from it to ease decision making in important agricultural situations. We delineate in our paper the effort and outcome while we examined Farm Accountancy Data Network data collected by the Hungarian Research Institute of Agricultural Economics (AKI). Our main interest was to find those questions which can be answered by the use of a matrix factorization (MF) model

1. Introduction

Similar to other sectors, agriculture is rapidly influenced by the digital transformation. The present and certainly the future will be increasingly data intensive, meaning that more and more sources will provide enormous amount

Key words and phrases: Matrix factorization, agriculture.
2010 Mathematics Subject Classification: 68P99.
The research has been supported by the European Union, co-financed by the European Social Fund Research and development activities at the Eötvös Loránd Universitiy's Campus in Szombathely, EFOP-3.6.1-16-2016-00023.

of data to be utilized [9]. The data revolution is expected to provide numerous benefits, including increasing efficiency and better sustainability performance to mention a few [11]. Overall, all of these foreseen advancements can effectively combat the challenges, providing food in a sustainable manner for the rapidly increasing population [10].

Our goal was to build a matrix factorization model working on a set of agricultural data collected by AKI. We built our model for predicting the missing values and recognising the 'incriminating' values, i.e. where the difference is significant between the collected value and the approximation. We concentrated our attention on the Explicit Matrix Factorization Method (EMFM). We speak about explicit collection of data when we get high-quality real information as in our project where data was collected directly from grain growers by questionnaires.

1.1. The Netflix Prize Competition

Matrix factorization models suddenly were the center of computer scientists' attention in 2006 when a contest was announced by Netflix, an on-line DVD-rental and video streaming service company. The goal of the competition was to develop a recommender system which can predict people's preferences for movies. In 2009 the research group BellKor's Pragmatic Chaos gained the 1.000.000 $ winner's prize. Their submitted program with prediction error RMSE (root mean squared error) firstly reached 10% better result than the Netflix Cinematch algorithm. The Netflix company released a training set of 100 480 507 ratings spanning 480 189 anonymous customers and their ratings on 17 770 movies, each movie being rated on a scale of 1 to 5 stars. Participating teams submitted the predicted ratings for a test set of 2 817 131 ratings, and Netflix calculated an RMSE by their own evaluation system.

Between 2006 and 2009 numerous articles were published characterizing various methods to implement effective recommender systems. We can observe that most participants of the Netflix Prize Contents used some variations of the matrix factorization model. This fact convinced us that using an appropriate MF model we could develop a decision preparing system in agricultural environment based on high confidence level predictions. Furthermore, the problem of 'big' and 'sparse' data arising in the agriculture is highly similar to the problems investigated in connection with the recommender systems.

2. The matrix factorization model

Although, in this section we give examples from the Netflix Prize Competition and keep the well-known notations introduced in this area, the meaning of concepts can be more generalized, e.g. the sets of *users* and *items*.

2.1. Collaborative Filtering (CF)

Due to total lack of detailed profiles of the users' preferences the recommender strategies based on content filtering had to be ignored in our project. Contrary to content filtering the approach called *Collaborative Filtering* relies on some events in the users' past. For example, a given Netflix costumer rated a movie 4 stars. There are two well-known methods for implementing collaborative filtering.

2.1.1. Neighbourhood method

One of these is the so-called *neighbourhood method.* An *item-oriented* version evaluates a user's preference for an item based on the connections of a similar item by the same user. So, a neighbour of an item is another item, for example in the Netflix database a neighbour of 'Once Upon a Time in the West' can be an other western movie, or another movie directed by Sergio Leone. On the other hand, the user-oriented approach is trying to find the similarities among different users. We can describe the item-oriented or item based neighbourhood.

Both user and item based variations exist of the *kNN* (k-Nearest Neighbours) method that one of the most popular and simplest manifestation of the above mentioned method. KNN is a so called 'lazy learner' algorithm which means that it does not built a model, and the whole training data set is stored and all computation is delayed until classification. As a matter of fact, the k-nearest neighbours is a relation defined by a distance function that makes the classification or regression.

