Noname manuscript No.
(will be inserted by the editor)

Exploiting recommendation confidence in decision-aware recommender systems

Rus M. Mesas · **Alejandro Bellogín**

Received: date / Accepted: date

Abstract The main goal of a Recommender System is to suggest relevant items to users, although other utility dimensions – such as diversity, novelty, confidence, possibility of providing explanations – are often considered. In this work, we investigate about confidence but from the perspective of the system: what is the confidence a system has on its own recommendations; more specifically, we focus on different methods to embed awareness into the recommendation algorithms about deciding whether an item should be suggested. Sometimes it is better not to recommend than fail because failure can decrease user confidence in the system. In this way, we hypothesise the system should only show the more reliable suggestions, hence, increasing the performance of such recommendations, at the expense of, presumably, reducing the number of potential recommendations. Different from other works in the literature, our approaches do not exploit or analyse the input data but intrinsic aspects of the recommendation algorithms or of the components used during prediction are considered. We propose a taxonomy of techniques that can be applied to some families of recommender systems allowing to include mechanisms to decide if a recommendation should be generated. In particular, we exploit the uncertainty in the prediction score for a probabilistic matrix factorisation algorithm and the family of nearest-neighbour algorithms, the support of the prediction score for nearest-neighbour algorithms, and a method independent of the algorithm.

We study how the performance of a recommendation algorithm evolves when it decides not to recommend in some situations. If the decision of avoiding a recommendation is sensible – i.e., not random but related to the information available to the system about the target user or item –, the performance is expected to improve at the expense of other quality dimensions such as coverage, novelty, or diversity. This balance is critical, since it is possible to achieve a very high precision recommending only one item to a unique user, which would not be a very useful recommender. Because of this, on the one hand, we explore some techniques to combine precision and coverage metrics, an open problem in the area. On the other hand, a family of metrics (correctness) based on the assumption that it is better to avoid a recommendation rather than providing a bad recommendation is proposed herein. In

R.M. Mesas
Telefónica

A. Bellogín
Universidad Autónoma de Madrid
E-mail: alejandro.bellogin@uam.es

summary, the contributions of this paper are twofold: a taxonomy of techniques that can be applied to some families of recommender systems allowing to include mechanisms to decide if a recommendation should be generated, and a first exploration to the combination of evaluation metrics, mostly focused on measures for precision and coverage. Empiric results show that large precision improvements are obtained when using these approaches at the expense of user and item coverage and with varying levels of novelty and diversity.

Keywords Confidence · Decision-aware · Evaluation · Accuracy · Coverage · Novelty · Diversity

1 Introduction

In the last years, the large growth on information in the Web and the increasing number of online services have created new information needs such as providing personalised suggestions of products. Several websites have huge item catalogues, which makes it harder for their clients to choose an interesting item from them. The same problem occurs in other systems, like in social networking systems, where a user should decide who to follow, or in music or movie streaming services, where the client wants to select a song or movie according to her tastes. Because of that, Recommender Systems – that aim at suggesting as many relevant items to the users as possible – are prevalent in this type of websites. Usually, these techniques exploit previous user interactions to suggest new items; in this way, they help users in making better choices, and, at the same time, improve their experience in the system, increase the user satisfaction, and act as a tool to discover all the items in the catalogue, by promoting diverse and novel recommendations, with the aim of increasing their profits.

Nowadays, recommender system techniques are analysed from many different perspectives, although most of the research is focused on producing more sophisticated algorithms able to handle large amounts of data in an efficient way. In general, the main goal of a recommender system is suggesting as many relevant items to the users as possible, although other goals are being considered recently (Gunawardana and Shani 2015): increasing the diversity or novelty of the recommendations, suggesting items in such a way that it is possible to explain where the recommendation is coming from, increasing the confidence of the user in the system, etc. In this work, in order to increase the amount of relevant items being presented to the user, we investigate how the system could measure the confidence on its own recommendations, and, therefore, in this way being aware of making decisions about whether an item should be recommended. Our hypothesis is that, on several occasions, it is better not to recommend rather than producing a bad recommendation, since a bad recommendation may decrease the user confidence on the system. The purpose of this decision making approach is that only items based on consistent, confident data are recommended and presented to the users.

Even though the concept of confidence has already been studied in different aspects of the field (Herlocker et al 2002), we focus on confidence but from the perspective of the system, considering intrinsic aspects of the algorithm. In this way, we propose a taxonomy of techniques for making recommendation decisions that can be applied to some families of recommender systems. In particular, we present how to exploit the prediction uncertainty in matrix factorisation and nearest-neighbour algorithms, the prediction support of the latter recommendation technique, and a method that works independently of the algorithm. Additionally, we also present a technique that exploits such uncertainties to create confidence intervals and recommends by either underestimating or overestimating these intervals.

The main objective analysed in this work is related to the hypothesis that a decision-aware system such as the one we propose here would only show those recommendations

generated from reliable and consistent data, which may lead to an increase in the number of relevant items presented to the user and in her confidence on the recommendations, which may, in turn, also affect negatively other dimensions (Gunawardana and Shani 2015). However, this hypothesis may not properly fit all the user tasks existing in recommender systems and, hence, the proposed methods might only be suitable for a subset of these tasks. More specifically, based on the taxonomy presented in (Herlocker et al 2004), we argue that our approaches are consistent with the *find good items* and *annotation in context* tasks, by presenting only those items with enough confidence in the first case, and filtering out the low-confident items in the second case; in principle, there would be no problem with the *recommend sequence* task. On the other hand, decision-aware recommendations are probably not suitable for the *just browsing* task, since other aspects besides accuracy are more important in this case; similarly, they should not be applied in the *find all good items* task, considering the potential coverage loss caused by our approaches. Finally, for the *find credible recommendation* task a user study might be needed, since it is not clear, *a priori*, that more confident recommendations from the system viewpoint would appear trustworthy to the user.

Nonetheless, considering the different purposes a recommender system can serve (Jannach and Adomavicius 2016), we should compare against other dimensions besides accuracy to further explore the potential of the proposed approaches. As a consequence, our second objective consists on studying how decision-aware algorithms affect different evaluation metrics such as precision, coverage, diversity, and novelty. As a matter of fact, by returning only reliable, confident recommendations, the precision of a recommender might be increased, at the expense of the number of users that receive the recommendations and/or the number of items being recommended, making harder the decision of selecting a recommender evidencing reasonable tradeoffs between both characteristics. Hence, another novelty in this work is the study and development of new techniques and evaluation metrics for a more complete evaluation of these new systems.

In summary, the contributions of this paper are twofold: a taxonomy of techniques that can be applied to any recommendation algorithm or to some families of recommender systems allowing to include mechanisms to decide if a recommendation should be generated, and a first exploration to the combination of evaluation metrics, where a family of metrics accounting for incomplete rankings was defined, mostly focused on measures for precision and coverage. Our results show that it is possible to increase the precision of recommender systems when using decision-aware strategies, however these improvements have to be balanced with a lack of coverage and diversity. Nonetheless, the proposed family of evaluation metrics allows us to compare different systems in a fair, non-parametric way, based on the assumption that it is better to avoid a recommendation rather than providing a bad recommendation.

The remaining of the paper has the following structure. Section 2 provides some background on the different recommendation techniques that will be used throughout this article and presents works related to our approach. Section 3 presents the techniques analysed to embed awareness into the recommendation algorithms about deciding whether an item should be suggested, and Section 4 introduces a technique to exploit the uncertainty when predicting item suggestions to improve the recommendation process. Then, Section 5 describes our proposed evaluation metrics especially designed to evaluate decision-aware algorithms in an unbiased way. Finally, Section 6 reports experiments conducted on three real-world datasets and Section 7 ends with some conclusions and potential future lines of work.

2 Background and Related Work

We present in this section basic formulation and concepts needed for the rest of the paper. Then, we survey some works related to our proposal.

2.1 Recommender systems

Throughout the paper, we use the inherent characteristics of two of the most used recommendation algorithms based on collaborative filtering techniques: k nearest-neighbours (kNN) and matrix factorisation (MF), which will be explained in the next sections.

2.1.1 k Nearest Neighbour (kNN)

Recommender systems based on the k nearest neighbours estimate their predictions for users by computing a distance between users or items, and then using the closest elements to the target user and item. Depending on whether users or items are being used in this process, we would have a user-based kNN or an item-based kNN. In the following, we will only describe the user-based kNN algorithm, the developments for the item-based kNN being very similar (Linden et al 2003; Sarwar et al 2001; Karypis 2001).

In order to compute the rating r_{ui} that user u may provide to item i, user-based nearest-neighbour algorithms exploit the ratings of the k most similar users to u. The subset of k most similar users to such user that voted this item is called u's neighbourhood, $N_i(u)$. For this, we need a similarity measure to account for the distance or closeness between two users, that is, a function $w_{uv} : \mathscr{U} \times \mathscr{U} \mapsto \mathbb{R}$ that, given two users u and v, returns the similarity between them. In this way, once all similarities between user u and the rest of users are computed, the k users with the highest similarities could be selected and a new rating could be estimated. This computation is typically performed by using a weighted average of the neighbours' ratings, using the similarities w_{uv} as weights (Ning et al 2015), as we show in Equation 1:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{1}$$

Furthermore, to take into account that each user may have a different rating behaviour – e.g., some users may provide consistently higher ratings than others –, different normalisation formulations for this equation have been proposed, so the computations are centered in the mean or a Z-score normalisation is applied to the ratings before being used (Ning et al 2015).

As it stems from their definition, the choice of the similarity metric is a key aspect of neighbour-based methods, since it is used to select the users for the neighbourhood, but also because the values are directly weighting the neighbours' ratings when predicting the rating. We now present the two most used user similarity functions in the literature:

– **Cosine similarity:** each user is represented by a $|\mathscr{J}|$-dimensional vector (considering $\mathscr{J}$ denotes the set of items in the system), where coordinate i denotes the rating provided by that user to the i-th item, or 0 if that item has not been rated by the user. Then, the similarity between two users is measured as the cosine distance between the two corresponding user vectors (Ning et al 2015; Ekstrand et al 2011). Equation 2 shows how this similarity is computed between users u and v:

$$\cos(u,v) = \frac{u^\top v}{\|u\|\|v\|} = \frac{\sum_{i\in\mathscr{J}} r_{ui} r_{vi}}{\sqrt{\sum_{i\in\mathscr{J}_u} r_{ui}^2}\sqrt{\sum_{j\in\mathscr{J}_v} r_{vj}^2}} \tag{2}$$

where $u = (r_{u1}, r_{u2}, ..., r_{uJ})$, $v = (r_{v1}, r_{v2}, ..., r_{vJ})$, $\mathscr{J}u$ denotes the items rated by u and $\mathscr{J}_v$ those rated by v.

