

Title	Estimating and evaluating the uncertainty of rating predictions and top-n recommendations in recommender systems
Authors	Coscrato, Victor;Bridge, Derek
Publication date	2023-02-16
Original Citation	Coscrato, V. and Bridge, D. (2023) 'Estimating and evaluating the uncertainty of rating predictions and top-n recommendations in recommender systems', ACM Transactions on Recommender Systems. doi: 10.1145/3584021
Type of publication	Article (peer-reviewed)
Link to publisher's version	10.1145/3584021
Rights	© 2023, the Authors. Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in ACM Transactions on Recommender Systems, <a href="https://doi.org/10.1145/3584021">https://doi.org/10.1145/3584021</a>
Download date	2024-05-21 06:39:14
Item downloaded from	<a href="https://hdl.handle.net/10468/14296">https://hdl.handle.net/10468/14296</a>



# Estimating and Evaluating the Uncertainty of Rating Predictions and Top-n Recommendations in Recommender Systems

VICTOR COSCRATO and DEREK BRIDGE, University College Cork, Ireland

Uncertainty is a characteristic of every data-driven application, including recommender systems. The quantification of uncertainty can be key to increasing user trust in recommendations or choosing which recommendations should be accompanied by an explanation; and uncertainty estimates can be used to accomplish recommender tasks such as active learning and co-training. Many uncertainty estimators are available but, to date, the literature has lacked a comprehensive survey and a detailed comparison. In this paper, we fulfil these needs. We review the existing methods for uncertainty estimation and metrics for evaluating uncertainty estimates, while also proposing some estimation methods and evaluation metrics of our own. Using two datasets, we compare the methods using the evaluation metrics that we describe, and we discuss their strengths and potential issues. The goal of this work is to provide a foundation to the field of uncertainty estimation in recommender systems, on which further research can be built.

CCS Concepts: • **Information systems** → **Recommender systems**; **Uncertainty**; • **Computing methodologies** → **Uncertainty quantification**; • **General and reference** → *Empirical studies*; *Evaluation*; *Estimation*; • **Mathematics of computing** → *Probability and statistics*.

Additional Key Words and Phrases: uncertainty, recommender systems

## 1 INTRODUCTION

Uncertainty is common to every machine learning (ML) task [19, 22]. In particular, model predictions carry a degree of uncertainty. This uncertainty is often neglected, but it can be beneficial if uncertainty is quantified and even exploited. The quantification of prediction uncertainty has different motivations in different domains. For example, in medical decision making, estimating the uncertainty of predictions is key to the management of risk [4]. In weather forecasting, estimating uncertainty and presenting the estimates to users can increase credibility [39].

The literature in the area often categorizes uncertainty into two types: *aleatoric* and *epistemic* [22]. The first refers to the randomness of the data itself, which can not be reduced by additional knowledge. The second type, which is the focus of this paper, instead refers to the uncertainty caused by lack of knowledge. Therefore, epistemic uncertainty reduces our ability to know which of several prediction models is the correct one, which ultimately leads to uncertainty around the chosen model's predictions. For this reason, we define uncertainty as the expected imprecision of a prediction. Some related work in the area uses terminology such as reliability [9] or confidence [13, 41], which, in the context of our work, we take to be the opposite of uncertainty, that is, the expected precision of a prediction.

We study uncertainty in Recommender Systems. A Recommender System (RS) exposes items (such as products, services, news articles or even people) to its users [36]. RSs help their users to discover items that they might not

---

Authors' address: Victor Coscrato, victor.coscrato@cs.ucc.ie; Derek Bridge, d.bridge@cs.ucc.ie, University College Cork, College Road, Cork, Ireland.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

2770-6699/2023/2-ART \$15.00

<https://doi.org/10.1145/3584021>

have found for themselves, and they help users manage the choice of which items to consume. RSs can be found in many different application domains, often employed at scale: for example, we find them in online shopping sites, in music and movie streaming services, in social media platforms, and in news story aggregators.

An RS may filter and rank candidate items in order to recommend a list of the top- $n$  candidate items. Often, the filtering and ranking are *personalized*: the top- $n$  on any occasion depends on the user and the context in which the items are to be consumed. An RS will often use a model that has been learned from knowledge about the users, knowledge about the items, and records of interactions between users and items (such as purchases, downloads, clicks and ratings). While there are many different types of models, content-based and collaborative-filtering (CF) recommenders are two major categories. The first uses descriptions of the items in the catalog allied with records of the active user's item consumption history. On the other hand, CF recommenders learn from historical user-item interactions; in this case, the recommendations to a user are affected by the preferences of other users in the system.

There can be high uncertainty in the recommendations that an RS makes to its users. In practical RS applications, the number of items is very large, meaning that users will have interacted with only a tiny fraction of them. This is known as *sparsity*. It means the system must infer a user's preferences from a relatively small amount of information, leading sometimes to the generation of unsuccessful but also uncertain recommendations. Moreover, there are challenges in collecting reliable user-item interaction data. This is particularly true in the case of explicit ratings, where a user supplies an opinion using, e.g., a 5-point rating scale. Ratings data may be unreliable due to its high degree of subjectivity [42], which can be explained by variations in user behavior, personality and mood; change of preferences over time; and also by different interpretations of the rating scales by the users [2, 8]. Furthermore, ratings are usually collected from open systems, being subject to natural noise (e.g. typing errors) and even malicious noise (e.g. from shilling attacks) [32].

Quantifying the uncertainty of an RS's predictions and recommendations is important [13, 28, 29]. Ideally, an uncertainty estimate should help detect which predictions and recommendations are more or less likely to be wrong [9, 44]. Estimates of uncertainty can improve the operation of the RS in at least the following three ways.

First, estimates of uncertainty can enhance the presentation of recommendations to users. This may make the RS more credible, increasing the user's trust and satisfaction. In the simplest case, we can present the degree of uncertainty alongside each recommendation [13, 29]. But there are other ways that uncertainty estimates could affect the presentation of a set of recommendations. For example, consider an RS that can offer explanations for its recommendations. The user is most likely to need explanations for recommendations where it is least clear why the item is being recommended. These are likely to be the recommendations about which the RS is least certain. The RS could use high uncertainty to trigger the offer of an explanation. This may avert the loss of credibility that can come from showing recommendations that do not make sense to the user.

Second, uncertainty measures offer new strategies for selecting the top- $n$  [28, 33]. Conventionally, the recommended items are those that the model predicts to have highest relevance to the user. But, equipped with estimates of uncertainty, an RS might discard highly uncertain recommendations, even when their predicted relevance is high: the user is shown items that the RS is certain are relevant. On the other hand, a RS might deliberately include some items that it predicts are relevant but where the relevance is uncertain: from a user point-of-view, this coincides with the idea that RSs are tools for item discovery; from a system point-of-view, this helps the RS explore user tastes by seeking feedback on uncertain recommendations. An RS might even combine these strategies: playing-it-safe by selecting some recommendations that are certain even though they may be obvious to the user; but taking-a-risk by also recommending some of the less certain ones. This is reminiscent of the exploration/exploitation trade-off that is widely recognized and studied in Reinforcement Learning and even in the RS literature, e.g. [3, 10].

Third, these estimates may be useful internally to an RS. A concrete example is found in the CoRec system, which uses co-training [15]. CoRec comprises more than one model and the most certain predictions of one

model are added to the training set of the other model. More generally, uncertainty measures might help a hybrid RS decide which of several models to use in a particular circumstance [11]. There are also examples of Active Learning in Recommender System in which users are prompted to rate items about which the RS is uncertain, e.g. [25].

In this paper, we examine several ways of estimating uncertainty in RSs. In terms of scope, we confine our attention to RSs where user-item interactions take the form of explicit, numeric ratings, such as opinions on a 5-point scale. We review, extend and evaluate the work done in this setting. Of course, the case where interactions are implicit (purchases, downloads, clicks, and so on) is important but it has been largely unexplored from the point-of-view of uncertainty quantification. We expect this paper, with its focus on explicit ratings, to be a solid foundation that we and others can build on when progressing to future work in the implicit feedback setting.