2.1.2. Latent Factor Model

An alternative realization of the collective filtering is the *Latent Factor Model* (LFM) that tries to predict the appropriate values by characterizing both items and users on some factors inferred from the collected data. The number of factors gives the dimension of the latent space in which the model sets the users and items up. Let us consider an example from the world of movies. Two factors (properties) determine a Cartesian coordinate plane. The axes correspond to two factors, namely the dimensions characterized as 'children's versus grown up' and 'drama versus comedy' movies. The closer a given user and item are situated in the system the closer the connection between them with respect the investigated properties. For example, Figure 1 shows that we expect the C to like the movies *Hannibal, English Patient* and *Schindler' list*, to hate *Toy Story* and *Gru.* Furthermore, on these two dimensions users A, B and movies *Batman, Doctor Who* and *Her* can be considered highly neutral.

One of the most effective methods for realizing a Latent Factor Model is matrix factorization.

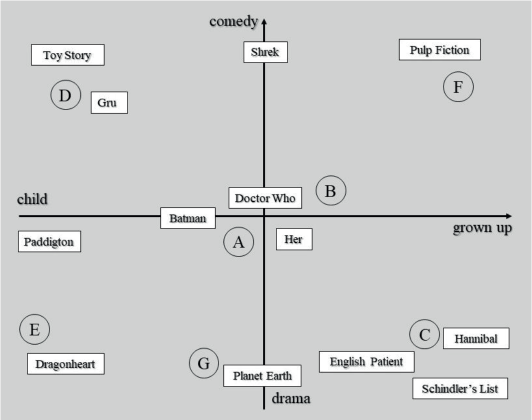


Figure 1. We can see a two dimensional virtual space including users (circled capital letters) and movies (white background). Axis x shows the values on a scale 'children's versus grown up' and axis y 'drama versus comedy'.

2.2. The supervised learning process

Let us consider two arbitrary sets: *users* and *items* with cardinality n and m respectively. Further, let the *set of original data* denoted by R, is a table that columns are identified by the items and row by the users. Some cells of R contains a positive number, the other cells are empty. These numbers characterize a connection, interaction or any other abstract phenomenon between a user and an item. Let us denote the set of the numbers found in the non-empty cells by $\mathscr{R}$ and the value in the intersection of the u-th row and i-th column by r_{ui}.

We randomly divide $\mathscr{R}$ into two disjoint subsets $\mathscr{T}$ and $\mathscr{P}$, namely the *training set* and *probe set* (the latter is also known as *holdout set* or *test set*), where $\mathscr{T} \cup \mathscr{P} = \mathscr{R}$ and $\mathscr{T} \cap \mathscr{P} = \emptyset$. The learning algorithm of the MF model learns on $\mathscr{T}$, the quality of learning is validated on $\mathscr{P}$ and finally we can except that the well-trained system gives sharp approximations on the original data set R.

In our matrix factorization model we correspond a vector $w_u \in \mathbb{R}^k$ to user u and a vector $h_i \in \mathbb{R}^k$ to the item i, where k is the number of factors. The resulting dot product $w_u h_i^T$ determines the 'closeness' between user u and item i in the k dimensional virtual space, e.g. customer u's approximated rating of item i. Let us introduce the following notation:

$$r_{ui} \approx \hat{r}_{ui} = w_u \cdot h_i^T = \sum_{j=1}^{k} w_{u_j} \cdot h_{i_j}^T. \tag{2.1}$$

We recall that $r_{ui} \in R$ and $r_{ui} \in \mathscr{R}$ if the value of r_{ui} is known. If the matrix $W_{n\times k}$ contains all user vectors and $H_{k\times m}$ contains all item vectors then we can estimate the missing elements of R by the product $W \cdot H^T$.