– **Pearson correlation:** this similarity measure accounts for the tendency that items are rated similarly by two users (Ning et al 2015; Ekstrand et al 2011). Equation 3 shows how this computation is performed, using the same notation as before, and considering $\mathscr{J}_{uv}$ the items rated by both users and $\overline{r}_u$ the average rating of user u.

$$PC(u,v) = \frac{\sum_{i\in\mathscr{J}_{uv}} (r_{uv} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum_{i\in\mathscr{J}_{uv}} (r_{ui} - \overline{r}_u)^2 \sum_{i\in\mathscr{J}_{uv}} (r_{vi} - \overline{r}_v)^2}} \tag{3}$$

2.1.2 Matrix Factorisation (MF)

Another major family of collaborative filtering algorithms conforms the methods based on latent factors, where both users and items are represented by a small number of unobserved or latent factors. Matrix factorisation is one of the most popular techniques to compute latent factors. These algorithms transform users and items into a space of latent factors with D dimensions, and the user-item interactions – ratings in the most typical scenarios in recommendation – are modelled as the scalar product between the user and item vectors in that space (Koren et al 2009; Nakajima and Sugiyama 2011). In other terms, the goal is to decompose the initial rating matrix R into the product:

$$R \approx R' = UI^\top \tag{4}$$

where U is a matrix of size $V \times D$ (with $D << V$) with coefficients for users, and I is the matrix with coefficients for items with size $J \times D$ ($D << J$), where V is the number of users and J the number of items in the system. In this way, the user u is represented by the u-th row of matrix U, the item i is represented by the i-th column of matrix I, and the rating provided by user u to item i is modelled by the product of these two vectors, as we present in Equation 5:

$$r_{ui} = u^\top \cdot i \tag{5}$$

The main problem in these models is how to find the best ranking of the D factors such that R' is as similar as possible to the original matrix R. This, in fact, is translated to a minimisation problem considering the following objective function (Lim and Teh 2007):

$$F(U,I) = \sum_{u,i} (u^\top \cdot i - r_{ui})^2 \tag{6}$$

It is worth noting that not all the entries in matrix R are known, otherwise it would mean all the users have rated every possible item and the recommendation problem would not make sense.

There are several ways of restoring the matrix while avoiding missing values. One approach was proposed by Srebro and Jaakkola in (Srebro et al 2003), where they use the Expectation-Maximisation (EM) algorithm and fill the missing values with predictions from lower range matrices restored in previous iterations. A probabilistic version of the MF problem (PMF) was proposed by Salakhutdinov and Mnih in (Salakhutdinov and Mnih 2011). In this case, the matrix is factorised based on a probabilistic lineal model with Gaussian noise

and the Maximum A Posteriori (MAP) method. Lim and Whye presented in (Lim and Teh 2007) a technique based on variational Bayesian inference to avoid overfitting. Here, all the parameters are estimated using variational inference. Since we will be using this algorithm in the paper, more details about this method are provided next, and a complete derivation is presented in Appendix A.

In this case, by minimising the objective function in Equation 6, we end up computing a predictive distribution on ratings given the observation matrix, where this distribution is approximated by using Bayesian inference, in particular, mean-field variational inference (Lim and Teh 2007). Note that this is one of the few recommendation methods with an analytical, closed-form formulation for the uncertainty of its prediction (standard deviation of the predicted rating) – besides providing an explicit formulation for the average of the predicted scores (the estimated rating) – which we shall later exploit (see Section 3.3):

$$\begin{aligned} \mathbb{E}(\hat{r}_{ui} \mid R) &= \overline{u}^\top \overline{i} \\ Var(\hat{r}_{ui} \mid R) &= \tau^2 + \text{trace}\left(\left(\phi_u + \overline{u}\overline{u}^\top\right)\left(\psi_i + \overline{i}\overline{i}^\top\right)\right) - \overline{i}^\top \overline{u}\overline{u}^\top \overline{i} \end{aligned} \tag{7}$$

where it is assumed the rating follows a normal distribution with mean $u^\top \cdot i$ and deviation τ, and ϕ_u and $\overline{u}$ represent the covariance matrix and vector of means for user u; ψ_i and $\overline{i}$ denote the same concepts but for item i.

2.2 Uncertainty and confidence in recommendation

The concept of confidence in recommendation has been applied to different aspects in the field. On the one hand, confidence is defined on the input data, where some authors have studied which combinations of users or (user, item) pairs let generate better recommendations, namely the *profile-level trust* and *item-level trust* approaches from (O'Donovan and Smyth 2005), or like in (Hu et al 2008), where it is used to interpret the confidence when transforming from implicit data (frequencies) into explicit. In (Latha and Nadarajan 2015), the authors focus on discarding noisy users from the prediction computation by estimating a popularity score for users based on their ratings towards popular items. The concept of noise to improve the confidence has also been used in other works like (Toledo et al 2015), where a fixing strategy is used to correct detected noisy ratings.

On the other hand, we can find different mechanisms that help to contextualise the predictions made by the recommendation algorithms. In this way, in (Herlocker et al 2002) a factor is defined (*significance weighting*) that is combined with the user similarity to devalue those cases where such similarity has been computed with not enough data. In a similar fashion, in (Adomavicius et al 2007) a method is proposed to filter out recommendations according to the rating deviation received by the items. Previously, in (Smyth et al 2002), other authors introduced the concepts of support and confidence in case-based recommenders computed from association rules.

In parallel, in (McNee et al 2003) it was studied how to introduce confidence measures when presenting the recommendations, that is, in the interface the user is interacting with. In that work, even though the confidence metric was very simple (number of ratings observed by the system for one item) it was enough to increase the satisfaction level of the user.

Confidence measures have also been applied to improve the accuracy of popular algorithms, like in (Himabindu et al 2016), where the authors use Conformal Predictions in the classical Item-Based Collaborative Filtering algorithm to associate a confidence measure for each individual prediction, or like in (Mazurowski 2013) where Bayesian intervals are used

to estimate the confidence of individual predictions. In the e-commerce area, (Zhang et al 2014) and (Zhang et al 2016) proposed new product ranking methods which takes uncertainty and prediction confidence into consideration.

In summary, the idea of measuring the confidence of a recommender system has attracted a lot of attention in the field, however, to the best of our knowledge, no other work has combined this concept to embed awareness to make decisions about which items should be recommended, as we present in this work.

2.3 Evaluation of recommender systems

Originally, recommender systems were evaluated using error metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE), largely due to the Netflix prize (Bell and Koren 2007), where the objective was to decrease the RMSE error a 10% with respect to the Netflix algorithm. However, this type of evaluation method has now become outdated because it does not match the user experience (McNee et al 2006). However, this type of evaluation method has now become outdated because it does not match the user experience (McNee et al 2006). As a consequence, Information Retrieval metrics such as precision or recall are used to measure the effectiveness of the rankings generated by recommendation algorithms. Nonetheless, how to evaluate top-N recommendation is still an open problem, where different methodologies could be applied, leading to very different ranges in their outputs (Bellogin et al 2011).

Even though these evaluation metrics are closer to the user experience, by optimizing only this type of metrics (typically aiming for high accuracy) we ignore other concepts important for the interaction with the system such as the discovery of new or surprising items (Castells et al 2015). Because of this, different ways to define novelty and diversity have emerged in the last years. In this context, multi-objective optimisation has been used when different criteria – such as different evaluation dimensions: accuracy, novelty, diversity, etc. – want to be optimised at the same time. Examples of these approaches can be found in (Lacerda 2017), where the authors proposed a multi-objective algorithm that is able to recommend accurate, novel, and diverse personalised lists, or in (Wang et al 2016), where the main goal is to recommend a combination of popular and long-tail items using their multi-objective recommendation framework. More recently, in (Jugovac et al 2017) the authors propose a generic solution creating a parametric scheme to balance accuracy with other quality factors.

However, one particular aspect of the evaluation problem has still not been addressed: incomplete rankings. In a typical situation, a recommender is asked to recommend N items and rank them (*top-N recommendation*); usually this assumes that the recommender always returns at least N items, but it may occur that it only suggests $M < N$ items. This aspect of evaluation has also been discussed in the Information Retrieval literature (Hull 1993), where the author states that in such context, *this can make search methods appear much worse than they actually are*. Because of this, in this paper we address this problem by combining precision and coverage metrics into a single evaluation metric in a formal, principled way, and, more specifically, by taking into account that it is better not to recommend something to a user than providing a bad recommendation. This is different, although related, to the multi-objective optimisation because the problem of incomplete rankings is still present when using standard performance metrics. Nonetheless, the metrics defined in this paper could be used in the future inside of any of the multi-objective recommendation frameworks mentioned before.

3 Embedding Decision-Awareness in Recommender Systems

Modifying recommendation algorithms so that they can decide whether a recommendation should be produced is not an easy task when formulated in a generic way, because each algorithm has different characteristics and hypothesis about the input data and produced suggestions. As a starting point, in this section we shall focus on one of the main types of recommender systems – Collaborative Filtering algorithms.

Different from other works in the literature, our approaches do not exploit or analyse the input data (unlike the works described in (McNee et al 2003; Hu et al 2008)), but intrinsic aspects of the recommendation algorithms or of the components used during prediction are considered, similar to the weights introduced for similarity computation in (Herlocker et al 2002). More specifically, we exploit the uncertainty in the prediction score for a probabilistic matrix factorisation algorithm and the family of nearest-neighbour algorithms (Section 3.3), the support of the prediction score for nearest-neighbour algorithms (Section 3.2), and a method independent of the algorithm (Section 3.1).

As a result, the three decision-aware recommendation techniques presented herein could potentially generate less items than requested by the user, since only those items with enough confidence will be included in the recommendation list. Hence, this effect should be considered when deciding the applicability of the suggestions generated in this way, where some recommendation tasks or domains might be less suitable than others for this type of setting, as discussed in the introduction and in (Herlocker et al 2004).

3.1 Algorithm independent

Interactions between users and items are typically recorded as scores or ratings in a particular scale. This scale may have different values depending on the system, for instance, some social networks let users indicate if something is liked – e.g., Facebook – whereas in other cases a negative opinion might also be expressed – e.g., Reddit or YouTube –, although the most common situation is having a whole range of values (*stars*), from one extreme (*I do not like this item*) to the other (*I really like this item*) – e.g., Google Play, IMDb, or Amazon.

In these situations, when a recommender system is used to predict the user preferences, the output of the recommender is usually in the same range as the input data – although some modifications of state-of-the-art algorithms tend to work better when this constraint is removed (Cremonesi et al 2010). Therefore, it would make sense to apply the explicit semantics behind the input data to the output generated by the recommendation algorithm. Hence, an algorithm could decide that **an item is worth a recommendation only if the predicted score is above a relevance threshold**.

For example, in a domain where user interactions are ratings between 1 and 5, it is usually assumed that values below 3 indicate the item is not relevant for that user; thus, the only recommendations that could be considered valid should be those predicted by the algorithm as a value larger than 3, otherwise, the recommender system would be recommending items that were predicted as not relevant for that user. Note that the same mechanism can also be applied when ratings are not available, as long as a relevance threshold might be applied to the output of the algorithms – for instance, a probabilistic matrix factorisation predicting a score of 0.2 in a one-class scenario could be considered as not relevant (since it is below 0.5) and, hence, should not be recommended.

As a consequence, this approach may produce recommendation lists of different lengths depending on the predictions made by the recommender. For instance, if we use a threshold of 3, this may mean that a user will receive her top-25 recommendations (i.e., item 26th is predicted below 3) whereas another user may only receive her top-7 recommendations

(probably because the highest score returned by the algorithm is lower than 5, and, hence, the relevance threshold is achieved sooner). Hence, embedding decision-awareness in recommender systems has a clear impact on the coverage of the system, as we shall explore with more detail in Section 5.

It is worth clarifying that this methodology is not equivalent to simply changing the cutoff N when evaluating top-N recommendation lists – something already studied in (Herlocker et al 2004). In order to test this approach, the items are ranked as usually according to their predicted score, however, depending on the relevance threshold, those items whose predicted score is lower than such threshold will not be included in the list. Therefore, changing the relevance threshold may or may not have an effect on the recommendation list length, which in turn, would affect the performance measured at different cutoffs. This is because there is no known *a priori* relation between the cutoff N and the relevance threshold, and, in any case, such relation is established in a per-user basis and, hence, a single relevance threshold would not behave uniformly across all users.

3.2 Based on prediction support

It is well known that, in order to compute scores for (user, item) pairs, some algorithms use more data than others, depending on the actual users and items under consideration. A paradigmatic example of this situation are the nearest-neighbour recommenders. These algorithms estimate the user preferences based on similar users or items, however, it is required that those neighbours have rated the same item (in the user-based scenario) or that the target user has rated those similar items (in the item-based case).