The contributions of this paper are:

- We present an extensive review of methods for estimating the uncertainty of rating predictions. These methods include information-based, stability-based, error-based, distribution-based and multinomial-based.
- We propose new techniques for estimating the uncertainty of rating predictions. In particular, we propose a new stability-based method, and also a new error-based method.
- We present an extensive review of the existing work on evaluating estimates of uncertainty for rating prediction and for top- $n$  recommendation.
- We extend and improve the techniques and metrics for evaluating uncertainty estimates in RS. In particular, we analyse how uncertainty correlates with dataset statistics, we fix possible issues with existing metrics, and we propose a new metric to evaluate uncertainty in the top- $n$  recommendation task.
- We introduce the idea of uncertainty-aware ranking, drawing in part on a recommendation strategy proposed in [33]. In particular, we distinguish uncertainty-based filtering (UBF) and probability-of-relevance ranking (PRR). We give methods for evaluating these recommendation strategies.
- We conduct a reproducible empirical study on two large datasets and report results of the most extensive comparison of approaches to uncertainty estimation that we are aware of.

In the next section, we will introduce the basic concepts of rating prediction and top- $n$  recommendation and the role of uncertainty in each. We will also set-up the notation used for the remainder of the paper. In Section 3, we will describe several methods for estimating uncertainty in RS. After that, Section 4 will present metrics for uncertainty evaluation. The remainder of the paper evaluates the uncertainty estimates from Section 3 using the evaluation metrics from Section 4. In particular, Section 5 describes the experiments, while Section 6 shows the results. Section 7 discusses the uncertainty estimation methods in the light of the experiments. Section 8 concludes the paper.

## 2 PREDICTION AND RECOMMENDATION

As already mentioned, in this work, we deal with explicit feedback data — specifically, numerical ratings. We will designate the ordered set of possible ratings by  $S = \{s_1, \dots, s_m\}$ , where  $m$  is the number of different possible rating values. For example, in the MovieLens dataset and the Netflix dataset that we use in Section 5,  $S = \{1, 1.5, \dots, 5\}$  and  $S = \{1, 2, \dots, 5\}$  respectively.

We will assume that our data consists of a set of users  $U$ , a set of items  $I$ , and a set  $R$  of user-item-rating triples, where triple  $\langle u, i, r_{ui} \rangle \in R$  means that user  $u$ 's rating of item  $i$  is  $r_{ui} \in S$ . A user's profile  $R_u$  is the set of that user's ratings,  $R_u = \{r_{ui} : \langle u, i, r_{ui} \rangle \in R\}$ . Similarly, an item's support set  $R_i$  is the set of that item's ratings,  $R_i = \{r_{ui} : \langle u, i, r_{ui} \rangle \in R\}$ .

## 2.1 Rating prediction

Given a user  $u \in U$  and item  $i \in I$ , the *rating prediction* task consists in designing or learning a function  $\Phi : U \times I \rightarrow \mathbb{R}$  that predicts user-item ratings. We will write  $\hat{r}_{ui}$  for a particular prediction, i.e.  $\hat{r}_{ui} = \Phi(u, i)$ .

In this work, we want, not only to predict ratings, but also the *uncertainty level*,  $\rho_{ui}$ , present in a prediction. We will refer to a pair  $\langle \hat{r}_{ui}, \rho_{ui} \rangle$  as an *uncertain prediction*. In all the work that we report in this paper, uncertainty levels are real-valued,  $\rho_{ui} \in \mathbb{R}$ . In some cases, they are probabilistic, which bounds them in  $[0, 1]$ , but this is not always the case. The higher the uncertainty level, the more uncertain the prediction.

Many of the methods for estimating uncertainty that we present in this paper assume a rating prediction model,  $\Phi$ . Wherever this is the case, we will employ Matrix Factorization (MF) for this underlying model. Specifically, we will use the form of MF known as FunkSVD [18]. This is known to be a fairly accurate predictor and, by using it in all cases where we need a separate predictor, we bring a degree of fairness to the comparisons. The idea is to represent users and items in a low dimension latent space. If the number of latent factors is denoted by  $d$ , then each user  $u$  is represented by a vector  $p_u$  and each item  $i$  by a vector  $q_i$ , both of dimension  $d$ . Then, we can compute a predicted rating  $\hat{r}_{ui}$  by taking a dot product:

$$\hat{r}_{ui} = p_u^T q_i \quad (1)$$

One way to learn the user and item embeddings,  $p_u$  and  $q_i$ , is through gradient descent: the vectors are initialised with randomly-chosen values; the algorithm proceeds by sampling instances from the training set, calculating the error between the predicted rating (Eq. 1) and the actual rating for these instances; and updating  $p_u$  and  $q_i$  through gradient descent to minimize these errors. In practice, the loss function to be minimized for each sampled instance is:

$$(r_{ui} - \hat{r}_{ui})^2 + \lambda_U \|p_u\|^2 + \lambda_I \|q_i\|^2 \quad (2)$$

where  $\lambda_U$  and  $\lambda_I$  are regularization hyperparameters and  $\|\cdot\|$  denotes the Frobenius norm. To speed up the training process, batches of instances can be sampled instead. In this case, the parameters are updated according to the average loss of the batch.

## 2.2 Top- $n$ recommendation

Of more importance than rating prediction is *top- $n$  recommendation* [21]. Given a user  $u \in U$  and a set of candidate items  $C_u \subseteq I$ , the top- $n$  recommendation task consists of retrieving  $Z_u^n \subseteq C_u$ , which is the set of  $n$  items from  $C_u$  that are predicted to be most appealing to  $u$ . There are several ways that this can be done.

The conventional way of selecting the top- $n$  is to use the predicted ratings alone (see *rating-based ranking*, below). But, given that we are considering models that can estimate prediction uncertainty, then there is an opportunity to consider other ways of selecting the top- $n$ . These use what we can collectively refer to as *uncertainty-aware ranking*. If the estimates of uncertainty are good enough, we may achieve better (or, at least, usefully different) recommendation performance by recommendation strategies that rank the candidates using some combination of the predicted ratings and their estimated uncertainties. We propose two such strategies: *uncertainty-based filtering* and *probability-of-relevance ranking*.

**2.2.1 Rating-based ranking.** When feedback data takes the form of explicit, numeric ratings, the top- $n$  recommendation task is conventionally performed by ranking the candidate items  $C_u$  in descending order of their predicted ratings and then forming  $Z_u^n$  by selecting the top- $n$  items from this ordering. We will refer to this way of doing top- $n$  recommendation as *rating-based ranking* (RBR).

RBR makes no use of the uncertainty levels associated with the predictions. Next, we present two kinds of uncertainty-aware ranking.

**2.2.2 Uncertainty-based filtering.** In some domains, we may not want to recommend candidate items if we are uncertain that the user will find them to be appealing. In this case, we can use uncertainty-based filtering (UBF). In UBF, we filter the candidate items, discarding those whose uncertainty is above some threshold  $\tau$ . After this filtering step, we rank the remaining candidates in descending order of their predicted ratings and form  $Z_u^n$  by selecting the top- $n$  items from this ordering.

If  $\tau$  is small, it is likely that several of a user's candidate items will be filtered-out, which lead to less uncertain items being recommended. Some of these items may, however, be ones with smaller predicted ratings. Furthermore, some users might have a very small candidate set after filtering. For this reason, there might be cases in which the desired amount of recommendations cannot be provided to some users. On the other hand, larger  $\tau$  values may mean that, for some users, few or even no items are filtered-out.

There are, of course, variations of these ideas. For example, we could consider ranking the candidate items by a linear combination of their predicted ratings and their uncertainty estimates.

**2.2.3 Probability-of-relevance ranking.** Both RBR and UBF rank the candidates item based on their predicted ratings. But, if predictions are accompanied by uncertainty levels, there may, in certain cases, be an opportunity to rank candidates based on uncertainty. We call this probability-of-relevance ranking (PRR) and we will present the exact details in later sections. But, in overview, the approach requires the use of a relevance threshold,  $\theta \in S$ . When a rating is greater than, or equal to, this threshold, then the item is deemed to be relevant to the user. In the datasets we use in Section 5, which use a 5-star rating scale,  $\theta$  might be 4, for example. In PRR, we must estimate the probability that  $r_{ui} \geq \theta$ ,  $\mathbb{P}(r_{ui} \geq \theta)$  – how certain we are that the item is relevant. Then, we rank the candidates in descending order of the relevance probabilities, so that those whose relevance is most certain are highest in the ranking. Finally, PRR recommends the top- $n$  from this ranking.

