

Submission

On the Difficulty of Evaluating Baselines A Study on Recommender Systems

Marc Feger B.Sc.

Contents

1	Introduction	2
2	A Study on Recommender Systems	2
2.1	Recommender Problem	2
2.2	Content-Based	3
2.3	Collaborative-Filtering	3
2.4	Matrix-Factorization	3
2.4.1	Basic Matrix-Factorization	4
2.4.2	Regulated Matrix-Factorization	4
2.4.3	Weighted Regulated Matrix-Factorization	4
2.4.4	Biased Matrix-Factorization	4
2.4.5	Advanced Matrix-Factorization	4
2.5	Optimization and Learning	5
2.5.1	Stochastic Gradient Descent	5
2.5.2	Alternating Least Square	5
2.5.3	Bayesian Learning	6

1 Introduction

Today's use of *recommender systems* finds an increased and yet unconscious access to our everyday life. More and more areas of life are therefore subject to constant optimisation. Companies such as *Netflix*, *Amazon* and *YouTube* adapt their product proposals to the individual wishes of their customers. To make this possible, the various *collaborative-filtering* and *content-based recommender systems* are used.

Since [Karlgrén \(1990\)](#) first presented *recommender systems* as a kind of intelligent bookcase, much effort has been put into the development and research of such systems. The most diverse subject areas were not only illuminated by the industry. A whole new branch of research also opened up for science.

In their work “*On the Difficulty of Evaluating Baselines A Study on Recommender Systems*” [Rendle et al. \(2019\)](#) show that current research on the *MovieLens10M* dataset leads in a wrong direction. In addition to general problems, they particularly list wrong working methods and misunderstood *baselines* by breaking them by a number of simple methods such as *matrix-factorization*.

They were able to beat the existing baselines by not taking them for granted. On the contrary, they questioned them and transferred well evaluated and understood properties of the baselines from the *Netflix-Challenge* to them.

As a result, they were not only able to beat the *baselines* reported for the *MovieLens10M*, but also the newer methods from the last 5 years of research. Therefore, it can be assumed that the current and former results obtained on the *MovieLens10M* dataset were not sufficient to be considered as a true baseline. Thus they show the community a critical error on which can be found not only in the evaluation of *recommender systems* but also in other scientific areas.

As a first problem, the authors point out that scientific papers whose focus is on better understanding and improving existing *baselines* do not receive recognition because they do not seem innovative enough. In contrast to industry, which tenders horrendous prizes for researching and improving such *baselines*, there is a lack of such motivation in the scientific field. From the authors point of view, the scientific work on the *MovieLens10M* dataset is misdirected, because one-off evaluations leading to one-hit-wonders, which are then used as a starting point for further work. Thus [Rendle et al. \(2019\)](#) points out as a second point of criticism that the need for further basic research for the *MovieLens10M* dataset is not yet exhausted.

This submission takes a critical look at the topic presented by [Rendle et al. \(2019\)](#). In addition, basic terms and the results obtained are presented in a way that is comprehensible to the non-experienced reader. For this purpose, the submission is divided into three subject areas. First of all, the non-experienced reader is introduced to the topic of recommender systems in the section “*A Study on Recommender Systems*”. Subsequently, building on the first section, the work in the section “*On the Difficulty of Evaluating Baselines*” is presented in detail. The results are then evaluated in a critical discourse.

2 A Study on Recommender Systems

This section explains the basics of *recommender systems* necessary for the essential understanding of the paper presented. Besides the general definition of the *recommender problem*, the corresponding solution approaches are presented. Furthermore, the main focus will be on the solution approach of *matrix factorization*.