Indeed, the number of ratings used to predict the user preferences (denoted as *support*) provides an indication of the confidence level the system has about the produced recommendation, since the system cannot trust in the same way a score produced based only on two neighbours or based on one hundred. This is actually the same idea behind the significance weighting approach proposed by Herlocker and colleagues in (Herlocker et al 2002).

Hence, we propose that a nearest-neighbour recommender could decide that **an item is worth a recommendation if at least n out of the k neighbours have participated in the preference computation**. In other words, a prediction would be ignored if less than n neighbours have contributed to the computation of the prediction score.

3.3 Based on prediction uncertainty

As described in the previous section, some algorithms compute the prediction scores based on an aggregation of values, usually an average or a weighted average (such as the aforementioned neighbour-based recommenders). Whenever an average is being calculated, it is also possible to compute a standard deviation of the predicted score. When doing this, the standard deviation can be interpreted as a confidence parameter of the algorithm about the score: the larger the deviation, the more uncertainty on the prediction, and hence, the lower the confidence on it.

Hence, we propose that an algorithm, for which it is possible to compute the standard deviation of a prediction score, could decide that **an item is worth a recommendation if the standard deviation of the prediction (*uncertainty*) does not exceed a specified threshold σ_τ**.

More specifically, we apply this approach to two families of recommender systems: nearest-neighbour and probabilistic matrix factorisation. We describe next how we can com-

pute the standard deviation in each case, whose general form is

$$Var(\hat{r}_{ui} \mid R) = \mathbb{E}(\hat{r}_{ui}^2 \mid R) - \mathbb{E}(\hat{r}_{ui} \mid R)^2 \tag{8}$$

when estimating rating $\hat{r}_{ui}$ for user u and item i.

First, for the case of a user-based nearest-neighbour algorithm, we use the fact that predictions are generated by a weighted average; hence, we should compute the weighted standard deviation, which has the following form:

$$Var(\hat{r}_{ui}) = \frac{\sum_{v \in N_i(u)} w_{uv}(r_{vi} - \overline{r_{ui}})^2}{V_1 - V_2/V_1} \tag{9}$$

where $\overline{r_{ui}} = (\sum_{v \in N_i(u)} w_{uv} r_{vi})/V_1$, $V_1 = \sum_{v \in N_i(u)} w_{uv}$, $V_2 = \sum_{v \in N_i(u)} w_{uv}^2$, and $N_i(u)$ is the user u's neighbourhood composed of users who rated item i.

Secondly, we make use of the Bayesian approximation for matrix factorisation proposed in (Lim and Teh 2007), although any other algorithm where an explicit formulation for the standard deviation is given could be used instead. In this algorithm, as presented in Section 2.1.2, the preference scores are computed by approximating a distribution, whose average and deviation can be estimated (see Appendix A). To compute the standard deviation of the prediction, we directly use Equation 7 where an analytical derivation of the uncertainty is provided:

$$Var(\hat{r}_{ui}) = Var(\hat{r}_{ui} \mid R) = \tau^2 + \text{trace}\left(\left(\phi_u + \overline{uu}^\top\right)\left(\psi_i + \overline{ii}^\top\right)\right) - \bar{i}^\top \overline{uu}^\top \bar{i} \tag{10}$$

In this way, this approach would decide if an item should be included in the recommendation list by computing its uncertainty ($Var(\hat{r}_{ui})$) according to a recommendation algorithm and comparing it against the threshold σ_τ.

4 A novel method to exploit prediction uncertainty for recommendation

In the previous section, we have presented a method to estimate the confidence or uncertainty in a rating prediction for two families of recommendation algorithms. Besides using this information to decide whether a recommendation should be generated or not (Section 3.3), we now propose how to integrate such confidence into the predicted score.

Let us assume we have estimated the predicted rating with some uncertainty value. In some situations – as in the case of the Variational Bayesian algorithm presented before – the predicted rating corresponds to the mean $\mu(\hat{r}_{ui})$ of a specific distribution, whereas the uncertainty $\sigma(\hat{r}_{ui})$ is its standard deviation. Based on this information, we can compute a *confidence interval on the mean* to estimate how good the predicted rating (mean) is according to the available data. To do this, the lower and upper limits of this interval are calculated by combining the mean and the standard deviation; more specifically: $\mu \pm \lambda\sigma$. Here, for simplicity we will use integer values for λ, even though classical values come from tabular data associated to a particular significance level[1]. In general, there is a one-to-one relation between the value of λ used in a confidence interval and its corresponding significance level, but this relationship will depend on the data distribution; nonetheless, a Normal distribution would usually fit, especially when the sample size is large enough. For a graphical interpretation of these λ values for a Normal distribution see Figure 1.

Once we have estimated a confidence interval based on our predicted score and an uncertainty value, we want to use it in the recommendation process to improve the accuracy of

[1] For instance, in a Normal distribution, $\lambda = 1.96$ for a confidence interval with a significance level of 0.05.

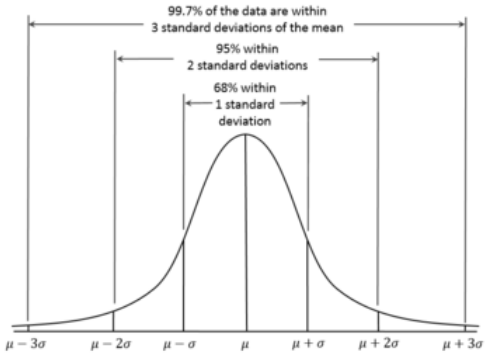

Fig. 1 Probability distribution around the mean for a Normal distribution $N(\mu, \sigma^2)$.

our algorithm. We propose to use the limits of such confidence interval as our final predicted score, as we present in Equation 11. Basically, the sign of the λ value determines whether the lower (negative λ) or the upper (positive λ) bound of the interval is meant to be used. Note that when $\lambda = 0$ this method is reduced to the original formulation.

$$\hat{r}_{ui} = \mu(\hat{r}_{ui}) + \lambda \cdot \sigma(\hat{r}_{ui}), \text{ where } \begin{cases} \text{if } \lambda < 0 \text{ underestimation} \\ \text{if } \lambda = 0 \text{ standard estimation} \\ \text{if } \lambda > 0 \text{ overestimation} \end{cases} \tag{11}$$

An additional interpretation of this approach is the following. Since we apply this transformation to all the items appearing in the ranking for a user, when a negative λ value is used, we are underestimating the predicted rating – i.e., the estimated mean $\mu(\hat{r}_{ui})$ – by substracting the uncertainty $\sigma(\hat{r}_{ui})$ a number of times specified by the λ factor; however, this transformation would have a different effect on the items with higher uncertainty values than on those with lower ones, up to the point of changing the ranking position of highly predicted but uncertain items with respect to other items, ranked lower in the original ranking but with less uncertainty. Similarly, when a positive λ factor is used in Equation 11, we overestimate the predicted score, featuring at top positions those items with higher estimated ratings and higher uncertainties. This might not be seen as a good idea in general, particularly when we are focused on optimising accuracy and precision metrics, but it should help in finding a good tradeoff between items with high predicted values and low uncertainty, and items with low predicted values and high uncertainty, and hence, it could help discover novel or diverse items.

5 Evaluating Decision-Aware Recommender Systems

As soon as a recommender system has control over which items should not be recommended for a particular user, it is very likely that the user coverage and the item coverage decrease, even though precision and other accuracy-related metrics increase. If the decision of avoiding a recommendation is sensible – i.e., not random but related to the information available to the system about the target user or item –, the performance is expected to improve at the expense of other quality dimensions such as coverage, novelty, or diversity. This balance is critical, since it is possible to achieve a very high precision recommending only one item to a unique user, which would not result in a very useful recommender. Memory-based algorithms are well known for suffering from this issue: if very few neighbours are considered, the coverage is lower, but the recommendations are of higher quality (as observed in terms of

error metrics (Herlocker et al 2002)); whereas larger neighbourhoods may increase the likelihood of receiving a noisy recommendation, but the chances of recommending more items are also higher. Because of this, it becomes very important to study and define metrics that, somehow, combine precision and coverage, especially in situations of confidence-awareness like the one we propose in this paper.

As noted by Herlocker and colleagues in (Herlocker et al 2004) there is no general coverage metric that, at the same time, gives more weight to relevant items when accounting for coverage, while combining coverage and accuracy measures. Moreover, Gunawardana and Shani mentioned the problem of balancing coverage and accuracy metrics in (Gunawardana and Shani 2015), and leave it as an open issue in the area. They propose to design an experiment where the precision of two recommenders is being compared after different thresholds are applied to the algorithm. We, on the other hand, aim to address this problem by combining the values of the different metrics to be compared, especially focused on deriving a metric that assess when a recommender does not return an item.

5.1 F-score or Harmonic Mean

The F-score is an evaluation metric very popular in Machine Learning to combine precision and recall measures. It produces the harmonic mean of both metrics, and it ranges between 0 and 1, 0 being the worst value and 1 the optimal value. Based on this idea, we propose to combine precision and coverage through the harmonic mean, whose general formulation is as follows:

$$F_\beta(P,Q) = (1+\beta^2)\cdot\frac{P\cdot Q}{\beta^2\cdot P+Q} \tag{12}$$

where β is used to control the importance of each metric in the final result: if $\beta = 1$ both metrics P and Q have the same importance, whereas if $\beta < 1$ P is more important than Q.

5.2 G-score or Geometric Mean

Instead of using the harmonic mean, we could also use the geometric mean as follows:

$$G_{\alpha_1,\alpha_2}(P,Q) = (P^{\alpha_1}\cdot Q^{\alpha_2})^{1/(\alpha_1+\alpha_2)} \tag{13}$$

where α_1, α_2 control the importance of each metric. In general, the result obtained for the G-score will always be larger (or equal) to the one obtained for the F-score.

5.3 Correctness

The two metrics defined in the previous sections simply combine the result of some evaluation measures; however we believe this is not enough for the problem we want to address. Typical ranking-based metrics – such as precision – assume that no returning an item which was previously asked to predict a rating for, is an advocate of that item being considered as not relevant by a specific recommendation method. However, this is in contrast with the (desired) situation that a recommender may not provide suggestions in some situations due to a low confidence in the accuracy of such predictions (Gunawardana and Shani 2015; Herlocker et al 2004).

We propose an evaluation metric that is able to assess when a recommender decides not to recommend a specific item. To do this, we adapt an extension of accuracy proposed in the context of Question Answering by Peñas and Rodrigo in (Peñas and Rodrigo 2011). In that work, the authors assume that there are several questions to be answered by a system, each question has several options, but one (and only one) of those options is correct. If it is possible to give no response for a given question, this action should not be correct, but not incorrect either. Hence, the authors propose a general formulation giving a different weight to the value of unanswered questions:

$$P(C) = P(C \cap A) + P(C \mid \neg A)P(\neg A) = \frac{n_{ac}}{n} + \frac{n_{ac}}{n}\frac{n_u}{n} \tag{14}$$

This metric satisfies the following basic properties:

1. A system that answers all the questions receives the same final score as if we use the standard precision (n_{ac}/n).
2. A system that answers no question ($n_{ac} = 0$) receives 0 as its final score.
3. The amount of unanswered questions add value to the metric, at the same proportion as the correctly answered questions; this means that the unanswered responses of a system with low accuracy do not add as much value to the metric as when considering a system with high accuracy.

To apply this evaluation metric to recommendation, we first assume that the set of recommenders we want to compare will receive the same list of items to be ranked, a standard situation shared by many evaluation methodologies (Bellogin et al 2011). Then, the equivalence between a Question Answering system and a recommender is made – in a user basis – by considering each recommendation algorithm as a different system that will answer (or not) the questions available, represented as the candidate items to be ranked by a specific methodology. In the following sections, we define four different instantiations of this metric, two of them based on users and two for items.