PRR was first proposed in [33], specifically for their BeMF method, which we will review in Section 3.5. Nevertheless, it also applies to many other models, as long as they are able to estimate  $\mathbb{P}(r_{ui} \geq \theta)$ . Some of the other methods that we review in Section 3 will be able to compute such probabilities by default, while we show that some others can also be extended to do so.

Notice that, with PRR,  $\mathbb{P}(r_{ui} \geq \theta)$  is used to rank the items. If one also wishes to associate each recommended item with a numeric uncertainty estimate (e.g. for display to the user), then the complementary probability  $1 - \mathbb{P}(r_{ui} \geq \theta)$  can be used.

### 2.3 Notation

Table 1 summarises the notation that we have introduced in this section. Other notation, which is specific to particular models, will be introduced on an as-needed basis in the remainder of the paper.

## 3 ESTIMATING UNCERTAINTY

There are many ways of estimating the level of uncertainty of a predicted rating in a Recommender System. Different sources of uncertainty can motivate different ways of estimating the uncertainty. For the purposes of surveying the approaches, we classify them into five broad types, as follows:

**Information-based:** Uncertainty is estimated from the amount of information that is known about users or items, or the dispersion of the ratings.

**Stability-based:** Uncertainty is estimated from the stability of the predictions across perturbations of the conditions under which we train the prediction model.

**Error-based:** Uncertainty is estimated from the expected prediction errors of an underlying rating prediction model.

**Distribution-based:** The ratings are assumed to follow a statistical distribution, whose dispersion is used to estimate uncertainty.

$U$	set of all users
$I$	set of all items
$r_{ui}$	user $u$ 's rating for item $i$
$S$	ordered set of $m$ possible rating values, $S = \{s_1, \dots, s_m\}$
$R$	set of all user-item-rating-triples
$R_{u\cdot}$	user $u$ 's profile, i.e. $R_{u\cdot} = \{r_{ui} : \langle u, i, r_{ui} \rangle \in R\}$
$R_{\cdot i}$	item $i$ 's support set, i.e. $R_{\cdot i} = \{r_{ui} : \langle u, i, r_{ui} \rangle \in R\}$
$\hat{r}_{ui}$	prediction of user $u$ 's rating of item $i$
$\Phi$	a prediction model, i.e. $\hat{r}_{ui} = \Phi(u, i)$
$\rho_{ui}$	estimated uncertainty of the prediction of user $u$ 's rating of item $i$
$\langle \hat{r}_{ui}, \rho_{ui} \rangle$	an uncertain prediction
$C_u$	candidate items, $C_u \subseteq I$ , that might be recommended to user $u$
$Z_u^n$	list of $n$ items that are recommended to user $u$
$\tau$	uncertainty threshold; in UBF, candidates for which $\rho_{ui} > \tau$ cannot be recommended to $u$
$\theta$	relevance threshold; item $i$ is considered relevant to user $u$ if $r_{ui} \geq \theta$

Table 1. General notation

**Multinomial-based:** Multinomial methods also estimate uncertainty from the dispersion of the predicted ratings but they use models that make discrete predictions for each rating value.

In the following sections, we describe each of these classes in more detail; we review work from the literature that falls into each class; and we even propose new stability-based and error-based methods.

### 3.1 Information-based uncertainty

The foremost source of uncertainty in a Recommender System comes from the lack of user-item interaction data. In practice, item catalogs are often large and therefore users will interact with only a small proportion of the items in the catalog. In our setting, this means that users will have rated only a small proportion of the items: in the majority of cases,  $\langle u, i, r_{ui} \rangle \notin R$ . Models learned from this sparse data may be unreliable. For this reason, simple statistics such as item support (the number of ratings an item has) can be used to measure the reliability of the predictions [28]. Hence, its negative can be used as an estimate of uncertainty:

$$\text{NEG-ITEM-SUPPORT: } \rho_{ui} = -\#R_{\cdot i}$$

Although item support is indicative of the information available about an item, it might also be misleading. For instance, two items might have the same number of ratings, but one of them might be rated similarly by different users, whereas the other might have received more divergent ratings. We expect a higher degree of uncertainty when predicting the ratings of items that have divergent ratings. For this reason, the variance of an item's ratings can also be used as an estimate of uncertainty [1, 28]:

$$\text{ITEM-VARIANCE: } \rho_{ui} = \text{Var}(R_{\cdot i})$$

In a similar vein, we can estimate uncertainty from user profile length (the number of ratings a user has) or the variance of a user's ratings:

$$\text{NEG-USER-SUPPORT: } \rho_{ui} = -\#R_{u\cdot}$$

$$\text{USER-VARIANCE: } \rho_{ui} = \text{Var}(R_{u\cdot})$$

Information-based measures of uncertainty can be insightful. Nevertheless, they are quite simple. One weakness is that they are model-independent, because they are computed strictly from the data, independently of the rating prediction model. In other words, changing from one prediction model to another does not result in any change

to the estimates of uncertainty. This is counter-intuitive, because different models utilise the data differently, and we would expect uncertainty estimates to reflect this behaviour. This will not be the case for information-based measures.

A second weakness is that NEG-ITEM-SUPPORT and ITEM-VARIANCE quantify uncertainty item-wise and not interaction-wise. That is, their estimated uncertainty for an item  $i$  is not personalised:  $\rho_{ui} = \rho_{vi} \forall (u, v) \in U^2$ . Similarly, NEG-USER-SUPPORT and USER-VARIANCE are strictly user-wise measures:  $\rho_{ui} = \rho_{uj} \forall (i, j) \in I^2$ .

Used in isolation, item-wise and user-wise measures lack the granularity that Recommender Systems typically need. This is especially the case for user-wise measures. A user-wise uncertainty estimate cannot help a system decide which of a set of recommendations need explanations, for example, since all of a user's recommendations will have the same uncertainty level. Similarly, user-wise uncertainty estimates cannot be used in top- $n$  recommendations that are uncertainty-aware, such as UBF and PRR, again because the estimated uncertainty associated with each of that user's predicted ratings will be the same. In fact, since top- $n$  recommendation is a more important task than rating prediction and since top- $n$  recommendation is a user-based task, user-wise uncertainty estimates are largely useless. For that reason, the literature on information-based uncertainty is focused on item-wise metrics [28]. In our experiments (Section 5), we do not include user-wise uncertainty estimates.

Nevertheless, user-wise and item-wise estimates of uncertainty can provide insights into Recommender System performance. For example, Bernardis et al. [7] show both theoretically and empirically how eigenvalues obtained from an item-item similarity matrix are strongly related to the model recommendation accuracy. The negatives of these eigenvalues can also be interpreted as user-wise uncertainty metrics.

Finally, we note that, instead of using these estimates in isolation, some authors have proposed heuristic combinations of NEG-USER-SUPPORT and NEG-ITEM-SUPPORT (e.g. based on their product) to give interaction-wise measures (e.g. [15, 43]).

### 3.2 Stability-based uncertainty

Prediction uncertainty is related to the level of arbitrariness present in the predictions. Hence, a prediction is uncertain the more it exhibits high variance across small perturbations to the conditions under which the prediction model is learned.

Mazurowski [28] explores this idea in a multi-modelling strategy. Consider that a model  $\Phi$  is used to predict ratings. Then, model stability can be measured by learning several different models from perturbations of the training dataset. A stochastic perturbation function,  $\mathcal{P}$ , is used to create  $N$  different versions of the original ratings training data, i.e.  $\mathcal{R}^{(k)} = \mathcal{P}(\mathcal{R})$ , for  $k \in 1, \dots, N$ . A ratings prediction model  $\Phi^{(k)}$  is learned from each of the versions of the dataset. Considering the predicted ratings as random variables, their standard deviation is an estimate of that rating's uncertainty.

In a strategy called RESAMPLE, Mazurowski [28] uses random sampling as the perturbation function. In this case, each individual model is trained using a random subset of the ratings available for training. The predicted rating  $\hat{r}_{ui}$  is given by a model  $\Phi$  trained on the entire training set; the estimated uncertainty of that prediction is given by the standard deviation of the predictions from the models  $\Phi^{(k)}$  that are trained on the different samplings of the training set:

$$\hat{r}_{ui} = \Phi(u, i) \quad (3)$$

$$\rho_{ui} = \frac{1}{N} \sqrt{\sum_{k=1}^N (\Phi^{(k)}(u, i) - \hat{r}_{ui})^2} \quad (4)$$



Mazurowski [28] proposes another such strategy, called INJECT. In this strategy, the perturbed datasets are obtained by adding noise to the original ratings, i.e. new ratings,  $r'_{ui}$ , are obtained as  $r_{ui} + \epsilon$ , where  $\epsilon$  is random Gaussian noise. The empirical results in [28] show that RESAMPLE generates better estimates of uncertainty compared with INJECT, which is why, in our own empirical work (Section 5), we include RESAMPLE and not INJECT.