2.2.1. The loss function

To calculate the value of the elements of user and item vectors is a difficult problem. In a general case we initialize the user and item vectors in a random way. The MF model then learns the factor vectors minimizing the error of the approximation (2.1). As a matter of fact we minimize the square of the difference between all known values in our dataset R, i.e. the elements of $\mathscr{R}$ and our predictions. We introduce a so-called *loss function* $err : \mathscr{R} \times \mathbb{R} \to \mathbb{R}$:

$$err(r, \hat{r}) = \sum_{r_{ui}\in\mathscr{R}} e_{ui}^2 = \sum_{r_{ui}\in\mathscr{R}} (r_{ui} - \hat{r}_{ui})^2 = \sum_{r_{ui}\in\mathscr{R}} \left(r_{ui} - w_u \cdot h_i^T\right)^2 . \tag{2.2}$$

In reality two phenomena can slow down the speed of the learning process. When a recommender system is not capable of modelling the addition of new users without retraining the whole model, we speak about *cold start problem*. To relieve this problem we can add new information sources or use implicit data collection. In our experiment both of these were impossible, so we had to develop a model which is resistant to the cold start problem ab ovo.

The other malfunction we mention is the *overfitting*, where the system learns only the unique properties of the elements of $\mathscr{T}$ and does not recognize the real connections and rules on the whole original data set R as illustrated in Figure 2.

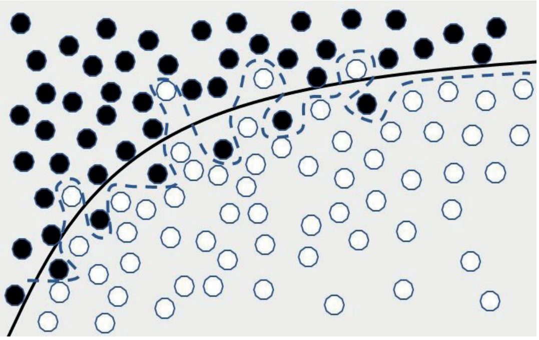


Figure 2. In this pattern the dashed line represents an overfitted model and the black line represents the real trends.

Thus, overfitting negatively impacts on the predictions given by the model on the original data set. To avoid the overfitting we limit the magnitude of the learned parameters i.e. the elements of W and H. This process is called *regularization*, which modifies our loss function adding a so-called *penalty term*:

$$f(r, \hat{r}) = \sum_{r_{ui} \in \mathscr{R}} \left(r_{ui} - w_u \cdot h_i^T\right)^2 + \lambda \left(\|w_u\|^2 + \|h_i\|^2\right), \tag{2.3}$$

where $\| \cdot \|^2$ means the Euclidean norm of a given vector.

2.2.2. The learning method: SGD

The *Stochastic Gradient Descent* (SGD) method is one of the numerous variations of the popular technique *Gradient Descent* (GD) for finding minimum of functions. S. Funk's described in [7] the application of SGD for the loss function given in the following form:

$$f(\Theta) = \sum_j f_j(\Theta),$$

where our case Θ means the set of all user and item vectors, i.g. $\Theta = W \cup H$. The main steps of SGD are shown in Algorithm 1.

Algorithm 1 BASICSGD

```
procedure BASICSGD($\varepsilon, \alpha, f_k$)                ▷ Basic idea of SGD
    random initialization of $\Theta$
    for number of iterations do
        for repetition for all $k$ do
            $\triangle := \frac{\delta f_k}{\delta \Theta}$
            $\Theta \leftarrow \Theta - \alpha \triangle$
        end for
    end for
end procedure
```

$\alpha > 0$ is called *learning rate* and its value depends on the magnitude of the error. The number of iterations z can be an input parameter or we use a *stopping criteria* ε, for example $|\alpha \triangle| < \varepsilon$. In the latter case ε is an input parameter instead of z.

For approximating the minimum of the loss function (2.3) we need the partial differentiation of f_{ui} with respect to any w_u and h_i. Let us introduce the following notation:

$$f_{ui}(w_u, h_i) := \left(r_{ui} - w_u \cdot h_i^T\right)^2.$$

Then we get that

$$\frac{\delta f_{ui}}{\delta w_u} = \frac{\delta}{\delta w_u}\left(\left(r_{ui} - w_u \cdot h_i^T\right)^2 + \lambda\left(\|w_u\|^2 + \|h_i\|^2\right)\right) =$$

$$= -2h_i\left(r_{ui} - w_u \cdot h_i^T\right) - 2\lambda w_u = -2\left(e_{ui}h_i - \lambda w_u\right),$$

and similarly

$$\frac{\delta f_{ui}}{\delta h_i} = -2\left(e_{ui}w_u - \lambda h_i\right).$$

Finally, considering the multiplicative factor 2 as a part of the constant α we obtain the pseudocode Algorithm 2.