5.3.1 User-based correctness

Let us assume each user receives a list with $\hat{N} \leq N$ recommendations from system r, that is, $L_r^N = (i_1, ..., i_{\hat{N}})$. Such a list is made up of:

– **Relevant items** for user u, considering as relevant those items rated by the user and included in the test set, $Te(u)$, optionally filtering out the items by the rating value that appears in the test set so only those above a threshold are considered:

$$TP = \sum_{i \in L_r^N(u)} \mathbb{1}(i \in Te(u)) = \sum_{i \in L_r^N(u)} \mathbb{1}_{Te(u)}(i) \leq N \tag{15}$$

– **Not relevant items** for user u, i.e., those in the list but not rated by u in the test set:

$$FP = \sum_{i \in L_r^N(u)} (1 - \mathbb{1}_{Te(u)}(i)) \leq N \tag{16}$$

– The user may receive less than N recommendations, denoting as $\neg A = NR$ the 'gaps' in the returned list or unanswered recommendations, satisfying

$$NR = N - (TP + FP) \tag{17}$$

with $0 \leq NR \leq N$. Hence, if $\hat{N} = N$ then $NR = 0$.

Table 1 Toy example to compare precision (*P*) and user correctness (*UC*). In bold we show the best results for each metric. Recommendation lists (a)-(f) consist of 5 elements ($N = 5$), two of them are relevant for this user (❶ and ❺).

Recommendation lists	Evaluation metrics	
	P	UC
(a) ❶②③④❺	**0.40**	0.40
(b) ❶②③	0.20	0.28
(c) ❶	0.20	0.36
(d) ②	0.00	0.00
(e)	0.00	0.00
(f) ❶❺	**0.40**	**0.64**

Our goal is to somehow reward not recommending instead of recommending something not relevant. In this way, a first instantiation of our metric arises: *User Correctness* (*UC*), defined in Equation 18:

$$\text{UserCorrectness}(u,r,N) = UC(u,r,N) = \frac{1}{N}\left(TP + TP\frac{NR}{N}\right) \in [0,1] \tag{18}$$

To better understand this metric, Table 1 shows a small example where explicit values of the metric are presented for six different recommendation lists. Besides, a comparison against standard precision metric is included. Among all the lists presented, the one with a higher value according to *UC* is list (f), since the two returned items are relevant for user u. However, precision achieves the same score with list (a) and list (f), since this metric is not able to consider that the first list contains 3 not relevant items.

Now let us suppose that two users, u_1 and u_2, receive only two items as the recommendations from a system that returns lists up to 5 elements. In both cases the system has decided to not recommend 3 items. Nonetheless, whereas u_1 only had those two items in her set of relevant items $Te(u_1)$, u_2 had another 10 items that could have been recommended. Despite this difference, according to *UC* both users would receive a score of 0.64. However, if we want to also consider the total amount of relevant items available for recommendation – a concept similar to the well-known ranking metric *recall* (Baeza-Yates and Ribeiro-Neto 2011) – a second instantiation becomes available where we include the assumption that it is worse to not recommend items when there are still several items available: *Recall User Correctness*, defined in Equation 19.

$$\text{RecallUserCorrectness}(u,r,N) = RUC(u,r,N) = \frac{1}{N}\left(TP + \frac{TP}{Rel}NR\right) \in [0,1] \tag{19}$$

where $Rel = |Te(u)|$ represents the number of relevant items available in the test set for user u. In the previous example, this second metric *RUC* for user u_1 would produce a value of 1.0, whereas for u_2 this value would only be 0.5.

5.3.2 Item-based correctness

Besides user coverage, the correctness metric is also able to integrate item coverage, which is related to the diversity of recommendations (Castells et al 2015). This property is interesting because, as we shall see in the next sections, decision-aware recommendation algorithms evidence not only a decrease on user coverage, but also on item coverage. Because of that, we now present how we can derive an item-based correctness variation.

First, we need to define the following sets, based on the recommendation lists $L_r^N(u)$ received by each user:

$$S(i,r,N) = \{u : i \in L_r^N(u)\} \tag{20}$$

In this case, $S(i,r,N)$ denotes the set of all users having item i in her recommendation list – for a system r and considering N as the length of the recommendation lists. Then, using the ground truth test set, we define the following sets:

$$T(i) = \{u : i \in Te(u)\} \tag{21}$$

This set denotes the users in the test set for whom item i is relevant. Based on this, for each item we can define:

– Users who received item i in their list, and this item is relevant:

$$TP = S(i) \cap T(i) \tag{22}$$

– Users with i in the list and i is not relevant:

$$FP = S(i) \cap \overline{T(i)} \tag{23}$$

– Users without i in the list but i is relevant:

$$FN = \overline{S(i)} \cap T(i) \tag{24}$$

– Users without i in their list and i is not relevant for them:

$$TN = \overline{S(i)} \cap \overline{T(i)} \tag{25}$$

Based on these equations, we can compute the information about the number of users for whom item i was not recommended (Equation 26) and the number of users for whom item i is relevant (Equation 27).

$$NR = TN + FN \tag{26}$$

$$Rel = TP + FN = |T(i)| \tag{27}$$

Once these different sets have been defined, we propose the metrics ItemCorrectness (*IC*, Equation 28) and RecallItemCorrectness (*RIC*, Equation 29), defined similarly as in the case for users:

$$\text{ItemCorrectness}(i,r,N) = IC(i,r,N) = \frac{1}{|U|}\left(TP + \frac{TP}{|U|}NR\right) \tag{28}$$

$$\text{RecallItemCorrectness}(i,r,N) = RIC(i,r,N) = \frac{1}{|U|}\left(TP + \frac{TP}{Rel}NR\right) \tag{29}$$

An important issue with the *IC* metric, that will become more obvious later in the experimental part, is that, since the number of users in the system is in the denominator, the values returned by this metric tend to be very small. On the other hand, since the *RIC* metric considers the ratio of users to whom relevant items were recommended – not taking into account those users where a particular item was not relevant – the values returned by this variation are, in general, higher.

5.3.3 Averaging correctness metrics

So far, the four instantiations of the correctness metric for recommendation defined before allow us to compute a score for each user (in the case of *UC* and *RUC*) and item (for *IC* and *RIC*). In order to obtain values for the whole system, the standard procedure – as is done for ranking-based metrics like precision or recall – is to compute a value for every user, and then aggregate all these values by computing an arithmetic mean, as we show in Equation 30:

$$\text{MeanUserCorrectness}(r,N) = \text{MeanUC}(r,N) = \frac{\sum_{u \in \text{UsrRec}(r)} UC(u,r,N)}{|\text{UsrRec}(r)|} \tag{30}$$

where $\text{UsrRec}(r)$ denotes the users with at least one recommendation from system r.

We can observe that, in this case, we do not consider those users that receive no recommendation at all. Note that the main idea for this family of metrics is to reward those systems that do not respond rather than those that respond incorrectly, however, it is also important to penalise whenever the system does not return nothing. Because of this, we propose a slight modification of Equation 30, where we substract the ratio of users without a recommendation, as we show in Equation 31:

$$\begin{aligned} \text{UserCorrectness}(r,N) &= \text{MeanUC}(r,N) - \text{MeanUC}(r,N) \cdot (1 - USC) \\ &= \text{MeanUC}(r,N) \cdot USC = \text{MeanUC}(r,N) \cdot \frac{|\text{UsrRec}(r)|}{V_r} \\ &= \frac{\sum_{u \in \text{UsrRec}(r)} UC(u,r,N)}{V_r} \end{aligned} \tag{31}$$

where USC denotes the user space coverage (or user coverage) and V_r is the number of users in the whole recommendation system r. In this way, we are explicitly accounting for the user coverage in the metric.

Regarding the *RUC* metric, we propose to use the same formulation as the one introduced in Equation 31, so that we combine in the same metric precision, recall, and user coverage. Furthermore, for the item-based correctness metrics, a similar procedure where the total number of items is considered in the aggregation can be developed.

Finally, the four instantiations of the correctness metric for recommendation are tailored to evaluate rankings or lists of recommendations; therefore, they can be computed at different cutoffs or positions in the rank by simply changing the value of N in the different formulations and considering the aggregation functions presented here; this means that the typical notation $UC@N$ would be computed as $UC@N(r) = \text{UserCorrectness}(r,N)$.

6 Experiments and Results

6.1 Experimental Settings

In this paper we have used three datasets from two different domains: two versions of the MovieLens (ML)[2] (ML-100K and ML-1M) dataset and Jester[3]. ML-100K includes $100,000$ ratings by 943 users on $1,681$ items (movies), ML-1M contains $1,000,209$ ratings by $6,040$

[2] https://grouplens.org/datasets/movielens/

[3] http://eigentaste.berkeley.edu/dataset/

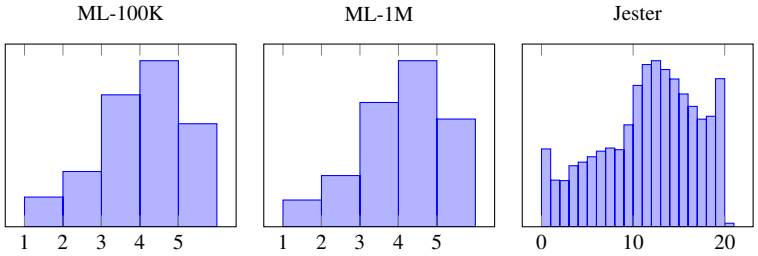

Fig. 2 Rating distribution for each dataset: ML-100K, ML-1M, and Jester.

users on 3,883 movies, and Jester includes 1,710,677 ratings on 150 items (jokes) by 59,132 users. The rating scale on the first two datasets is $[1,5]$, and on Jester is $[-10,10]$ (that we moved into $[0,20]$ to avoid negative ratings). Figure 2 shows the rating distribution for each dataset; we observe all of them tend to be biased towards positive ratings, however Jester also includes a large amount of very negative ratings.

Some of the algorithms used in the experiments are based on implementations found in RankSys[4], specifically, the nearest-neighbour algorithms and their modifications. The probabilistic matrix factorisation method was implemented by ourselves.

Finally, regarding the evaluation, two publicly available frameworks were used: RankSys and RiVal[5]. On top of the latter framework we implemented the following evaluation metrics: User Space Coverage representing the ratio of users that have received at least one (USC) or N ($USC@N$) items as recommendations, Item Space Coverage representing the ratio of items that were recommended to any user by a system returning at most N items ($ISC@N$), and the Correctness metrics as defined in Section 5.3: $UC@N$ (User Correctness), $RUC@N$ (Recall User Correctness), $IC@N$ (Item Correctness) and $RIC@N$ (Recall Item Correctness). The RankSys framework was used to obtain novelty and diversity metrics, specifically EPC and AggrDiv (Vargas and Castells 2011).

Unless stated otherwise, all the reported metrics are computed at cutoff 10 (i.e., $N = 10$). Furthermore, we considered every item included in the test set as relevant for that user, meaning that we do not require any threshold to define a relevant item for a user. While we agree that this setting might be counter-intuitive (since recommending something the user rated with a very low rating will be considered as a positive recommendation), in some preliminary experiments we observed almost no difference in the general results, meaning that the ranking of the algorithms remained the same, even though the actual values of the evaluation metrics could be different. This fact is consistent with previous works in the area (Herlocker et al 2004; Bellogin et al 2011).

6.2 Performance of decision-aware strategies

In this section we evaluate the performance of the different decision-aware strategies presented in Section 3 using the evaluation metrics described in Section 5.