2.1 Recommender Problem

The *recommender problem* consists of the entries of the sets \mathcal{U} and \mathcal{I} , where \mathcal{U} represents the set of all *users* and \mathcal{I} the set of all *items*. Each of the *users* in \mathcal{U} gives *ratings* from a set \mathcal{S} of possible scores for the available *items* in \mathcal{I} . The resulting *rating-matrix* \mathcal{R} is composed of $\mathcal{R} = \mathcal{U} \times \mathcal{I}$. The entries in \mathcal{R} indicate the *rating* from *user* $u \in \mathcal{U}$ to *item* $i \in \mathcal{I}$. This entry is then referred to as r_{ui} . Due to incomplete *item-ratings*, \mathcal{R} may also be incomplete. In the following, the subset of all *users* who have rated a particular *item* i is referred to as \mathcal{U}_i . Similarly, \mathcal{I}_u refers to the subset of *items* that were rated by *user* u . Since \mathcal{R} is not completely filled, there are missing values for some *user-item relations*. The aim of the *recommender system* is to estimate the missing *ratings* \hat{r}_{ui} using a *prediction-function* $p(u, i)$. The *prediction-function* consists of $p : \mathcal{U} \times \mathcal{I} \rightarrow \mathcal{S}$ ([Desrosiers and Karypis, 2011](#)). In the further course of the work different methods are presented to determine $p(u, i)$.

In the following, the two main approaches of *collaborative-filtering* and *content-based recommender systems* will be discussed. In addition, it is explained how *matrix factorization* can be integrated into the two ways of thinking.

2.2 Content-Based

Content-based recommender systems work directly with *feature vectors*. Such a *feature vector* can, for example, represent a *user profile*. In this case, this *profile* contains information about the *user's preferences*, such as *genres*, *authors*, etc. This is done by trying to create a *model* of the *user*, which best represents his preferences. The different *learning algorithms* from the field of *machine learning* are used to learn or create the *models*. The most prominent *algorithms* are: *tf-idf*, *bayesian learning*, *Rocchio's algorithm* and *neural networks* (Lops et al., 2011; Dacrema et al., 2019; Desrosiers and Karypis, 2011). Altogether the built and learned *feature vectors* are compared with each other. Based on their closeness, similar *features* can be used to generate *missing ratings*. Figure 1a shows a sketch of the general operation of *content-based recommenders*.

2.3 Collaborative-Filtering

Unlike the *content-based recommender*, the *collaborative-filtering recommender* not only considers individual *users* and *feature vectors*, but rather a *like-minded neighborhood* of each *user*. Missing *user ratings* can be extracted by this *neighbourhood* and *networked* to form a whole. It is assumed that a *missing rating* of the considered *user* for an unknown *item* i will be similar to the *rating* of a *user* v as soon as u and v have rated some *items* similarly. The similarity of the *users* is determined by the *community ratings*. This type of *recommender system* is also known by the term *neighborhood-based recommender* (Desrosiers and Karypis, 2011). The main focus of *neighbourhood-based methods* is on the application of iterative methods such as *k-nearest-neighbours* or *k-means*. A *neighborhood-based recommender* can be viewed from two angles: The first and best known problem is the so-called *user-based prediction*. Here, the *missing ratings* of a considered *user* u are to be determined from his *neighborhood* $\mathcal{N}_i(u)$. $\mathcal{N}_i(u)$ denotes the subset of the *neighborhood* of all *users* who have a similar manner of evaluation to u via the *item* i . The second problem is that of *item-based prediction*. Analogously, the similarity of the *items* is determined by their received ratings. This kind of problem considers the *neighborhood* $\mathcal{N}_u(i)$ of all *items* i which were similar rated via the *user* u . The similarity between the objects of a *neighborhood* is determined by *distance functions* such as *mean-squared-difference*, *pearson-correlation* or *cosine-similarity*. Figure 1b shows a sketch of the general operation of the *collaborative-filtering recommender*.

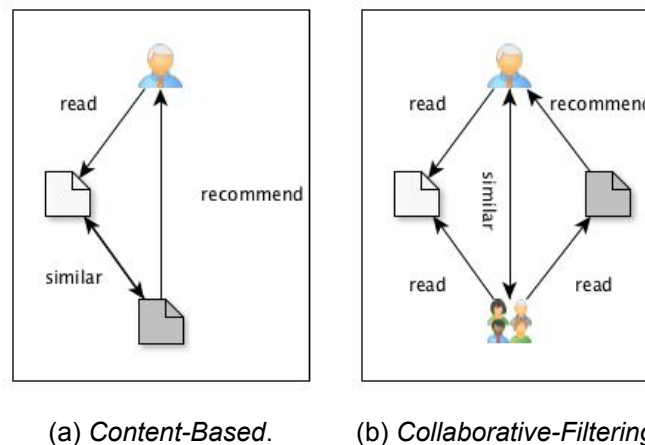


Figure 1: Overview of *content-based* (left) and *collaborative-filtering* (right) recommender systems. *Content-based recommender systems* work via *feature vectors*. In contrast, *collaborative filtering recommender systems* work over *neighborhoods*.