Unlike the information-based uncertainty measures, which are model-independent, the stability-based measures do depend on the prediction algorithm itself. Furthermore, the stability-based measures are interaction-wise measures. On the other hand, in order to obtain good estimates for the predicted rating variance, several different models have to be learned, hence, computing these measures can be slow. In [28], this problem motivates FAST-RESAMPLE, which approximates RESAMPLE by using in each  $\Phi^{(k)}$  some model parameters that have been computed once on the original ratings.

In this paper, we introduce a perturbation strategy that has not previously been evaluated in the context of uncertainty estimation. It is model-specific and, indeed, depends on using models whose parameters require random initialisation. The strategy, which we designate ENSEMBLE, consists in multi-modelling for different initialisation points. The predicted rating is the mean of the predicted ratings of the individual models, and the estimated uncertainty is their standard deviation as before:

$$\hat{r}_{ui} = \frac{1}{N} \sum_{k=1}^N \Phi^{(k)}(u, i) \quad (5)$$

$$\rho_{ui} = \frac{1}{N} \sqrt{\sum_{k=1}^N (\Phi^{(k)}(u, i) - \hat{r}_{ui})^2} \quad (6)$$

Using different initialisation points for the perturbations in multi-modelling is interesting since it means that all of the training ratings are used to learn each model  $\Phi^{(k)}$ ; by contrast, RESAMPLE uses only a subset of the training data for each model. This might lead to better rating estimation, and we see some evidence of this in the results of our experiments.

For our empirical work (Section 5), the models in the ensemble use matrix factorization (MF) (Section 2). MF is known to suffer from strong instability, depending on the initialisation of the vectors. This has been shown by [16] in a recommendation context and also by [5] and [35] for topic modelling. Ensembles of MF models that have been initialised differently can result in lower error and greater stability but also enable us to estimate uncertainty.

### 3.3 Error-based uncertainty

Uncertainty in rating predictions can be estimated through the expected error in those predictions. The more certain a prediction is, the more accurate the prediction should be. Zhu et al. [44] give a method for estimating prediction error, and then use this as the measure of uncertainty.

Zhu et al. [44] use a cross-validation procedure to obtain a set of prediction errors  $E$ . First, they partition the training ratings  $R$  into  $K$  folds. It follows that each known rating  $r_{ui} \in R$  will appear in exactly one test fold. Then, they learn  $K$  models,  $\Phi^{(k)}$  for  $k \in 1, \dots, K$ , where  $\Phi^{(k)}$  is learned from all the training ratings except those in the  $k$ -th fold. Now, they can make a prediction  $\hat{r}_{ui}$  that corresponds to each known rating  $r_{ui} \in R$ . Specifically, if  $r_{ui}$  is in the  $k$ -th fold, then  $\Phi^{(k)}$  predicts  $\hat{r}_{ui}$ . This allows a calculation of training error,  $e_{ui} = r_{ui} - \hat{r}_{ui}$ . At the end of this cross-validation phase, they have a set of prediction errors  $e_{ui} \in E$  for each  $r_{ui} \in R$ . Finally, they learn two models. One is a ratings prediction model  $\Phi$ , learned from the entire ratings training set  $R$ . The second is an error prediction model,  $\mathcal{E} : U \times I \rightarrow \mathbb{R}^+$ , that can predict the rating prediction errors, which is learned from  $E$ .

At prediction time, we want to compute uncertain predictions,  $\langle \hat{r}_{ui}, \rho_{ui} \rangle$ . The predicted rating,  $\hat{r}_{ui}$ , is given by the final rating prediction model,  $\Phi$ ; the uncertainty level,  $\rho_{ui}$ , is the estimated prediction error, given by model  $\mathcal{E}$ .

In Zhu et al. [44], FunkSVD was chosen both as  $\Phi$  and  $\mathcal{E}$ . That is, one FunkSVD model  $\Phi$  is responsible for the rating predictions, while another FunkSVD model  $\mathcal{E}$  estimates uncertainty (i.e. errors). In Section 5, we designate this strategy by EB-FunkSVD. More precisely, in EB-FunkSVD, we have:

$$\hat{r}_{ui} = p_u^T q_i \quad (7)$$

$$\rho_{ui} = p_u'^T q_i' \quad (8)$$

where  $p_u$  and  $q_i$  are embeddings learned from  $R$  and  $p_u'$  and  $q_i'$  are embeddings learned from  $E$ .

Nevertheless, FunkSVD is only one of many possible uncertainty estimators  $\mathcal{E}$ . During our empirical study, we found EB-FunkSVD to perform below our expectations. For some of our evaluation metrics, much simpler estimates, such as NEG-ITEM-SUPPORT, had better performance. These results raised a concern about possible over-complexity of FunkSVD for the uncertainty estimator. For this reason, we also experimented with a much simpler choice of  $\mathcal{E}$ , which we refer to as EB-Linear.

In EB-Linear, we define linear weights  $b_u \in \mathbb{R}$  for  $u \in U$  and  $b_i \in \mathbb{R}$  for  $i \in I$  for every user and item. Now, we propose that the uncertainty for a user-item pair (i.e. the predicted error) to be simply the sum of the user's and item's weights. With this formulation, instead of  $\mathcal{E}$  having a  $d$ -dimensional embedding to model each user and item, there is only a single scalar parameter for each. More precisely, in EB-Linear, we have:

$$\rho_{ui} = b_u + b_i \quad (9)$$

The user and item weights,  $b_u$  and  $b_i$ , are learned, by gradient descent, to minimize the mean squared error between the predicted and observed errors.

Finally, we also acknowledge the contribution by Cleger-Tamayo et al. [13], where another uncertainty estimation approach based on prediction errors was introduced. There, uncertainty estimation is viewed as a binary classification task, where the goal is to segregate certain from uncertain predictions. This task is solved by learning a classifier that predicts whether a user-item interaction is uncertain or not. The classifier is learned based on both the prediction errors committed by the rating prediction model (in their case, a  $k$ -nearest neighbours model) and the ratings given by similar users to the same item. We will not further explore this approach for two reasons. First, it is restricted to neighborhood models. Second, the segregation of items into two types (certain and uncertain) is quite different from the formulation used in our work, where we seek to estimate the level of uncertainty,  $\rho_{ui}$  (Section 2). In any case, EB-FunkSVD and EB-Linear are also based on prediction errors and can be applied to a wider class of collaborative filtering models.

### 3.4 Distribution-based uncertainty

The Recommender Systems literature contains several methods that consider each user-item rating to be a random variable, each having a certain probabilistic distribution  $\mathcal{F}|\times_{\Theta}$ , where  $\Theta$  is the set of distribution parameters. Modelling the ratings with a distribution rather than a point estimate allows the recommender to produce a much richer output, from which it is possible not only to obtain point estimates for the ratings but also to infer the uncertainty associated with these point estimates.

The Gaussian distribution emerges as a natural choice: its mean,  $\mu$ , and variance,  $\sigma^2$ , are obvious choices for the rating point estimates and their uncertainties. In this case,  $r_{ui} \sim \mathcal{N}(\mu_{ui}, \sigma_{ui}^2)$ , and the task of estimating  $r_{ui}$  is performed by estimating  $\Theta_{ui} = \langle \mu_{ui}, \sigma_{ui}^2 \rangle$  for every user-item pair.