Let us start with the algorithm independent strategy. Recall that this strategy only includes in the recommendation list those items whose predicted score is above a relevance threshold γ. Tables 2 and 3 show the results when using this strategy. First, in Table 2 we compare different evaluation metrics for the kNN algorithm. We observe that the change in

[4] https://github.com/RankSys/RankSys
[5] https://github.com/recommenders/rival

Table 2 Comparison of performance metrics when using a decision-aware strategy based on relevance threshold (γ), for a *k*NN algorithm on ML-100K.

γ	P	USC	ISC	F_1	F_2	$F_{0.5}$	$G_{1,1}$	$G_{1,2}$	$G_{2,1}$	UC	RUC	IC	RIC
1	**0.037**	**100.0**	**62.1**	**0.071**	**0.159**	**0.045**	**0.191**	**0.332**	**0.110**	**0.037**	**0.037**	**0.000**	**0.015**
2	0.037	100.0	62.1	0.071	0.159	0.045	0.191	0.332	0.110	0.037	0.037	0.000	0.015
3	0.037	100.0	62.1	0.071	0.159	0.045	0.191	0.332	0.110	0.037	0.037	0.000	0.015
4	0.036	100.0	62.1	0.070	0.159	0.045	0.191	0.331	0.110	0.036	0.036	0.000	0.015
5	0.020	99.2	60.0	0.040	0.094	0.025	0.142	0.272	0.074	0.022	0.022	0.000	0.011

Table 3 Comparison of performance metrics when using a decision-aware strategy based on relevance threshold (γ), for a probabilistic matrix factorisation algorithm on ML-100K.

γ	P	USC	ISC	F_1	F_2	$F_{0.5}$	$G_{1,1}$	$G_{1,2}$	$G_{2,1}$	UC	RUC	IC	RIC
1	**0.093**	**100.0**	**22.7**	**0.170**	**0.338**	**0.113**	**0.304**	**0.453**	**0.205**	**0.093**	**0.093**	0.001	0.009
2	0.093	100.0	22.7	0.170	0.338	0.113	0.304	0.453	0.205	0.093	0.093	0.001	0.009
3	0.093	99.9	22.7	0.170	0.338	0.113	0.304	0.452	0.205	0.093	0.093	**0.001**	**0.009**
4	0.086	97.8	22.3	0.158	0.317	0.105	0.290	0.434	0.193	0.086	0.085	0.001	0.007
5	0.024	59.0	15.5	0.047	0.104	0.030	0.120	0.204	0.070	0.018	0.015	0.000	0.002

coverage is not significant, while the performance remains steady. Hence, there is no balance to solve, and the algorithm with the highest precision ($\gamma = 1$) is the clear winner. In this situation, the proposed metrics UC and IC are not necessary since the maximum precision and coverage values are obtained for the same parameter ($\gamma = 1$); however, they are included to check they behave consistently.

Then, Table 3 shows these results for VB, where a very similar situation is observed. In this context, we hypothesise that, because we are measuring precision at high positions (recall the cutoff is 10), removing those items predicted by the recommender as not relevant (score $\leq \gamma$) has almost no effect in the final ranking, as evidenced in these results, where lower γs obtain the best results.

Now, when we apply the strategy based on prediction support, a similar situation occurs – see in Table 4 the results for ML-100K. User coverage remains almost unchanged until $n \geq 7$, although precision increases even for smaller values of n until $n = 7$. Considering this information, the three versions of the harmonic mean, G, and $G_{2,1}$ all agree on the ranking of the systems, where the best algorithms are those with n equals 5, 6, and 4, in that order. We observe that UC and RUC do not discriminate much more than that, however, when IC and RIC are analysed, the best recommenders are not the same as before ($n = 4,5$) which makes sense because these techniques take the item coverage into account, which decreases more abruptly than the user coverage. Similarly, $G_{1,2}$ also changes the ranking of systems because it gives a higher weight to coverage than precision.

We now analyse the decision-aware strategy based on prediction uncertainty. Due to space constraints we shall focus on the results for the probabilistic matrix factorisation algorithm, and skip those from the nearest-neighbour recommender. As we show in Table 5, this strategy evidences a strong tradeoff between coverage and precision, since introducing a threshold of 0.82 increases the performance by a factor of 4 but reduces the coverage a 70% with respect to no threshold. This situation is very interesting, because the optimal recommender depends on the evaluation metric: for instance, $\sigma_\tau = 0.84$ obtains the highest value for F_1 and $G_{2,1}$ but not in the other metrics, this is because they are too sensitive to the precision value, since that recommender achieves the second best value. This example evidences the differences between the proposed correctness metrics and the other combination met-

Table 4 Comparison of performance metrics when using a decision-aware strategy based on prediction support, for a nearest-neighbour recommender on ML-100K.

n	P	USC	ISC	F_1	F_2	$F_{0.5}$	$G_{1,1}$	$G_{1,2}$	$G_{2,1}$	UC	RUC	IC	RIC
1	0.037	**100.0**	**62.1**	0.070	0.159	0.045	0.191	0.332	0.110	0.037	0.037	0.000	0.015
2	0.133	100.0	46.9	0.234	0.433	0.160	0.364	0.510	0.260	0.133	0.133	0.002	0.021
3	0.188	100.0	39.5	0.317	0.537	0.225	0.434	0.573	0.329	0.189	0.189	0.002	0.026
4	0.230	100.0	35.1	0.374	0.599	0.272	0.480	0.613	0.376	0.234	0.236	0.003	0.029
5	**0.245**	99.7	32.3	**0.393**	**0.618**	**0.288**	**0.494**	**0.624**	**0.391**	**0.259**	**0.266**	**0.003**	**0.029**
6	0.241	96.4	28.5	0.386	0.603	0.284	0.482	0.607	0.383	0.257	0.263	0.003	0.026
7	0.237	85.9	24.8	0.371	0.563	0.277	0.451	0.559	0.364	0.231	0.231	0.002	0.023
8	0.226	66.9	21.7	0.338	0.480	0.260	0.389	0.466	0.324	0.180	0.171	0.002	0.018

Table 5 Comparison of performance metrics when using a decision-aware strategy based on prediction uncertainty, for a probabilistic matrix factorisation algorithm on ML-100K.

σ_τ	P	USC	ISC	F_1	F_2	$F_{0.5}$	$G_{1,1}$	$G_{1,2}$	$G_{2,1}$	UC	RUC	IC	RIC
–	0.093	**100.0**	22.7	0.170	0.338	0.113	0.304	0.453	0.205	0.093	0.093	0.001	0.009
0.82	**0.326**	28.2	9.1	0.303	0.290	**0.316**	0.303	0.296	0.311	0.100	0.094	0.001	0.006
0.84	0.283	59.0	15.1	**0.382**	0.484	0.316	0.408	0.462	**0.361**	0.174	0.170	0.002	0.011
0.86	0.214	80.9	19.6	0.338	**0.520**	0.251	**0.416**	0.519	0.333	**0.177**	**0.176**	**0.002**	0.012
0.88	0.181	95.6	22.2	0.304	0.514	0.216	0.415	**0.548**	0.315	0.176	0.176	0.002	**0.013**
0.90	0.165	99.5	24.8	0.283	0.495	0.198	0.405	0.546	0.300	0.165	0.165	0.002	0.013
0.92	0.156	100.0	26.0	0.269	0.480	0.187	0.395	0.538	0.289	0.156	0.156	0.002	0.012
0.94	0.145	100.0	27.3	0.254	0.459	0.175	0.381	0.526	0.276	0.145	0.145	0.002	0.011
0.96	0.139	100.0	28.2	0.245	0.447	0.168	0.373	0.518	0.269	0.139	0.139	0.002	0.011
0.98	0.133	100.0	**28.6**	0.235	0.435	0.161	0.365	0.511	0.261	0.133	0.133	0.002	0.011

Table 6 Summary of optimal precision (P), coverage (USC), and correctness (UC) values along with the optimal parameters found in the three datasets tested for the three decision-aware strategies presented in Section 3.

Dataset	Metric	Algorithm independent				Prediction support		Prediction uncertainty			
		kNN	γ^*	VB	γ^*	kNN	n^*	kNN	σ_τ^*	VB	σ_τ^*
ML-100K	P	0.037	1	0.093	1	0.245	5	0.037	–	0.326	0.82
	USC	100	1	100	1	100	1	100	–	100	–
	UC	0.037	1	0.093	1	0.259	5	0.037	–	0.177	0.86
ML-1M	P	0.023	1	0.055	3	0.243	6	0.023	–	0.189	0.82
	USC	100	1	100	1	100	1	100	–	100	–
	UC	0.023	1	0.055	3	0.245	6	0.023	–	0.115	0.84
Jester	P	0.289	6	0.165	9	0.308	5	0.260	–	0.436	3.8
	USC	98.2	1	100	1	98.2	1	0.982	–	100	–
	UC	0.280	1	0.162	1	0.326	4	0.280	–	0.212	4.5

rics: whereas the latter metrics simply combine two values, the former ones include further assumptions that, in principle, help to interpret the comparisons between recommenders. According to these results, $\sigma_\tau = 0.84$ is preferred over $\sigma_\tau = 0.86$ by F_1, but UC inverts this relation; we can infer that $\sigma_\tau = 0.86$ is better suited for deciding when an item should be recommended, since we are rewarding unanswered recommendations above incorrect recommendations.

In summary, decision-aware strategies usually help improving the performance of the recommendation techniques, as we have seen in the previous results for ML-100K. Table 6 shows a summary of the best results obtained for the rest of the datasets. We observe that the algorithm independent strategy produces almost no tradeoff, as discussed previously, and hence the optimal parameters are in most cases the same, regardless of the evaluation metric. The same situation occurs with the prediction uncertainty for kNN, where the highest values are found for the baseline (no decision-aware is used). On the other hand, when prediction support or prediction uncertainty (in VB) strategies are used, we obtain high improvements both in terms of P and UC.

Nonetheless, it should be noted that some of these metrics – especially *IC* – achieve very low values. This is due to, as stated in their description, these metrics need to be normalised by the number of users whenever we want to compare across different systems and algorithms, as we present here. However, let us suppose that a system designer wants to use this metric, they would probably be only interested in comparing algorithms inside the same recommendation system, and hence, using the same number of users. Because of this, such designer could remove the outer normalisation by the number of users in the formulations for $IC(i,r,N)$ and $RIC(i,r,N)$ (Equations 28 and 29), this would give more significant figures when averaging these metrics for all the items in the system. However, if we do not want (or we cannot) modify those equations the interpretation of these metrics is very limited, as with other ranking metrics (Bellogin et al 2011). At the end, since the information we have from the users is far from complete, most evaluation metrics are only useful to rank the recommender systems *under the same situations* (under- or over-estimating their quality, depending on the completeness of the groundtruth information).

It is worth noting that in more general scenarios, for instance, as we show in Table 5, the metrics do not need to agree on the optimal recommender, but this is because each metric is actually measuring a different nuance of what quality means. In this example, *UC*, *RUC*, and *IC* agree that the best recommender is the one with $\sigma_\tau = 0.86$, however, *RIC* produces a higher value for $\sigma_\tau = 0.88$. These results may indicate that a value of $\sigma_\tau = 0.86$ produces incomplete rankings but with more relevant items than other configurations; however, when we consider the ratio of users to whom relevant items were recommended, $\sigma_\tau = 0.88$ produces slightly better results. Hence, in the end, depending on what criteria we optimise for, we should select the recommenders according to a different evaluation metric.

Finally, if we want to extrapolate these results to obtain a set of guidelines that can be general enough for different systems, a user study would be required to measure the sensitivity levels of these metrics, that is, whether a difference of $\pm\varepsilon$ is noticed by an actual user or not. Once that information is available, proper and realistic guidelines could be proposed, matching the user experience with the system and the measurements provided by the correctness metrics. This is something we would like to explore in the future and is out of the scope of this work.