2.4 Matrix-Factorization

The core idea of *matrix factorization* is to supplement the not completely filled out *rating-matrix* \mathcal{R} . For this purpose the *users* and *items* are to be mapped to a joined *latent feature space* with *dimensionality* f . The *user* is represented by the vector $p_u \in \mathbb{R}^f$ and the *item* by the vector $q_i \in \mathbb{R}^f$. As a result, the *missing ratings* and thus the *user-item interaction* are to be determined via the *inner product* $\hat{r}_{ui} = q_i^T p_u$ of the corresponding vectors (Koren et al., 2009). In the following, the four most classical matrix factorization

approaches are described in detail. Afterwards, the concrete learning methods with which the vectors are learned are presented. In addition, the *training data* for which a *concrete rating* is available should be referred to as $\mathcal{B} = \{(u, i) | r_{ui} \in \mathcal{R}\}$.

2.4.1 Basic Matrix-Factorization

The first and easiest way to solve *matrix-factorization* is to connect the *feature vectors* of the *users* and the *items* using the *inner product*. The result is the *user-item interaction*. In addition, the *error* should be as small as possible. Therefore, $\min_{p_u, q_i} \sum_{(u, i) \in \mathcal{B}} (r_{ui} - \hat{r}_{ui})^2$ is defined as an associated *minimization problem*.

2.4.2 Regulated Matrix-Factorization

This problem extends the *basic matrix-factorization* by a *regulation factor* λ in the corresponding *minimization problem*. Since \mathcal{R} is thinly occupied, the effect of *overfitting* may occur due to learning from the few known values. The problem with *overfitting* is that the generated *ratings* are too tight. To counteract this, the magnitudes of the previous vectors is taken into account. High magnitudes are punished by a factor $\lambda(\|q_i\|^2 + \|p_u\|^2)$ in the *minimization problem*. Overall, the *minimization problem* $\min_{p_u, q_i} \sum_{(u, i) \in \mathcal{B}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$ is to be solved. The idea is that especially large entries in q_i or p_u cause $\|q_i\|$, $\|p_u\|$ to become larger. Accordingly, $\|q_i\|$, $\|p_u\|$ increases the larger its entries become. This value is then additionally punished by squaring it. Small values are rewarded and large values are penalized. Additionally the influence of this value can be regulated by λ .

2.4.3 Weighted Regulated Matrix-Factorization

A *regulation factor* λ is introduced in analogy to *regulated matrix-factorization*. Additional *weights* α and β are introduced to take into account the individual magnitude of a vector. The *minimization problem* then corresponds to $\min_{p_u, q_i} \sum_{(u, i) \in \mathcal{B}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(\alpha\|q_i\|^2 + \beta\|p_u\|^2)$.

2.4.4 Biased Matrix-Factorization

A major advantage of *matrix-factorization* is the ability to model simple relationships according to the application. Thus, an excellent data source cannot always be assumed. Due to the *natural interaction* of the *users* with the *items*, *preferences* arise. Such *preferences* lead to *behaviour patterns* which manifest themselves in the form of a *bias* in the data. In principle, a *bias* is not bad, but it must be taken into account when modeling the *recommender system*. The most popular model that takes *bias* into account is called *biased matrix-factorization*. In addition, the *missing rating* is no longer determined only by the *inner product* of the two vectors q_i and p_u . Rather, the *bias* is also considered. Accordingly, a *missing rating* is calculated by $\hat{r}_{ui} = b_{ui} + q_i^T p_u$, where b_{ui} is the *bias* of a *user* u and an *item* i . The *bias* is determined by $b_{ui} = \mu + b_u + b_i$. The parameter μ is the *global average* of all *ratings* $r_{ui} \in \mathcal{R}$. Furthermore, $b_u = \mu_u - \mu$ and $b_i = \mu_i - \mu$. Here μ_u denotes the *average* of all *assigned ratings* of the *user* u . Similarly, μ_i denotes the *average* of all *received ratings* of an *item* i . Thus b_u indicates the *deviation* of the *average assigned rating* of a *user* from the *global average*. Similarly, b_i indicates the *deviation* of the *average rating* of an *item* from the *global average*. In addition, the *minimization problem* can be extended by the *bias*. Accordingly, the *minimization problem* is then $\min_{p_u, q_i} \sum_{(u, i) \in \mathcal{B}} (r_{ui} - \hat{r}_{ui})^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2)$. Analogous to the *regulated matrix-factorization*, the values b_u and b_i are penalized in addition to $\|q_i\|$, $\|p_u\|$. In this case b_u , b_i are penalized more if they assume a large value and thus deviate strongly from the *global average*.