One notable algorithm to employ a Gaussian model is Probabilistic Matrix Factorization (PMF) [30]. In PMF, the ratings are assumed to follow a Gaussian distribution, with a fixed variance parameter  $\sigma^2 = 1$ . As before, let

$p_u$  and  $q_i$  be user and item embeddings, both of dimension  $d$ . Then, the likelihood of the training data can be written as:

$$\mathbb{P}(R) = \prod_{i \in I} \prod_{u \in U} [\mathcal{N}(r_{ui} | p_u^T q_i, 1)]^{\delta_{ui}} \quad (10)$$

where  $\mathcal{N}(\cdot | \mu, \sigma^2)$  is the probability density function of the Gaussian distribution with mean  $\mu$  and variance  $\sigma^2$ , and  $\delta_{ui}$  is an indicator function that is equal to 1 if user  $u$  rated item  $i$  and 0 otherwise. The latent factors are assumed, in prior, to follow a zero-mean spherical Gaussian distribution. In this case, maximising the log-posterior distribution is equivalent to minimising the sum of squared errors subject to  $L_2$  regularisation terms:

$$\sum_{u \in U} \sum_{i \in I} \delta_{ui} (r_{ui} - p_u^T q_i)^2 + \lambda_U \sum_{u \in U} \|p_u\|^2 + \lambda_I \sum_{i \in I} \|q_i\|^2 \quad (11)$$

Notice that, although having a different motivation, the PMF model optimizes an objective function which is identical to FunkSVD's (Equation 2).

PMF assumes every user-item pair to have the same variance. Not only is this a very strong assumption in general, it also means that PMF is not suitable for uncertainty estimation, since every user-item pair would have the same estimated uncertainty  $\sigma^2$ . For this reason, Wang et al. [41] extend PMF to Confidence-aware Probabilistic Matrix Factorization (CPMF). There, each user and item is assumed to have a latent variance parameter,  $\sigma_u$  and  $\sigma_i$ , respectively. The variance of each user-item pair is obtained by some function  $\gamma : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^+$  that composes these two latent variance parameters. Therefore, the likelihood of the training data is:

$$r_{ui} \sim \mathcal{N}(p_u^T q_i, \gamma(\sigma_u, \sigma_i)) \quad (12)$$

The latent factors,  $p_u$  and  $q_i$ , and the variance parameters,  $\sigma_u$  and  $\sigma_i$ , are estimated through gradient descent on the negative log-likelihood.

Wang et al. [41] use a simple product for  $\gamma$ . In our experiments (Sections 5), we do the same and this is what we are referring to when we write CPMF in that later section of this paper.

Wang et al. [41] also propose a Bayesian version of the algorithm, similar to [37]. The idea of the Bayesian framework is to automatically control model complexity through prior distributions in order to alleviate the effects of over-fitting and to improve model generalisation.

The next section presents further distribution-based methods for the special case where ratings are not treated as continuous variables.

### 3.5 Multinomial-based uncertainty

Ratings are typically discrete; for example, the ratings in the Netflix Prize dataset are integers from 1 to 5 [6]. But, in rating prediction, ratings are often treated as continuous variables. In other words, the model predicts real values:  $\Phi : U \times I \rightarrow \mathbb{R}$ . Then, as we have seen, there are various ways of augmenting each predicted rating with an estimated uncertainty level.

Multinomial models treat predicted ratings as discrete variables. For each point on the rating scale, they predict a probability. Hence, the output of a multinomial prediction model is a vector, containing one probability for each score  $s$  on the rating scale  $S = \{s_1, \dots, s_m\}$ , where  $\mathbb{P}(r_{ui} = s)$  for the probability that user  $u$  will assign rating  $s \in S$  to item  $i$ , and  $\sum_{s \in S} \mathbb{P}(r_{ui} = s) = 1$ . For instance, in the Netflix case, the model will predict  $\mathbb{P}(r_{ui} = s) \forall s \in \{1, \dots, 5\}$ . This idea has motivated several recommendation algorithms, including the User Rating Profile Model (URP) [27], Bernoulli Matrix Factorization (BeMF) [33] and OrdRec [26]. From a statistical standpoint, this way of formulating rating prediction consists of assuming  $r_{ui}$  follows a multinomial distribution. Hence, it is possible to extract not only point estimates for the ratings, but also dispersion statistics, which can be used as estimates of uncertainty.

The BeMF model is one way to estimate the vector of probabilities [33]. BeMF learns  $m$  different Bernoulli factorization models, one for each point on the rating scale, i.e. it learns a set of models  $\Phi = \{\Phi_1, \dots, \Phi_m\}$ . To

learn each individual model  $\Phi_s$ , it uses a modified version of the training set  $R_s = \{(u, i, r_{uis})\}$ , such that:

$$r_{uis} = \begin{cases} 1, & r_{ui} = s \\ 0, & \text{otherwise} \end{cases} \quad \forall (u, i, r_{ui}) \in R \quad (13)$$

Each Bernoulli factorization model  $\Phi_s$  makes predictions in the same way as the FunkSVD method (Equation 1), but with the difference that the predictions are scaled through a logistic function, which bounds them to the  $[0, 1]$  interval. To learn the latent factors, for each individual model, the cross-entropy loss function is used. The loss for a single training instance is:

$$\Phi_s(u, i)^{r_{uis}} (1 - \Phi_s(u, i))^{1-r_{uis}} \quad (14)$$

The parameters of the model are updated by performing gradient descent on this loss function. Finally, the probabilities are given by:

$$\mathbb{P}(r_{ui} = s) = \frac{\Phi_s(u, i)}{\sum_{s \in S} \Phi_s(u, i)} \quad (15)$$

With the probabilities estimated, [33] proposes that the predicted rating will be the most probable value in the rating scale, and the uncertainty estimate will be the probability mass lying outside this most probable rating value:

$$\hat{r}_{ui} = \arg_{s \in S} \max \mathbb{P}(r_{ui} = s) \quad (16)$$

$$\rho_{ui} = 1 - \max_{s \in S} \mathbb{P}(r_{ui} = s) \quad (17)$$

A rather different multinomial approach, OrdRec, is proposed in [26]. While OrdRec has a very different formulation from BeMF, its ultimate goal is also to predict the vector of probabilities. OrdRec learns a single rating prediction model, e.g. using FunkSVD, which will be denoted  $\Phi$ . Then, for every  $s \in \{s_1, \dots, s_{m-1}\}$ , the cumulative probability function for the ratings is:

$$\mathbb{P}(r_{ui} \leq s) = \frac{1}{1 + e^{\Phi(u, i) - t_{us}}} \quad (18)$$

where  $t_{us}$  for each  $s \in S$  is a user-specific set of threshold parameters. Notice that  $\mathbb{P}(r_{ui} \leq s_m) = 1$  as this is a cumulative distribution. After estimating the cumulative functions, the individual probabilities are obtained by:

$$\mathbb{P}(r_{ui} = s_i) = \mathbb{P}(r_{ui} \leq s_i) - \mathbb{P}(r_{ui} \leq s_{i-1}) \quad (19)$$

Now, for every  $u \in U$ , let  $t_{u0} = 0$ . Then, the gaps between a user's thresholds are encoded as a set of parameters  $\beta_{u1}, \beta_{u2}, \dots, \beta_{um-1}$ , such that

$$t_{us_i} = t_{us_{i-1}} + e^{\beta_{usi}}, \quad \forall i \in \{1, \dots, m-1\} \quad (20)$$

Therefore, the whole set of parameters to be learned for OrdRec includes the underlying model's parameters, as well as  $\{\beta_{u1}, \beta_{u2}, \dots, \beta_{um-1}\} \forall u \in U$ . The likelihood function for the model is:

$$\prod_{i \in I} \prod_{u \in U} \prod_{s \in S} (\mathbb{P}(r_{ui} = s))^{\delta(r_{ui}=s)} \quad (21)$$

Learning is performing by stochastic gradient descent on the negative log-likelihood function.

With the rating distribution estimated, there are many ways in which rating predictions and uncertainty estimates can be extracted. Koren and Sill [26] argue that the distribution's average and some measure of its dispersion can be used to predict ratings and uncertainty respectively. Moreover, any distribution dispersion metric can be employed, such as standard deviation, Gini impurity or entropy. Among these, they empirically

found that the standard deviation performed best for uncertainty and, therefore, we will use the same when we estimate uncertainty using OrdRec in this paper:

$$\hat{r}_{ui} = \sum_{s \in S} s \mathbb{P}(r_{ui} = s) \quad (22)$$

$$\rho_{ui} = \sqrt{\frac{1}{m} \sum_{s \in S} (s^2 \mathbb{P}(r_{ui} = s)) - m \hat{r}_{ui}^2} \quad (23)$$

Moreover, we highlight one concern regarding the uncertainty estimates given by Equation 23: when  $\hat{r}_{ui}$  is either too small or too big, that is, close to  $s_1$  or to  $s_m$ , then the uncertainty  $\rho_{ui}$  will always be very low. This happens because, in order to predict one of the distribution's extremes, the vast majority of the distribution's probability have to fall on this extreme rating value, leading to very low standard deviation. Intermediate rating values do not suffer from the same problem because, in this case, the distribution's probability can be dispersed symmetrically around that intermediate rating value.