6.3 Performance of strategies based on confidence intervals

In this section we explore how the strategies based on confidence intervals may help improving the performance of decision-aware recommendation algorithms. Recall that this approach combines the mean and standard deviation of the predicted rating in such a way that the lower or upper limit of the corresponding confidence interval is used instead, depending on the value of λ (see Section 4).

A summary of the results for these strategies is presented in Figures 3 (for the *k*NN algorithm) and 4 (for the probabilistic matrix factorisation method). Here, we show the value obtained by each algorithm without using any decision-aware strategy in the horizontal line, which in fact matches the point corresponding to $\lambda = 0$ and *None* for the confidence threshold.

We observe that, when the confidence threshold is not used – hence, we are not using the uncertainty prediction strategy, only the confidence interval – there is always some value of λ that improves the performance, even twice the value. It is worth noting that this improvement is performed while the coverage remains unchanged (exhaustive results as the ones presented before are omitted due to space constraints). In general, positive values of λ work better for *k*NN, whereas the opposite is true for the Variational Bayesian method.

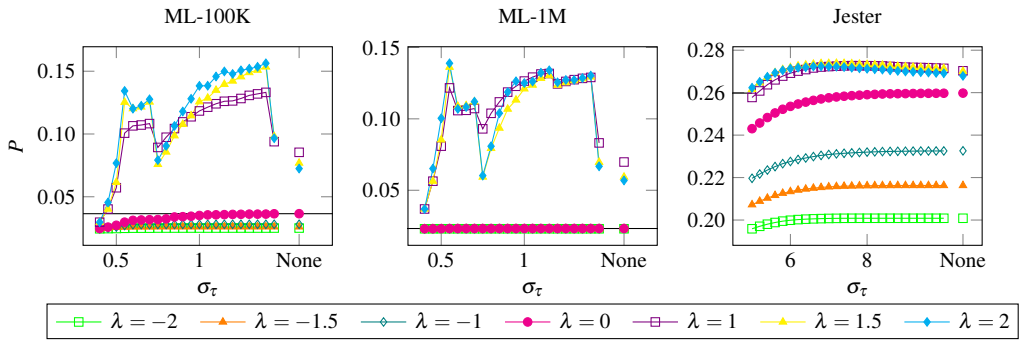

Fig. 3 Precision (*P*) when using an strategy based on **confidence intervals** (points over the *None* label), and a combination of this strategy and the **prediction uncertainty** strategy for a *k*NN algorithm. Baseline value (no strategy) is presented as the horizontal line.

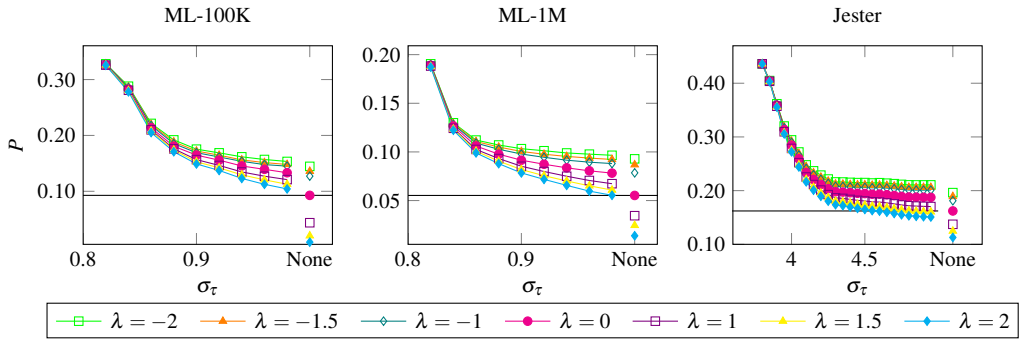

Fig. 4 Precision (*P*) when using an strategy based on **confidence intervals** (points over the *None* label), and a combination of this strategy and the **prediction uncertainty** strategy for a Variational Bayesian algorithm. Baseline value (no strategy) is presented as the horizontal line.

When we combine the strategy based on confidence intervals with a decision-aware technique based on uncertainty, we find that the situation is more homogeneous for the Variational Bayesian method, whereas *k*NN is more erratic and the trends depend on the dataset. Hence, Figure 4 shows the same behaviour for the three datasets: whenever the constraints on σ_τ are more strict, precision increases, the same result observed before in Table 5; at the same time, underestimating the predicted rating (negative part of the λ-spectrum) works better, although in general the baseline is outperformed in almost every case in the three datasets.

On the other hand, Figure 3 presents a more erratic behaviour for *k*NN, probably because the average and standard deviation are computed empirically, instead of coming from sound theoretical models, as in the case of the Variational Bayesian algorithm. In this case, as mentioned before, overestimating the predicted rating improves the performance of this method in every dataset. This might be attributed to this method promoting those ratings predicted by more neighbours, which, as we saw in Table 4, tend to be more precise, even though they may have larger deviations, which are thus exploited by the use of positive λs.

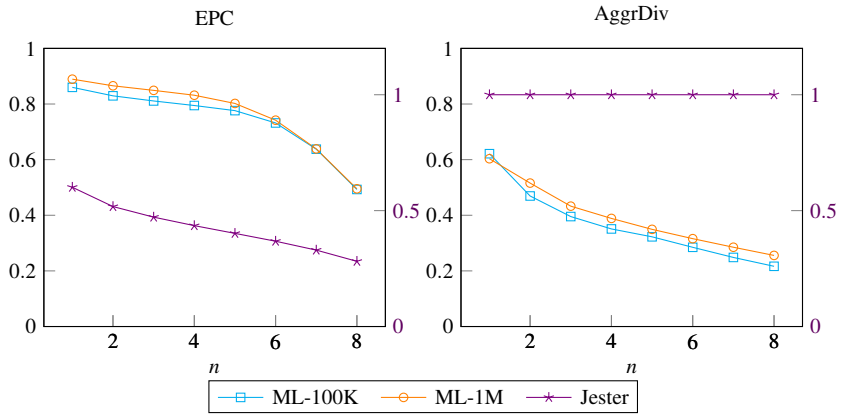

Fig. 5 Novelty and diversity metrics for a decision-aware strategy based on prediction support. Jester is plotted in the secondary (right) axis.

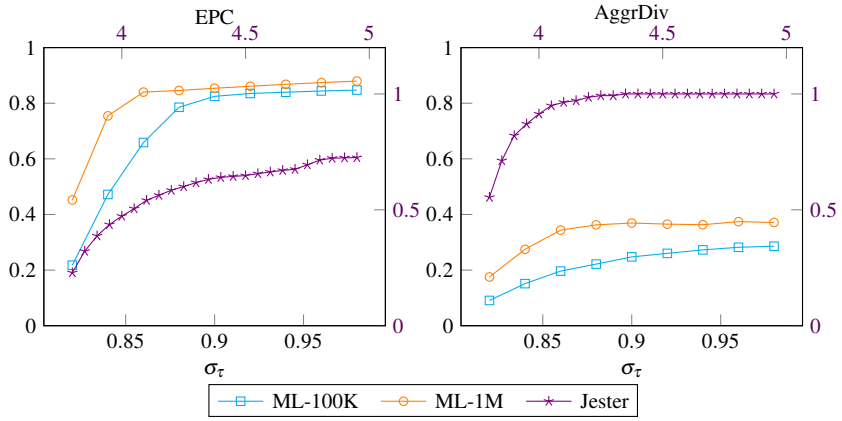

Fig. 6 Novelty and diversity metrics for a decision-aware strategy based on prediction uncertainty. Jester is plotted in the secondary (right and upper) axis.

6.4 Impact on beyond-accuracy metrics

In this section we explore what is the impact on beyond-accuracy evaluation metrics – such as diversity and novelty – of the proposed decision-aware strategies. As we already discussed in the previous section, the algorithm independent strategy does not exhibit a trade-off between precision and coverage; hence, the impact this strategy imposes on diversity and novelty metrics is negligible. On the other hand, the rest of the strategies have a clear effect on them, as we present next.

Figure 5 shows the impact that a decision-aware strategy based on prediction support has on novelty and diversity. We observe that, for larger n, both the diversity of the lists and the novelty decrease (which means that the recommended items are more and more popular). The rationale behind these results is that, when the constraint n becomes more strict, more users are required to have seen (rated) those items, which, in the long term, produces that more popular items are being recommended. This behaviour is consistent for the three analysed datasets, except for Jester's diversity, because of its very small catalogue.

Figure 6 compares novelty and diversity evolution for different thresholds using a decision-aware strategy based on prediction uncertainty for the Variational Bayesian algorithm – we

omit the results for the *k*NN algorithm due to space constraints. We observe that, similarly as in the previous strategy (Figure 5), when the constraints are more strict (here this means that σ_τ decreases) both novelty and diversity decrease in the three tested datasets. In this case the rationale is slightly different to what happened in the previous strategy: those items with a lower standard deviation seem to correspond with popular items (like before), however, this constraint really imposes a limit on the number of different items that can be recommended, which ends up producing very low diversity scores.

Finally, Figure 7 shows the effect of exploiting prediction uncertainty in novelty and diversity metrics. Following the same notation as in Figures 3 and 4, this figure presents the evolution of diversity and novelty metrics when different values of λ are used in an isolated way (points over the *None* label), and whenever the decision-aware strategy based on prediction uncertainty is applied. We only present the results for the Variational Bayesian algorithm because of the erratic behaviour of the *k*NN algorithm in this experiment.

We observe, as in the previous experiment, that when the constraints are more strict (σ_τ decreases) both novelty and diversity decrease in the three tested datasets, and this effect produces a clear tradeoff with accuracy: those methods that retain more diversity and novelty (larger λs) correspond to those that were producing worse recommendations (lower precision values, as shown in Figure 4). However, it is worth noting that when the prediction uncertainty strategy is not used, the effect of confidence intervals in isolation is very different depending on the evaluation metric: whereas overestimating ($\lambda > 0$) produces higher novelty values, it generates items with lower diversity. This result may indicate that, in this recommender, overestimating the rating prediction generates a very limited number of items (low diversity), although these items are novel for the users (high novelty) but not relevant for them (low precision).

In summary, we have found that decision-aware strategies have a profound impact on beyond-accuracy metrics, in particular, in the diversity and novelty dimensions measured by EPC and AggrDiv. In general, most of the evaluated strategies evidence a loss on novelty and diversity when the constraints are more strict, either for those based on prediction uncertainty (when less uncertainty is allowed in the recommendations) or those based on prediction support (where a prediction needs to receive more support from the recommender so it decides to return it). This leads to the following conclusion: the cost of producing more confident recommendations is the generation of more popular (less diverse and novel) items, which degenerates into non-personalised algorithms such as returning the most popular items. Finding a good tradeoff between obvious and interesting suggestions is still an open problem; to some extent, this issue might be addressed by means of the methods presented herein, for instance, by selecting the amount of decision-awareness we want to incorporate into the algorithm.