Algorithm 2 SGD for f_{ui}

```
procedure SGD($\varepsilon, \alpha, \lambda$)                ▷ SGD for $f_{ui}$
    random initialization of the vectors $w_u$ and $h_i$
    for number of iterations do
        for repetition if $r_{ui}$ is known do
            $\triangle_u := \frac{\delta f_{ui}}{\delta w_u}$, $\triangle_i := \frac{\delta f_{ui}}{\delta h_i}$
            $w_u \leftarrow w_u - \alpha\triangle_u = w_u + \alpha\left(e_{ui}h_i - \lambda w_u\right)$
            $h_i \leftarrow h_i - \alpha\triangle_i = h_i + \alpha\left(e_{ui}w_u - \lambda h_i\right)$
        end for
    end for
end procedure
```

2.2.3. Biased Matrix Factorization (BMF)

We can observe a phenomenon called *bias* in collaborative filtering, when some users or items systematically deform the values of user-item interactions. For example, a rigorous film critic notoriously gives lower ratings then others or a bad movie gets higher ratings because of the brilliant casting or a famous director. The former is called *user bias*, the latter *item bias*. The bias can be built in r_{ui} in the following way:

$$b_{ui} = \mu + b_i + b_u,$$

where μ means the overall average rating, b_i is the item bias and b_u is the user bias. Now, modifying (2.1) we get the extended version of the estimation:

$$\hat{r}_{ui} = \mu + b_i + b_u + w_u h_i^T.$$

Finally, we get the loss function in the following form:

$$f(r, \hat{r}) = \sum_{r_{ui} \in \mathscr{R}} \left(r_{ui} - \mu - b_u - b_i - w_u \cdot h_i^T\right)^2 + \lambda\left(\|w_u\|^2 + \|h_i\|^2 + b_u^2 + b_i^2\right).$$

Although the loss function is different in the MF and the BMF method, the learning process remains the same.

2.2.4. SVD Plus Plus

Singular Value Decomposition (SVD) is a well-known method for factorizing a matrix. Although it does not work properly on sparse matrices, using additional data sources, e.g. implicit data collection, highly effective SVD based algorithm can be developed for the lower rank decomposition of the original matrix R. A good example is the *SVD Plus Plus* latent factor method that is derived from an SVD-type decomposition model, as we can see in [3]. Contrary to the matrix factorization models described above, SVD Plus Plus represents the users through the items with which the interaction value is high, instead of providing explicit parametrization. Of course, the loss function $f(r, \hat{r})$ is modified because of the possibility of implicit data.

2.2.5. Hyperparameters

The fine-tuning of most parameters is taking place during the learning process but some parameters can be set before the learning begins. These are called *hyperparameters* and these are input parameters of the learning algorithm. For setting the appropriate values of the hyperparameters we used in our experiment the well-known *cross-validation* method. The following hyperparameters were set by minimizing the RMSE on the test set: k is the number of factors, it is the number of iterations, λ is the *regularization coefficient* and α is the learning rate.

$\mathscr{R}$ is divided into 5 disjoint subset with even cardinality each $\mathscr{R} = \mathscr{P}_1 \cup \cdots \cup \mathscr{P}_5$ and every set $\mathscr{P}_i$ is also divided into 5 disjoint subsets with even cardinality each $\mathscr{P}_i = \mathscr{P}_{i_1} \cup \cdots \cup \mathscr{P}_{i_5}$. The learning algorithm ran in 5×5 rounds, where $\mathscr{P}_{i_j}$ was the test set and the training set was

$$\bigcup_{l \in [1,5] \setminus \{j\}} \mathscr{P}_{i_l} \quad i, j \in [1, 5] \,.$$

We can say that we completed a 2-level 5-fold cross-validation. In total, 360 000 different combinations of hyperparameter values were compared, and the best scoring k, it, λ and α values were chosen for every $\mathscr{P}_{i_j}$. Finally, the average of the best values on the sets $\mathscr{P}_1, \ldots, \mathscr{P}_5$ was calculated.