2.4.5 Advanced Matrix-Factorization

This section is intended to show that there are *other approaches* to *matrix-factorization*. Thus, *implicit data* can also be included. First of all, it should be mentioned that *temporary dynamics* can also be included. On the one hand, it is not realistic that a *user* cannot change his taste. On the other hand, the properties of an *item* remain constant. Therefore, *missing ratings* can also be determined *time-based*. A *missing rating* is then determined by $\hat{r}_{ui} = \mu + b_i(t) + b_u(t) + q_i^T p_u(t)$ (Koren et al., 2009). As a second possibility, *implicit influence* can be included. This can involve the *properties* of the *items* a *user* is dealing with. A *missing rating* can be determined by $\hat{r}_{ui} = \mu + b_i + b_u + q_i^T (p_u + |\mathcal{I}_u|^{-\frac{1}{2}} \sum_{i \in \mathcal{I}_u} y_i)$. $y_i \in \mathbb{R}^f$ describes the *feature vectors* of the *items* $i \in \mathcal{I}_u$ which have been evaluated by *user* u . The corresponding *minimization problems* can be adjusted as mentioned in the sections above (Koren, 2008).

2.5 Optimization and Learning

An important point that does not emerge from the above points is the question of how the individual components p_u, q_i, b_u, b_i are constructed. In the following, the three most common methods are presented.

2.5.1 Stochastic Gradient Descent

The best known and most common method when it comes to *machine learning* is *stochastic gradient descent* (SGD). The goal of SGD is to *minimize* the *error* of a given *objective function*. Thus the estimators mentioned in section 2.4 can be used as *objective functions*. In the field of *recommender systems*, Funk (2006) presented a *modified* variant of SGD in the context of the *Netflix Challenge*. SGD can be applied to *regulated matrix-factorization* with *bias* as well as without *bias*. This method can be described by the following pseudo code:

Algorithm 1 SGD of Funk

Require: training-matrix \mathcal{R}_{train} , initial mean μ , initial standard deviation σ^2 , regularization parameter λ , learning rate γ , feature embedding f , epochs to train n_{epochs}

```

1:  $\mathcal{P} \leftarrow \mathcal{N}(\mu, \sigma^2)^{|\mathcal{U}| \times f}$ 
2:  $\mathcal{Q} \leftarrow \mathcal{N}(\mu, \sigma^2)^{f \times |\mathcal{I}|}$ 
3: for  $epoch \in \{0, \dots, n_{epochs} - 1\}$  do
4:   for  $(u, i) \in \mathcal{R}_{train}$  do
5:      $e_{ui} \leftarrow r_{ui} - \hat{r}_{ui}$ 
6:      $q_i \leftarrow q_i + \gamma(e_{ui}p_u - \lambda q_i)$ 
7:      $p_u \leftarrow p_u + \gamma(e_{ui}q_i - \lambda p_u)$ 
8:      $b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i)$ 
9:      $b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u)$ 
10:  end for
11: end for
12: return  $\mathcal{P}, \mathcal{Q}$ 