7 Conclusions and Future Work

In this work, we have studied how to increase the system level confidence on its own recommendations by making the system aware of the decisions taken; we expect these improvements to have an impact on users' confidence however this should be validated in the future with user studies. For this, we have proposed three strategies to decide if an item should be included in a recommendation list of a specific user, based on the consistency and reliability of the data that will be used by the recommender system to estimate the preferences. These strategies (one based on the support of the prediction, other on its uncertainty, and another that can be applied to any recommendation algorithm) have been evaluated in terms of precision, coverage, novelty, and diversity. We have also proposed another technique that exploits

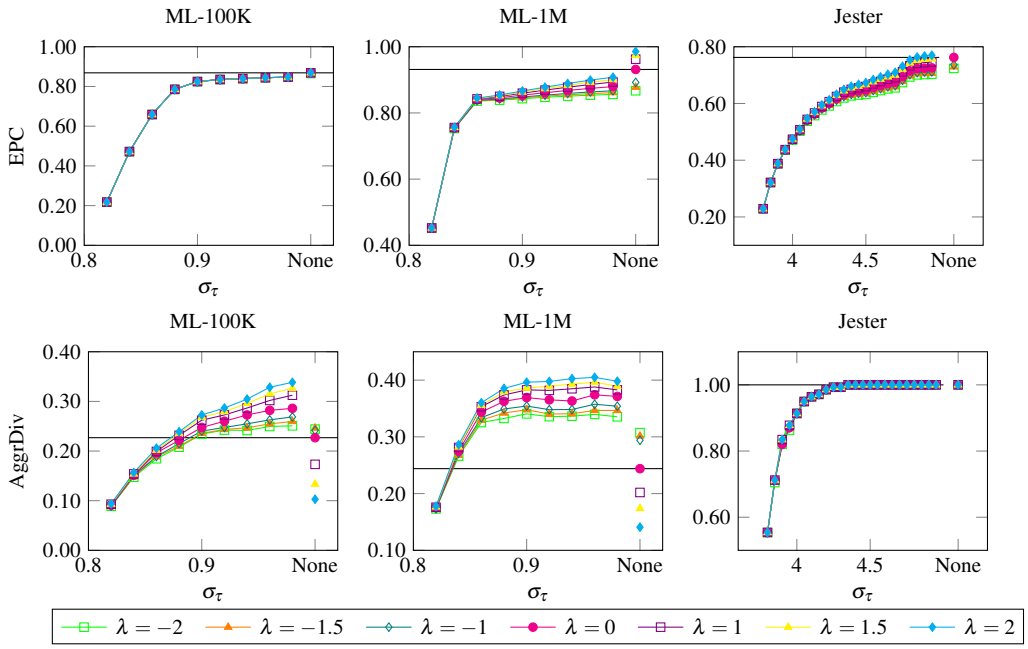

Fig. 7 Novelty and diversity metrics when using an strategy based on **confidence intervals** (points over the *None* label), and a combination of this strategy and the **prediction uncertainty** strategy for a Variational Bayesian algorithm. Baseline value (no strategy) is presented as the horizontal line.

the prediction uncertainty directly to produce recommendations, instead of to decide if an item should be included or not in a recommendation list.

We have shown that a balance between these evaluation dimensions – especially between precision and coverage – is critical, and different metrics have been studied to draw conclusions from them. As a first step towards improving the understanding of this tradeoff, we have proposed a family of metrics (correctness) based on the assumption that it is better to avoid a recommendation rather than providing a bad recommendation. These metrics take into account the amount of unanswered items returned by a recommender system to produce its score. One advantage of the proposed correctness metrics is that they do not need additional parameters – hence, they are not sensitive to possible biases towards one of the combined metrics like other combination measures analysed – e.g., F-score. Furthermore, they allow to consider the relevant items received by the user, the user coverage, and the item coverage, while, at the same time, accounting for incomplete rankings by penalising more those recommenders that retrieve less relevant items. However, further analysis is needed in the future, especially to find an objective way to discriminate between these systems and decide which of these metrics correlates better with the user satisfaction.

The results so far evidence that it is possible to improve precision while keeping moderate reductions on coverage, a tradeoff successfully captured by the proposed correctness evaluation metrics. More specifically, the experiments show that improvements up to 250% can be achieved using the prediction uncertainty with a probabilistic matrix factorisation algorithm; however, when this technique is used with a nearest-neighbour algorithm, the improvements are much smaller, probably because the computation of the prediction uncertainty is not based on a closed formulation but it is calculated empirically. In contrast,

the nearest-neighbour algorithm achieves large performance improvements when using the prediction support technique. Nonetheless, the algorithm independent technique (based on the final predicted score) does not produce positive results with the tested algorithms.

In the future, we aim at extending the correctness family of metrics so that other evaluation dimensions could be combined under the same framework: diversity, novelty, or even other accuracy metrics like normalised discounted cumulative gain. A possible line of work would be on applying multi-objective optimisation frameworks with this new set of metrics but considering also the number of unanswered items in their computation. We also want to integrate implicit feedback in the decision-aware recommendation strategies presented herein, since at the moment the three techniques proposed assume the algorithm is predicting a rating for every user and item pair. Additionally, the psychological aspect of the recommendations should also be considered, since if a user expects to receive N recommendations, she may decrease her confidence on the system if less than N recommendations are presented, although such behaviour should be linked to a target user task, i.e., *find good items* vs *just browsing* (Herlocker et al 2004). Moreover, we aim at validating these results in an online setting with real users, in particular, how users value incorrect recommendations in comparison with unanswered/missing recommendations.

Acknowledgements This work was funded by the national Spanish Government under project TIN2016-80630-P. The authors also acknowledge the very helpful feedback from the three anonymous reviewers.

References

Adomavicius G, Kamireddy S, Kwon Y (2007) Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance, Social Science Research Network, pp 152–157

Baeza-Yates RA, Ribeiro-Neto BA (2011) Modern Information Retrieval - the concepts and technology behind search, Second edition. Pearson Education Ltd., Harlow, England

Bell RM, Koren Y (2007) Lessons from the netflix prize challenge. SIGKDD Explorations 9(2):75–79

Bellogin A, Castells P, Cantador I (2011) Precision-oriented evaluation of recommender systems: an algorithmic comparison. In: Proceedings of the fifth ACM conference on Recommender systems, ACM, pp 333–336

Bishop CM (2006) Pattern recognition and machine learning. springer

Box GE, Tiao GC (2011) Bayesian inference in statistical analysis, vol 40. John Wiley & Sons

Castells P, Hurley NJ, Vargas S (2015) Novelty and diversity in recommender systems. In: Recommender Systems Handbook, Springer, pp 881–918

Cremonesi P, Koren Y, Turrin R (2010) Performance of recommender algorithms on top-n recommendation tasks. In: Amatriain X, Torrens M, Resnick P, Zanker M (eds) Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010, ACM, pp 39–46

Ekstrand MD, Riedl JT, Konstan JA (2011) Collaborative filtering recommender systems. Foundations and Trends in Human-Computer Interaction 4(2):81–173

Gunawardana A, Shani G (2015) Evaluating recommender systems. In: Recommender Systems Handbook, Springer, pp 265–308

Herlocker JL, Konstan JA, Riedl J (2002) An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf Retr 5(4):287–310

Herlocker JL, Konstan JA, Terveen LG, Riedl J (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53

Himabindu TVR, Padmanabhan V, Pujari AK, Sattar A (2016) Prediction with confidence in item based collaborative filtering. In: Booth R, Zhang M (eds) PRICAI 2016: Trends in Artificial Intelligence - 14th Pacific Rim International Conference on Artificial Intelligence, Phuket, Thailand, August 22-26, 2016, Proceedings, Springer, Lecture Notes in Computer Science, vol 9810, pp 125–138, DOI 10.1007/978-3-319-42911-3_11, URL `https://doi.org/10.1007/978-3-319-42911-3_11`

Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: ICDM, IEEE Computer Society, pp 263–272

Hull DA (1993) Using statistical testing in the evaluation of retrieval experiments. In: SIGIR, pp 329–338

Jannach D, Adomavicius G (2016) Recommendations with a purpose. In: Sen S, Geyer W, Freyne J, Castells P (eds) Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016, ACM, pp 7–10, DOI 10.1145/2959100.2959186, URL http://doi.acm.org/10.1145/2959100.2959186

Jugovac M, Jannach D, Lerche L (2017) Efficient optimization of multiple recommendation quality factors according to individual user tendencies. Expert Syst Appl 81:321–331, DOI 10.1016/j.eswa.2017.03.055, URL https://doi.org/10.1016/j.eswa.2017.03.055

Karypis G (2001) Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the tenth international conference on Information and knowledge management, ACM, pp 247–254

Koren Y, Bell R, Volinsky C, et al (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37

Lacerda A (2017) Multi-objective ranked bandits for recommender systems. Neurocomputing 246:12–24, DOI 10.1016/j.neucom.2016.12.076, URL https://doi.org/10.1016/j.neucom.2016.12.076

Latha R, Nadarajan R (2015) Ranking based approach for noise handling in recommender systems. In: Dziech A, Leszczuk M, Baran R (eds) Multimedia Communications, Services and Security, Springer International Publishing, Cham, pp 46–58

Lim YJ, Teh YW (2007) Variational bayesian approach to movie rating prediction. In: Proceedings of KDD cup and workshop, Citeseer, vol 7, pp 15–21

Linden G, Smith B, York J (2003) Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing 7(1):76–80

Mazurowski MA (2013) Estimating confidence of individual rating predictions in collaborative filtering recommender systems. Expert Syst Appl 40(10):3847–3857, DOI 10.1016/j.eswa.2012.12.102, URL https://doi.org/10.1016/j.eswa.2012.12.102

McNee SM, Lam SK, Guetzlaff C, Konstan JA, Riedl J (2003) Confidence displays and training in recommender systems. In: INTERACT, IOS Press

McNee SM, Riedl J, Konstan JA (2006) Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI, ACM, pp 1097–1101

Nakajima S, Sugiyama M (2011) Theoretical analysis of bayesian matrix factorization. Journal of Machine Learning Research 12(Sep):2583–2648

Ning X, Desrosiers C, Karypis G (2015) A comprehensive survey of neighborhood-based recommendation methods. In: Recommender Systems Handbook, Springer, pp 37–76

O'Donovan J, Smyth B (2005) Trust in recommender systems. In: IUI, ACM, pp 167–174

Peñas A, Rodrigo Á (2011) A simple measure to assess non-response. In: Lin D, Matsumoto Y, Mihalcea R (eds) The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA, The Association for Computer Linguistics, pp 1415–1424

Salakhutdinov R, Mnih A (2011) Probabilistic matrix factorization. In: NIPS, vol 20, pp 1–8

Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, ACM, pp 285–295

Smyth B, Wilson DC, O'Sullivan D (2002) Data mining support for case-based collaborative recommendation. In: AICS, Springer, Lecture Notes in Computer Science, vol 2464, pp 111–118

Srebro N, Jaakkola T, et al (2003) Weighted low-rank approximations. In: Icml, vol 3, pp 720–727

Toledo RY, Mota YC, Martínez-López L (2015) Correcting noisy ratings in collaborative recommender systems. Knowl-Based Syst 76:96–108, DOI 10.1016/j.knosys.2014.12.011, URL https://doi.org/10.1016/j.knosys.2014.12.011

Vargas S, Castells P (2011) Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the fifth ACM conference on Recommender systems, ACM, pp 109–116

Wang S, Gong M, Li H, Yang J (2016) Multi-objective optimization for long tail recommendation. Knowl-Based Syst 104:145–155, DOI 10.1016/j.knosys.2016.04.018, URL https://doi.org/10.1016/j.knosys.2016.04.018

Zhang M, Guo X, Chen G, Wei Q (2014) Predicting consumer information search benefits for personalized online product ranking: a confidence-based approach. In: Siau K, Li Q, Guo X (eds) 18th Pacific Asia Conference on Information Systems, PACIS 2014, Chengdu, China, June 24-28, 2014, p 375, URL http://aisel.aisnet.org/pacis2014/375

Zhang M, Guo X, Chen G (2016) Prediction uncertainty in collaborative filtering: Enhancing personalized online product ranking. Decision Support Systems 83:10–21, DOI 10.1016/j.dss.2015.12.004, URL https://doi.org/10.1016/j.dss.2015.12.004

A Complete Derivation of Variational Bayesian

The goal of this algorithm is to minimise the objective function in Equation 6 using a probabilistic model where R are the observations, U and I are the parameters, and some regularisation is introduced through priors on these parameters. As before, user u is represented by the u-th row of matrix U and item i is represented by the i-th column of matrix I; nonetheless, from now on, we shall consider these column vectors as row vectors.