We have concluded our review of ways of *estimating* ratings prediction uncertainty. We turn our attention now to ways of *evaluating* these estimates.

#### 4 EVALUATING UNCERTAINTY ESTIMATES

After training a rating prediction model, it is common to evaluate its performance on a test set, this being a set of known ratings that was held-out during training. The most common rating prediction evaluation metrics, such as root mean squared error (RMSE), simply compare the predicted ratings to the known ratings in the test set  $T$ , that is:

$$\text{RMSE} = \sqrt{\frac{\sum_{r_{ui} \in T} (\hat{r}_{ui} - r_{ui})^2}{\#T}} \quad (24)$$

In this case, the known ratings give us a 'ground-truth' for the evaluation. Similarly, after estimating the uncertainty of predicted ratings, it is desirable to evaluate the quality of the uncertainty estimates. But, evaluation of the uncertainty estimates is not as simple as evaluating the predicted ratings because there are no 'ground-truth' values that can be compared with the uncertainty estimates.

Our review of ways of evaluating estimates of rating prediction uncertainty starts with a discussion of expected correlations. Subsequent sections then correspond to the different tasks that we laid out in Section 2, namely rating prediction and top- $n$  recommendation, the latter being split into rating-based ranking, uncertainty-based filtering and probability-of-relevance ranking.

##### 4.1 Expected correlations

Despite the differences in the uncertainty estimates that we reviewed in Section 3, there are 'patterns' that we expect to observe, including the following:

**Negative correlation with user profile size:** The less we know about a user's tastes, the more uncertain we expect that user's predictions ratings to be.

**Negative correlation with item support:** Similarly, the less that an item has been interacted with, the less certain we expect that item's predicted ratings to be.

**Positive correlation with user rating variance:** The more a user's ratings vary from each other, the less certain we expect that user's predictions to be.

**Positive correlation with item rating variance:** The more an item's ratings vary from each other, the less certain we expect that item's predictions to be.

To the best of our knowledge, we are the first to propose checking these correlations. Computing them can serve as an initial check that a particular way of estimating uncertainty behaves, on the whole, in the way we might expect. We have chosen to carry out this check using Spearman rank correlation, because the correlations might not always be linear.

It is important to point out that some of these correlations might not always hold for individual user-item interactions. For example, in the movie domain, a new movie from a famous director is usually a very safe and expected recommendation, even though the movie, being newly-released, will have low item support.

We also expect to see some correlations between the uncertainty estimates and prediction error and top- $n$  recommendations accuracy:

**Prediction error:** The more wrong a predicted rating is, the less certain we expect the prediction to be.

**Recommendation accuracy:** Similarly, the more accurate a set of top- $n$  recommendations is, the more certain we expect the predicted rating for each item in the top- $n$  to be.

The literature offers several metrics that we can compute to confirm these correlations, and we explore these in more detail in the following subsections.

## 4.2 Rating prediction

Uncertainty measures are expected to reflect prediction errors. For this reason, the foremost method for evaluating the quality of estimates of rating prediction uncertainty consists of simply evaluating the correlation between the prediction errors in a holdout test set and their estimated uncertainty. Different correlation coefficients can be used. For example, Pearson correlation can be used to measure the linear correlation, while Spearman correlation can be applied if rank correlation is preferred. Later, we refer to these correlations (between rating prediction errors and uncertainty estimates) as Pearson $_{\rho}$  and Spearman $_{\rho}$ .

Mazurowski [28] proposes a visualization of the correlation between uncertainty and prediction error. In fact, Mazurowski refers to “reliability estimates” on predicted ratings, rather than uncertainty estimates. But, as his concept of reliability is the opposite of uncertainty, his visualisation can be used here, after some adaptations. In what follows, we ‘translate’ his proposals to our uncertainty framework.

Mazurowski splits the predictions for the test set into  $B$  equal-sized bins based on, and ordered by, their uncertainty levels — that is, bin 1 contains the least uncertain predictions and bin  $B$  the most uncertain ones. For each bin  $b$ , Mazurowski calculates an error threshold,  $\epsilon_b$ . The error threshold for bin  $b$  measures how inaccurate the predicted ratings within the bin are. The thresholds are calculated such that Bayesian confidence intervals of the form  $[\hat{r}_{ui} - \epsilon_b, \hat{r}_{ui} + \epsilon_b]$  contain the real rating  $r_{ui}$  for at least  $(1 - \alpha) \times 100\%$  of the bin’s predictions, where  $\alpha \in [0, 1]$  is a fixed significance level. The quality of the confidence intervals are determined by the curve of the error thresholds against their confidence bin.

One problem with the latter method is that the curves will depend on the choice of  $\alpha$ . In fact, the reliance on  $\alpha$  means that each uncertainty estimate will have a family of curves (for each value of  $\alpha$ ), which makes it more difficult to compare uncertainty estimates. To solve this issue, and also to adapt Mazurowski’s confidence intervals to our uncertainty framework, we propose to use uncertainty bins, rather than confidence bins, and to calculate the average RMSE in each of these uncertainty bins. In this case, given  $B$  equal-sized uncertainty bins, we calculate the following for each bin  $b$ :

$$RMSE_b = \sqrt{\frac{\sum_{\rho_{ui} \in b} (\hat{r}_{ui} - r_{ui})^2}{\#\rho_{ui} \in b}} \quad (25)$$

where the denominator is the number of test instances falling into bin  $b$ . Similarly to before, we then plot the curve of the  $RMSE_b$  values against their bin index. We refer to these as RMSE-uncertainty curves. Now these

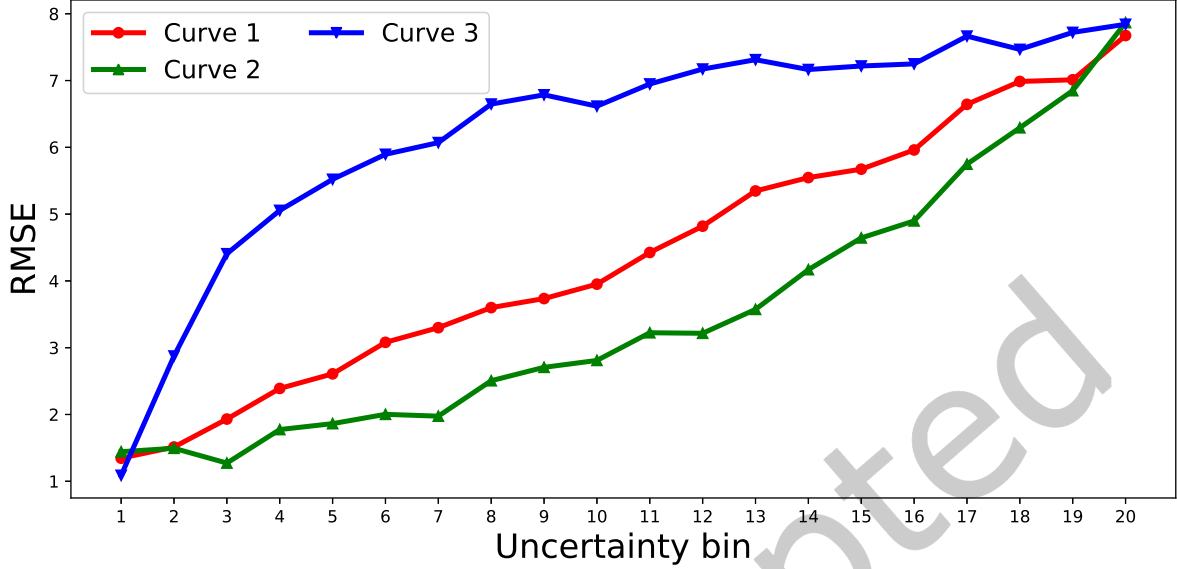


Fig. 1. Example RMSE-Uncertainty curves.

curves have no reliance on additional parameters other than the number of reliability bins — in particular, there is no longer the parameter  $\alpha$ .