3. Results

The data set supported by AKI contained $|\mathscr{R}| = 8\ 179\ 590$ known values. The set of users consisted of sown areas identified by a number. The items are quantitative information with respect to different crops, for example, ’Insecticides costs for wheat’, ’Machinery costs for maize’, ’Fertilizer costs for barley’, ’Sown area for rapeseed’, ’Nitrogen fertilizer costs for wheat’, and so on.

Our computational results prove that a well-constructed MF model effectively learns on the data supported by AKI. This fact makes us convinced that our model is capable of approximating sharply the missing values of the original set even if the rate of unknown data exceeds 95%.

Further, our model is able to detect ’suspicious’ values among the collected data, i.e. a predicted value significantly differs – according to a predefined precision criteria – from the value collected on questionnaire.

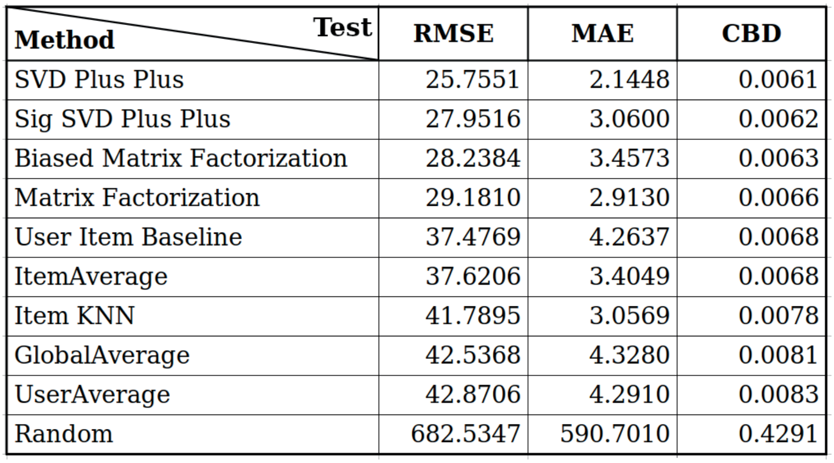
The programs were developed in Python and C languages, some tests were performed with the help of the MyMediaLite free recommender system library [12].

3.1. The investigated methods

Figure 3 shows a comparison of the performances of different recommender methods. The first 4 models can be considered as a variations of the matrix factorization model. The error was measured by three different techniques: RMSE (Root Mean Square Error), MAE (Mean Absolute Error) and CBD (Capped Binomial Deviation).

In each program the SGD (Stochastic Gradient Descent) was implemented as a learning algorithm (Algorithm 2). The efficiency of learning can be seen in Figure 4.

SVD Plus Plus is described in 2.2.4.
Sig SVD Plus Plus is a variation of SVD Plus Plus that uses a sigmoid function.
Biased Matrix Factorization is described in 2.2.3.
Matrix Factorization is the basic MF model described in 2.
User Item Baseline uses the average interaction value, and a regularized user and item bias for prediction.
Item Average uses the average interaction value of an item for prediction.
Item KNN Item-based kNN is described in 2.1.1.
Global Average uses the average interaction value over all interactions for prediction.
User Average uses the average interaction value of a user for prediction.
Random uses a random interaction value for prediction.



Test Method	RMSE	MAE	CBD
SVD Plus Plus	25.7551	2.1448	0.0061
Sig SVD Plus Plus	27.9516	3.0600	0.0062
Biased Matrix Factorization	28.2384	3.4573	0.0063
Matrix Factorization	29.1810	2.9130	0.0066
User Item Baseline	37.4769	4.2637	0.0068
ItemAverage	37.6206	3.4049	0.0068
Item KNN	41.7895	3.0569	0.0078
GlobalAverage	42.5368	4.3280	0.0081
UserAverage	42.8706	4.2910	0.0083
Random	682.5347	590.7010	0.4291

Figure 3. Comparison between the performances of recommender methods on AKI's dataset using three different types of error.