```

At the beginning, the matrices \mathcal{P}, \mathcal{Q} are filled with *random numbers*. According to Funk (2006) this can be done by a *gaussian-distribution*. Then, for each element in the *training set*, the entries of the corresponding vectors $p_u \in \mathcal{P}, q_i \in \mathcal{Q}$ are recalculated on the basis of the *error* that occurred in an *epoch*. The parameters μ, γ are introduced to avoid *over-* and *underfitting*. These can be determined using *grid-search* and *k-fold cross-validation*. For the *optimization* of the parameters μ and γ the so-called *grid-search* procedure is used. A *grid* of possible parameters is defined before the analysis. This *grid* consists of the sets Λ and Γ . The *grid-search* method then trains the algorithm to be considered with each possible pair of $(\lambda \in \Lambda, \gamma \in \Gamma)$. The models trained in this way are then tested using a *k-fold cross-validation*. The data set is divided into k -equally large fragments. Each of the k parts is used once as a test set while the remaining $(k - 1)$ parts are used as training data. The average error is then determined via the *k-folds* and entered into the *grid*. Thus the pair $(\lambda \in \Lambda, \gamma \in \Gamma)$ can be determined for which the *error* is lowest. This approach is also called *Funk-SVD* or *SVD* in combination with section 2.4.2 and 2.4.4 (Rendle et al., 2019). The algorithm shown above can also be extended. Thus procedures like in section 2.4.5 can be solved. The second method from section 2.4.5 is then also called *SVD++*. A coherent SGD approach was given by Koren and Bell (2011).

2.5.2 Alternating Least Square

The second method often used is *alternating least square* (ALS). In contrast to SGD, the vectors q_i, p_u are adjusted in *two steps*. Since SGD q_i and p_u are both unknown, this is a *non-convex problem*. The idea of ALS is to capture one of the two vectors and work with one unknown variable each. Thus the problem becomes *quadratic* and can be solved optimally. For this purpose the matrix \mathcal{P} is filled with *random numbers* at the beginning. These should be as small as possible and can be generated by a *gaussian-distribution*. Then \mathcal{P} is recorded and all $q_i \in \mathcal{Q}$ are recalculated according to the *least-square problem*. This step is then repeated in reverse order. ALS terminated if a *termination condition* such as the *convergence* of the error is satisfied for both steps (Zhou et al., 2008).

2.5.3 Bayesian Learning

The third approach is known as *bayesian learning*. With this approach the so-called *gibbs-sampler* is often used. The aim is to determine the *common distribution* of the vectors in \mathcal{P}, \mathcal{Q} . For this purpose the *gibbs-sampler* is given an initialization of *hyperparameters* to generate the *initial distribution*. The *common distribution* of the vectors $q_i \in \mathcal{Q}, p_u \in \mathcal{P}$ is approximated by the *conditional probabilities*. The basic principle is to select a variable in a *reciprocal* way and to generate a value dependent on the values of the other variable according to its *conditional distribution*, with the other values remaining unchanged in each *epoch*. A detailed representation of the *gibbs-sampler* was written by [Salakhutdinov and Mnih \(2008\)](#).

References

- Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. 07 2019. ISBN 978-1-4503-6243-6. doi: 10.1145/3298689.3347058.
- Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In P.B. Kantor, F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 107–144. Springer, 01 2011. doi: 10.1007/978-0-387-85820-3_4.
- Simon Funk. Netflix update: Try this at home. <https://sifter.org/~simon/journal/20061211.html>, 12 2006. Accessed: 2019-12-12.
- Jussi Karlgren. *An algebra for recommendations : Using reader data as a basis for measuring document proximity*. Number 179 in SYSLAB technical reports. Department of Computer and Systems Sciences, Stockholm University, 1990.
- Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. pages 426–434, 08 2008. doi: 10.1145/1401890.1401944.
- Yehuda Koren and Robert Bell. Advances in collaborative filtering. In P.B. Kantor, F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 145–186. Springer, 01 2011. doi: 10.1007/978-0-387-85820-3_4.
- Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, 08 2009.
- Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In P.B. Kantor, F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 74–105. Springer, 01 2011. doi: 10.1007/978-0-387-85820-3_4.
- Steffen Rendle, Li Zhang, and Yehuda Koren. On the difficulty of evaluating baselines: A study on recommender systems. *CoRR*, abs/1905.01395, 2019. URL <http://arxiv.org/abs/1905.01395>.
- Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. volume 25, pages 880–887, 01 2008. doi: 10.1145/1390156.1390267.
- Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. pages 337–348, 06 2008. doi: 10.1007/978-3-540-68880-8_32.