The first hypothesis of the model is that the rating probability, considering matrices U and I, follows a Normal distribution with mean $u^\top i$ and variance τ^2:

$$P(r_{ui} \mid U, I) = N(u^\top \cdot i, \tau^2) \tag{A.1}$$

Moreover, the entries of these matrices U and I follow independent distributions; in particular, $u_l \sim N(0, \sigma_l^2)$ and $i_l \sim N(0, \rho_l^2)$. As a consequence, the density functions of U and I are formulated as in Equation A.2:

$$P(U) = \prod_{u=1}^{V} \prod_{l=1}^{D} N(u_l \mid 0, \sigma_l^2) \qquad P(I) = \prod_{i=1}^{J} \prod_{l=1}^{D} N(i_l \mid 0, \rho_l^2) \tag{A.2}$$

The goal now is to compute or approximate the posterior probability $P(U, I \mid R)$, so, in this way, we can compute a predictive distribution on ratings given the observation matrix, as in Equation A.3:

$$P(\hat{r}_{ui} \mid R) = \int P(\hat{r}_{ui} \mid u, i, \tau^2) P(U, I \mid R) \tag{A.3}$$

Since computing this posterior probability $P(U, I \mid R)$ is unfeasible, the proposed algorithm aims at approximating such distribution by using Bayesian inference, more specifically, through the method known as *Mean-Field Variational Inference*.

Bayesian inference

Bayesian inference is a method that approximates a posterior distribution of a set of unobserved variables $Z = \{Z_1, \cdots, Z_D\}$ knowing a subset of information R, $P(Z \mid R)$, through a variational distribution $Q(Z)$ (Box and Tiao 2011). For this, a function $Q(Z)$ is sought that minimises the dissimilarity function $d(Q, P)$. In the specific case of the *Mean-Field Variational Inference* method, such a dissimilarity function is the Kullback-Leibler divergence between P and Q (Bishop 2006):

$$D_{KL}(Q \,||\, P) = \sum_Z Q(Z) \log \frac{Q(Z)}{P(Z \mid R)} = \sum_Z Q(Z) \log \frac{Q(Z)}{P(Z, R)} + \log P(R) \tag{A.4}$$

where we use that $P(Z \mid R) = P(Z, R)/P(R)$. Furthermore, by solving Equation A.4 for $\log P(R)$ we obtain the following Equation A.5:

$$\log P(R) = D_{KL}(Q \,||\, P) - \sum_Z Q(Z) \log \frac{Q(Z)}{P(Z, R)} = D_{KL}(Q \,||\, P) + \mathscr{F}(Q) \tag{A.5}$$

If we set $\log P(R)$ as a constant in Equation A.5, it is enough with maximising $\mathscr{F}(Q)$ to minimise $D_{KL}(Q \,||\, P)$. Indeed, $\mathscr{F}(Q)$ is denoted as the *(negative) variational free energy*, since it can be written as a sum of Q's entropy and an energy (Equation A.6):

$$\begin{aligned}\mathscr{F}(Q) &= -\sum_Z Q(Z)\log Q(Z) + \sum_Z Q(Z)\log P(Z,R) \\ &= H(Q) + \mathbb{E}[\log P(Z,R)] = \mathbb{E}_{Q(z)}[\log P(Z,R) - \log Q(Z)]\end{aligned} \tag{A.6}$$

Bayesian inference in VB

Once we apply the *Mean-Field Variational Inference* method presented above where $Z = \{U,I\}$ are the unobserved variables, and the rating matrix R is the known information, the goal is to estimate a variational distribution function $Q(U,I)$ that approximates the distribution $P(U,I \mid R)$. For this, we aim at maximising the so-called *(negative) variational free energy* defined as in Equation A.7:

$$\begin{aligned}\mathscr{F}(Q(U,I)) &= \mathbb{E}_{Q(U,I)}[\log P(U,I,R) - \log Q(U,I)] \\ &= -\sum Q(U,I)(\log Q(U,I) - \log P(U,I,R))\end{aligned} \tag{A.7}$$

In practice, it is intractable to maximise $F(U,I)$ and, because of this, this algorithm applies another simplification by considering that $Q(U,I) = Q(U)Q(I)$. We now arrive to the definitive formulation for $\mathscr{F}(Q(U)Q(I))$, Equation A.8, as it is defined in (Lim and Teh 2007):

$$\begin{aligned}\mathscr{F}(Q(U)Q(I)) &= \mathbb{E}_{Q(U)Q(I)}[\log P(U,I,R) - \log Q(U,I)] \\ &= \mathbb{E}_{Q(U)Q(I)}\left[\log \frac{P(R \mid U,I)}{P(U,I)}\right] + H\left(Q(U,I)\right) \\ &= \mathbb{E}_{Q(U)Q(I)}\left[\log P(R \mid U,I) - \log P(U) - \log P(I)\right] + H(Q(U,I)) \\ &= -\frac{K}{2}\log(2\pi\tau^2) - \frac{1}{2}\sum_{(u,i)}\frac{\mathbb{E}_{Q(U)Q(I)}[(r_{ui} - u^\top i)^2]}{\tau^2} \\ &- \frac{V}{2}\sum_{l=1}^{D}\log(2\pi\sigma_l^2) - \frac{1}{2}\sum_{l=1}^{D}\frac{\sum_{u=1}^{V}\mathbb{E}_{Q(U)}[u_l^2]}{\sigma_l^2} \\ &- \frac{J}{2}\sum_{l=1}^{D}\log(2\pi\rho_l^2) - \frac{1}{2}\sum_{l=1}^{D}\frac{\sum_{i=1}^{J}\mathbb{E}_{Q(I)}[i_l^2]}{\rho_l^2} \\ &+ H(Q(U,I))\end{aligned} \tag{A.8}$$

where K is the number of observed entries in R, or, in other terms, the number of ratings or interactions known by the system at training time.

Maximising $\mathscr{F}(Q(U)Q(I))$ with respect to $Q(U)$ keeping $Q(I)$ fixed, and viceversa, we obtain the distributions for items and users. Equations A.9 and A.10 describe the covariance matrix ϕ_u and user's u mean $\overline{u}$ in $Q(U)$, respectively.

$$\phi_u = \left(\begin{pmatrix}\frac{1}{\sigma_1^2} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sigma_D^2}\end{pmatrix} + \sum_{i \in O(u)}\frac{\psi_i + \bar{i}^\top \bar{i}}{\tau^2}\right)^{-1} \tag{A.9}$$

$$\overline{u} = \phi_u\left(\sum_{i \in O(u)}\frac{r_{ui}\bar{i}}{\tau^2}\right) \tag{A.10}$$

where $O(u)$ are the items observed by user u, which corresponds to the identifiers i such that r_{ui} was observed in the rating matrix R. Similarly, Equations A.11 and A.12 show the formulation for the covariance matrix ψ_i and item's i mean $\bar{i}$ in $Q(I)$, respectively.

$$\psi_i = \left(\begin{pmatrix} \frac{1}{\rho_1^2} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\rho_D^2} \end{pmatrix} + \sum_{u \in O(i)} \frac{\phi_u + \overline{u}^\top \overline{u}}{\tau^2} \right)^{-1} \tag{A.11}$$

$$\bar{i} = \psi_i \left(\sum_{u \in O(i)} \frac{r_{ui} \overline{u}}{\tau^2} \right) \tag{A.12}$$

where $O(i)$ denotes the set of users that have rated item i.

The variances σ_l, ρ_l, and τ can also be estimated and learned by the model. By differentiating Equation A.8 with respect to σ_l, ρ_l, and τ, setting the derivatives to zero, and solving for the optimal parameters, we obtain Equations A.13, A.14, and A.15.

$$\sigma_l^2 = \frac{1}{V-1} \sum_{u=1}^{V} (\phi_u)_{ll} + \overline{u_l}^2 \tag{A.13}$$

$$\rho_l^2 = \frac{1}{J-1} \sum_{i=1}^{J} (\psi_i)_{ll} + \overline{v_l}^2 \tag{A.14}$$

$$\tau^2 = \frac{1}{K-1} \sum_{(u,i)} r_{ui}^2 - 2 r_{ui} \overline{u}^\top \bar{i} + trace[(\phi_u + \overline{u}^\top \overline{u})(\psi_i + \bar{i}^\top \bar{i})] \tag{A.15}$$

Rating prediction in VB

Once the optimal approximation for $P(U,I \mid R)$ is calculated, that is, $Q(U,I)$, it can be used to predict future interactions between users and the system. For this, given a matrix R, we find that the distribution of a new rating is the following (Equation A.16):

$$P(\hat{r}_{ui} \mid R) \sim \int P(r_{ui} \mid u^\top i, \tau) Q(U,I) dU dI = \int N(\hat{r}_{ui} \mid u^\top i, \tau^2) Q(U,I) dU dI \tag{A.16}$$

Therefore, we can obtain its mean and, hence, the estimated or predicted rating:

$$\mathbb{E}(\hat{r}_{ui} \mid R) = \int \int \hat{r}_{ui} N(\hat{r}_{ui} \mid u^\top i, \tau^2) Q(U,I) dU dI d\hat{r}_{ui} \tag{A.17}$$

Finally, when we apply Fubini's theorem, we obtain the following formulation:

$$\mathbb{E}(\hat{r}_{ui} \mid R) = \int \underbrace{\int \hat{r}_{ui} N(\hat{r}_{ui} \mid u^\top i, \tau^2) d\hat{r}_{ui}}_{u^\top i} Q(U,I) dU dI = \mathbb{E}(u^\top i) = \overline{u}^\top \bar{i} \tag{A.18}$$

Standard deviation of predicted rating in VB

We use the following explicit formulation for the standard deviation, derived by using mean-field variational inference. First, we use the general form to compute the standard deviation when estimating rating $\hat{r}_{ui}$ for user u and item i:

$$Var(\hat{r}_{ui} \mid R) = \mathbb{E}(\hat{r}_{ui}^2 \mid R) - \mathbb{E}(\hat{r}_{ui} \mid R)^2 \tag{A.19}$$

Considering the formulation presented in Equation A.18, we know that $\mathbb{E}(\hat{r}_{ui} \mid R)^2 = (\overline{u}^\top \overline{i})^2 = \overline{i}^\top \overline{uu}^\top \overline{i}$. Hence:

$$\begin{aligned}
\mathbb{E}(\hat{r}_{ui}^2 \mid R) &= \int\int \hat{r}_{ui}^2 N(\hat{r}_{ui} \mid u^\top i, \tau^2) Q(U,I) dU dI d\hat{r}_{ui} \\
&= \int \underbrace{\int \hat{r}_{ui}^2 N(\hat{r}_{ui} \mid u^\top i, \tau^2) d\hat{r}_{ui}}_{\mathbb{E}(r_{ui}^2) = \tau^2 - \mathbb{E}(r_{ui})^2 = \tau^2 + i^\top uu^\top i} Q(U,I) dU dV \\
&= \mathbb{E}(\tau^2 + i^\top uu^\top i) \\
&= \tau^2 + \mathbb{E}(i^\top uu^\top i) \\
&= \tau^2 + trace((\phi_u - \overline{uu}^\top)(\psi_i - \overline{ii}^\top))
\end{aligned} \tag{A.20}$$

And therefore:

$$Var(\hat{r}_{ui}) = Var(\hat{r}_{ui} \mid R) = \tau^2 + \text{trace}\left(\left(\phi_u + \overline{uu}^\top\right)\left(\psi_i + \overline{ii}^\top\right)\right) - \overline{i}^\top \overline{uu}^\top \overline{i} \tag{A.21}$$

which gives us an explicit estimation of the deviation (uncertainty) on the predicted rating $\hat{r}_{ui}$.