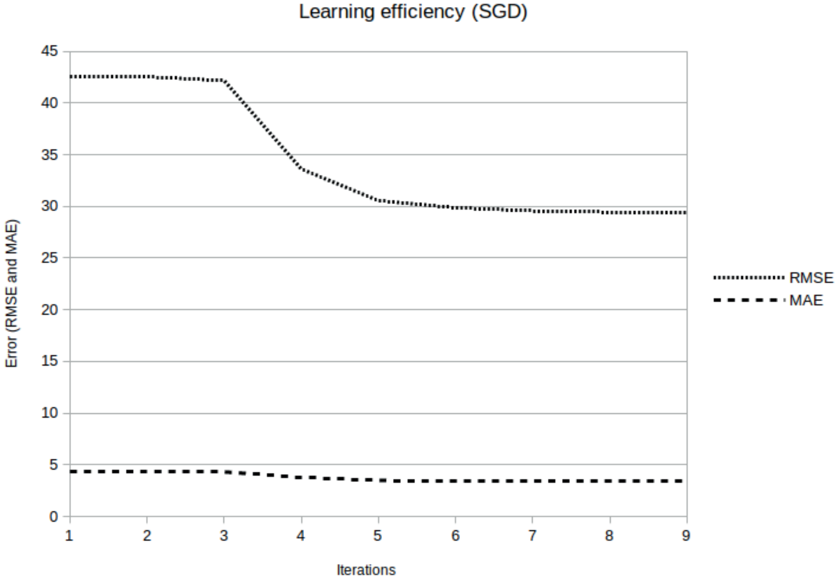


Figure 4. The efficiency of the stochastic gradient desscent (SGD) method on AKI's data set.

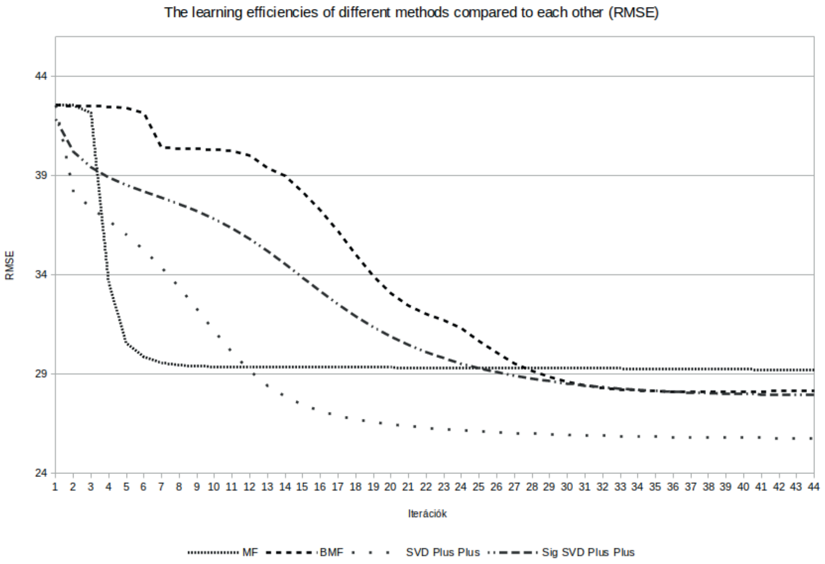


Figure 5. The comparison of different matrix factorization strategies on AKI's data set. The error was measured by RMSE.