Nevertheless, a problem that remains is how to interpret curves like these. To illustrate, we refer to Figure 1. The Figure shows three example RMSE-uncertainty curves. Notice that all the curves have very similar  $RMSE_1$  and  $RMSE_B$  values. On the other hand, their behaviours in between these extremes are different. In this case, there is no clear definition that says which curve is preferred, and hence which of the underlying uncertainty estimates is better.

To alleviate this problem, some simple metrics can be extracted from the curves. One option, which has been explored in [28], is to calculate the difference in the values between the first and last bins. In [28], this metric is called  $d_w$ . Here, to avoid confusion due to the fact that we are using RMSE uncertainty curves, we will refer to it as  $\delta RMSE$ :

$$\delta RMSE = RMSE_B - RMSE_1 \quad (26)$$

Higher values of  $\delta RMSE$  are indicative that prediction error grows as uncertainty grows.

In a similar vein, Bobadilla et al. [9] argue that an evaluation metric for an uncertainty measure should penalise predictions where certainty is high but predictive error is also high and should strongly reward predictions where uncertainty is high and predictive error is high. Table 2 schematically describes the desired behaviour.

Bobadilla et al. [9] propose a metric that matches the criteria in Table 2. Like the work in [28], Bobadilla et al.’s Reliability Prediction Improvement metric is given in terms of reliability. We adapt it from reliability to uncertainty and call it Uncertainty Prediction Improvement (UPI).

To define UPI, we first compute the absolute errors  $e_{ui} = |\hat{r}_{ui} - r_{ui}|$ ,  $\forall (u, i, r_{ui}) \in T$ , where  $T$  is a holdout test set. Then, we define the metric criteria,  $C^{UPI}$ , as follows:

$$C_{ui}^{UPI} = e_{ui}(e_{ui} - \bar{e})(\bar{\rho} - \rho_{ui}) \quad (27)$$

Prediction error	Uncertainty	Uncertainty evaluation metric
High	Low	Big penalty
Low	High	Small penalty
High	High	Big reward
Low	Low	Small reward

Table 2. Criteria for a good estimate of rating prediction uncertainty

where  $\bar{e}$  and  $\bar{\rho}$  are the means of the absolute errors and the uncertainty estimates for all the test instances in  $T$ . UPI is then defined as:

$$UPI = \frac{1}{\sigma_e \sigma_\rho \#T} \frac{\sum_{r_{ui} \in T} C_{ui}^{UPI}}{\bar{e}} \quad (28)$$

where  $\sigma_e$  and  $\sigma_\rho$  are the standard deviations of the absolute errors and uncertainty estimates. The divisor term,  $\sigma_e \sigma_\rho \#T$ , standardises the criteria values, while the term  $\bar{e}$  compares the standardised criteria values to the mean absolute error.

[26] take a rather different approach to the evaluation of uncertainty estimates. Instead of using metrics, such as  $\delta RMSE$  and UPI, they build and evaluate what they call *confidence classifiers*. These are binary classifiers that use the uncertainty estimates as predictors (features) to predict whether or not the rating error will be greater than 1. In practice, given a set of predicted ratings  $\hat{r}_{ui}$  and their respective observed ratings  $r_{ui}$ , a confidence classifier is a logistic model, where the uncertainty estimate is the predictive feature, and the model predicts the probability  $\mathbb{P}(|\hat{r}_{ui} - r_{ui}| > 1)$ . Once trained, the confidence classifier is evaluated using a holdout set. Its accuracy on the holdout set is an indicator of the effectiveness of the uncertainty estimate. Koren and Sill use the classifier's Area Under the Curve (AUC) for this.

Care must be taken to avoid training and evaluating the confidence classifier on the instances used to train the rating predictor. In our experiments, we train the rating predictor on the training set, then we use 2-fold cross-validation on the test set to train and evaluate the confidence classifier. In other words, we train the confidence classifier on half of the test instances and evaluate its AUC on the other half of the test instances. We repeat this, after interchanging the two folds. The final results that we report are the average AUC across the two folds. In Section 6, we refer to this approach to evaluation as Error-Uncertainty Classification (EUC).

We highlight that the choice to predict  $\mathbb{P}(|\hat{r}_{ui} - r_{ui}| > 1)$  with this method is arbitrary. The method is general: we could build confidence classifiers based on thresholds other than 1. In our experiments, however, we follow [26]'s use of 1.

### 4.3 Top- $n$ recommendation

In traditional evaluation of recommender models, it has been common to calculate metrics based on prediction error, such as RMSE. Nevertheless, at recommendation time, a top- $n$  items are shown, typically those with the highest predicted ratings. Hence, it is also necessary to evaluate the recommendation accuracy at top- $n$  [38]. Many metrics have been defined for evaluating the top- $n$  accuracy. In this paper, where evaluation of uncertainty estimates is more import than evaluation of top- $n$  accuracy, we use just two top- $n$  accuracy metrics, namely MAP@ $n$  and Recall@ $n$ . Let  $rel_u$  be  $u$ 's ratings in test set  $T$  where  $r_{ui} \geq \theta$ , where  $\theta$  is the relevance threshold. Then,

$$MAP@n = \frac{1}{\#U} \sum_{u \in U} \sum_{k=1}^n Precision@k_u \times \delta(Z_u^n(k) \in rel_u) \quad (29)$$



where

$$\text{Precision@}k_u = \frac{\#\{Z_u^k \cap \text{rel}_u\}}{k} \quad (30)$$

and where  $\delta(Z_u^n(k) \in \text{rel}_u) = 1$  if the  $k$ -th item in  $Z_u^n$  is in  $\text{rel}_u$  and 0 otherwise.

$$\text{Recall@}n = \frac{1}{\#U} \sum_{u \in U} \frac{\#\{Z_u^n \cap \text{rel}_u\}}{\#\text{rel}_u} \quad (31)$$

As we said above, we also want to evaluate the quality of the uncertainty estimates at top- $n$ . Corresponding with Section 2.2, we divide our discussion of ways of doing this into three.

**4.3.1 Rating-based ranking.** As previously stated, we expect recommendation accuracy to be greater in cases where the recommendations are less uncertain. Therefore, one metric to evaluate the uncertainty estimates is the Spearman correlation between some recommendation accuracy metric, e.g. MAP@ $n$ , and the average uncertainty of the recommended items for the user. We will refer to this as Uncertainty Accuracy Correlation (UAC). Notice that, for this metric, smaller values denote better performance.

In a similar vein, Bobadilla et al. [9] propose Reliability Recommendation Improvement, a metric that evaluates reliability estimates in the case of recommendations. Their metric is based on similar criteria to those in Table 2, this time that hits (top- $n$  recommendations that are correct) should be associated with lower uncertainty levels than the non-hits. We present Uncertainty Recommendation Improvement (URI), which is our adaptation from reliability to uncertainty.

Our adaptation starts by defining  $C^{URI}$  as follows:

$$C_u^{URI} = \sum_{i \in Z_u^n | r_{ui} \geq \theta} (\bar{\rho} - \rho_{ui}) \quad (32)$$

Notice that there is a strong relation with  $C^{UPI}$ . However, URI is focused on relevant items, i.e. test set ratings for which  $r_{ui} \geq \theta$ . Hence, in  $C_u^{URI}$ , the mean uncertainty  $\bar{\rho}$  is calculated only over test instances where  $r_{ui} \geq \theta$ .

In this definition of  $C^{URI}$ , based on [9] and similar to  $C^{UPI}$ ,  $\bar{\rho}$  is a global average — the average of all the uncertainty values in test set  $T$ . In other words, it is not the average of these values just for user  $u$ . We believe that averaging over all  $T$  can lead to problems. For example, it is likely that some users will have high overall uncertainty. In an extreme case, there may even be users  $u$  for which  $\rho_{ui} < \bar{\rho} \quad \forall i \in I$ . For such users, if a recommendation hit occurs, then  $C_u^{URI} < 0$  even if the recommended hit is less uncertain than all of the other recommended items for that user. Given a set of recommendations for a user, it is desirable for recommendation hits to have a lower estimated uncertainty than non-hits. For this reason, we propose an alternative definition of URI that is based on  $Z_u^n$ , the top- $n$  items recommended to  $u$ . We redefine it as follows:

$$C_u^{URI} = \sum_{i \in Z_u^n | r_{ui} \geq \theta} (\bar{\rho}_u - \rho_{ui}) \quad (33)$$

where  $\bar{\rho}_u$  is the mean uncertainty for the items in  $Z_u^n$ , i.e. the  $n$  items that are recommended to user  $u$ .