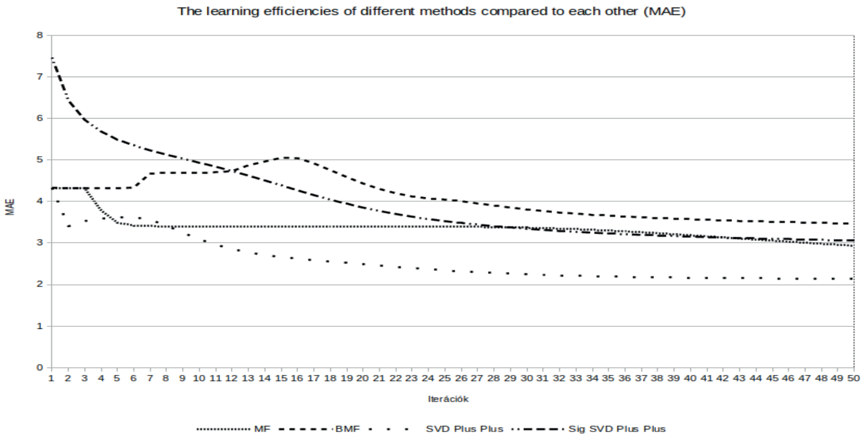


Figure 6. The comparison of different matrix factorization strategies on AKI's data set. The error was measured by MAE.

3.2. Conclusion

Although the SVD Plus Plus and Sig SVD Plus Plus methods give the best approximation on the collected agricultural data set, due to the total lack of additional information sources and possibility of implicit data collection, they are not resistant to the cold start problem. The 'fastest learner' is the basic MF model, and its running time is also the best. The BMF is slower than MF, but gives slightly better approximations.

Thus, our experiment confirms that the explicit matrix factorization model can be a highly effective tool to develop recommender or decision support systems for predicting the missing values and recognising the 'incriminating' values in a data set collected from an agricultural environment.

References

[1] **Paterek, A.,** Improving regularized singular value decomposition for collaborative filtering, *Proceedings of KDD Cup and Workshop*, San Jose, California, USA (2007).

[2] **Koren, Y., R. Bell and C. Volinsky,** Matrix factorization techniques for recommender systems, *Computer*, **42(8)** (2009), 30–37.

[3] **Koren, Y., R. Bell and C. Volinsky,** Factorization meets the neighborhood: a multifaceted collaborative filtering model, *KDD '08, Las Vegas, Nevada, USA*, (2008).

[4] **Goldberg, D. et al.,** Using collaborative filtering to weave an information tapestry, *Comm. ACM*, **35** (1992), 61–70.

[5] **Moreira, J., A. de Carvalho, A. and T. Horvath,** *A General Introduction to Data Analytics*, Wiley, LCCN 2017060728, 2018.

[6] **Takacs, G., I. Pilaszy, B. Nemeth and D. Tikk,** Major components of the gravity recommendation system, *ACM SIGKDD Explorations Newsletter*, **9(2)** (2007), 80–83.

[7] **Funk, S.,** Netflix Update: Try This at Home, `https://sifter.org/simon/journal/`, (2006).

[8] **Bodon, F. and K. Buza,** Adatbányászat, `https://www.tankonyvtar.hu`, (2014).

[9] **Bonneau, V., B. Copigneaux, L. Probst and B. Pedersen,** Industry 4.0 in agriculture: Focus on IoT aspects, `https://ec.europa.eu/growth/tools-databases`, (2017).

[10] **Bögel, Gy.,** Digitális transzformáió a mezőgazdaságban, *Magyar Tudomány*, **179(5)** (2018), 693–701.

[11] **OECD** Digital opportunities for trade in agriculture and food sectors, http://www.oecd.org/officialdocuments, (2019).

[12] **Gantner, Z., S. Rendle, C. Freudenthaler and L. Schmidt-Thieme,** MyMediaLite: A Free Recommender System Library, *5th ACM International Conference on Recommender Systems (RecSys 2011), Chicago, USA*, (2011).

G. Farkas
Department of Computer Algebra
Eötvös Loránd University
Budapest
Hungary
farkasg@inf.elte.hu

K. Lőrincz
Research Institute of
Agricultural Economics,
Budapest
Hungary
lorincz.katalin@aki.naik.hu

P. Magyar
Eötvös Loránd University
Szombathely
Hungary
map115599@gmail.com

A. Molnár
Research Institute of
Agricultural Economics,
Budapest
Hungary
molnar.andras@aki.naik.hu