Now, the URI is defined as:

$$URI = \frac{\sum_u C_u^{URI}}{\sigma_{\rho_u} \sum_u \#\{i \in Z_u^n | r_{u,i} \geq \theta\}} \quad (34)$$

where  $\sigma_{\rho_u}$  is the standard deviation of the uncertainty values for  $Z_u^n$ . The denominator is used to guarantee that the URI is a valid metric when comparing different uncertainty measures.

**4.3.2 Uncertainty-based filtering.** In Section 2.2.2, we argued that, in some circumstances, it can be useful to constrain the set of items that might be included in a top- $n$  by discarding candidates whose uncertainty value exceeds some threshold  $\tau$ . As initially proposed in [33], we can use this as a way of evaluating uncertainty estimates. In certain cases, the recommendation accuracy (e.g. its precision) can increase — if the candidate items that are discarded are not only uncertain but also not relevant. In practice, the recommendation accuracy can be measured for a range of different values for  $\tau$ , with the expectation that accuracy will increase as  $\tau$  increases. Nevertheless, restricting the candidate set might also decrease the average predicted rating in the recommended items. For instance, before filtering, a user might have several candidates that receive high predicted ratings but also high uncertainty. If these are filtered out, not only will uncertainty be reduced in the filtered recommendation set, but also the average predicted rating. For this reason, together with the accuracy, it is important to monitor the Mean Predicted Rating. To do so, we use:

$$\text{Mean Predicted Rating@}n = \frac{1}{\#U} \sum_{u \in U} \frac{1}{n} \sum_{i \in Z_u^n} \hat{r}_{ui} \quad (35)$$

Finally, since we are preventing the RS from recommending highly uncertain items to the user, there may be users for whom the RS cannot recommend a full set of  $n$  items. Hence, the recommendation *coverage* should also be measured. Here we measure coverage as:

$$\text{Coverage}_u = \frac{\#Z_u}{n} \quad (36)$$

where  $\#Z_u$  is the number of recommendations made to user  $u$  and  $n$  is the desired recommendation list size. The average coverage is then obtained by averaging the individual coverage of each user.

**4.3.3 Probability-of-relevance ranking.** In RBR, uncertainty estimates are not used for ranking the candidates and selecting the top- $n$ , and therefore uncertainty in the case of RBR has to be evaluated through specific metrics, such as UPI. In PRR, on the other hand, uncertainty estimates are used to rank the candidates and select the top- $n$ ; specifically, we recommend the candidates for which  $P(r_{ui} \geq \theta)$  is highest. This means that, in the case of PRR, the uncertainty estimates can be evaluated through recommendation accuracy metrics. That is, the quality of the recommendation set given by the uncertainty estimates serve as an indirect way to evaluate these estimates.

Furthermore, in the cases where the system is expected to provide uncertainty estimates together with the uncertainty-based recommendations, then these uncertainty values also have to be evaluated. In Section 2.2.3, we propose that  $1 - \mathbb{P}(r_{ui} \geq \theta)$  can be used as the uncertainty estimate. In this case, the uncertainty estimates will increase as we progress through the item ranking. That is, the uncertainty of the top-ranked item will be the smallest, followed by the second, and so on. Under this setup, URI is not a feasible evaluation metric anymore, because the differences  $\bar{\rho}_u - \rho_{ui}$  will always be the largest for the top-ranked items, leading to unrealistically high URI values. On the other hand, the correlation between the uncertainty estimates and accuracy (UAC) is still a feasible metric, and we will use it below.

## 5 EMPIRICAL STUDY

In this section, we report the results of an empirical study that we have conducted. We believe it is the most extensive study in the field to this date. Compared with previous studies, it covers more ways of estimating uncertainty and uses more evaluation methods, as well as making use of two large datasets. Previously, the most comprehensive study was [28], but it compared only information-based and stability-based uncertainty estimates. The study in [9] proposed and empirically tested some uncertainty quality measures but, similarly, it did not consider error-based, distribution-based or multinomial-based uncertainty estimates. Other studies in the field, e.g. [26, 41], use few, or no, uncertainty evaluation metrics, and therefore also lack empirical support for

Dataset	MovieLens	Netflix
num. users, $\#U$	162,542	53,424
num. items, $\#I$	59,047	480,189
num. ratings, $\#R$	25,000,095	100,480,507
sparsity	99.74%	99.61%
avg. num. ratings per user	154	1881
avg. num. ratings per item	423	209
rating scale	$\{0.5, 1, 1.5, \dots, 5\}$	$\{1, 2, \dots, 5\}$

Table 3. Characteristics of the datasets used in our experiments.

their uncertainty estimates. Finally, [33] has two major issues with their experiments: it does not compare BeMF against OrdRec, which has a similar motivation to it; and it has an unconventional recommendation evaluation, where candidates for recommendation are taken only from the test set and therefore unrated items are never recommended.

Our implementation makes considerable use of PyTorch [34] and is partially inspired by the Spotlight Python libraries.<sup>1</sup> For reproducibility, all the code used is available.<sup>2</sup>

### 5.1 Models

We compare the following ways of estimating uncertainty, each of which was described in Section 3: NEG-ITEM-SUPPORT, ITEM-VARIANCE, RESAMPLE, ENSEMBLE, EB-FunkSVD, EB-Linear, CPMF, BeMF and OrdRec. As explained previously, we do not include the user-wise estimates, NEG-USER-SUPPORT or USER-VARIANCE: they assign the same level of uncertainty to all of a user’s candidate items, which is not helpful to a top- $n$  recommender system.

Most of these techniques rely on a separate rating predictor; only CPMF, BeMF and OrdRec have their own methods for predicting the ratings. For those that need a separate rating predictor, we use FunkSVD (Equation 1). Moreover, FunkSVD is also the underlying rating prediction model  $\Phi$  in the case of OrdRec and BeMF, and has the same formula as the distribution average in the case of CPMF (Equation 12).

Furthermore, we also use FunkSVD separately as a baseline for recommendation accuracy. We will want to determine whether the new techniques have similar or better prediction error (RMSE) and top- $n$  accuracy (MAP@ $n$  and Recall@ $n$ ) than FunkSVD.

### 5.2 Datasets

We use two large ratings datasets: the MovieLens 25 million ratings dataset [20] and the Netflix prize dataset [6]. Table 3 shows some information about each dataset. We chose these datasets because they are by far the most popular ones for the rating prediction task.

### 5.3 Methods

**5.3.1 Data splitting.** We select a random sample of 10,000 users to be test users. The restriction to 10,000 users is to reduce the computational cost of running this extensive set of experiments on such large datasets. For these sampled test users, we chronologically order their ratings and place the last 20% of them in the test set. We use the remaining ratings (from *every* user, not just the test users) for training or validation. Specifically, we order

<sup>1</sup><https://github.com/maciejkula/spotlight>

<sup>2</sup><https://github.com/vcoscrato/uncertain/tree/SnapExplicit>

these remaining ratings chronologically; for each user, we use the first 80% of these ratings for our training set, and the rest for our validation set.

**5.3.2 Tuning.** We have a number of hyperparameters, whose values we need to choose. We tune the models by training them with different hyperparameter values on the training set and evaluating them on the validation set, choosing values that give the lowest validation set RMSE.

Wherever we are using matrix-factorization, the latent feature vector dimension  $d$  and the regularisation strengths  $\lambda_U$  and  $\lambda_I$  are hyperparameters. The latent feature vector dimension is chosen from  $\{50, 100, 200\}$ , while the regularisation strengths are chosen from  $\{0.1, 0.01, 0.001\}$ . The latent feature vectors are initialised by sampling from a zero-centered Gaussian distribution with a standard deviation of 0.01. We set the learning rate to 0.0001 in all cases and used the Adam optimiser to account for learning rate adaptations. Furthermore, to suppress the need to optimise the number of training epochs, we employed early-stopping, monitoring RMSE after every epoch and stopping training when validation RMSE does not improve for five consecutive epochs. This setup is similar to the one used in [16], for example.

For ENSEMBLE and RESAMPLE, which involve multi-modelling (Eqns. 5 and 6), we use  $N = 5$  different models. For RESAMPLE, we use an 80% sample fraction. For the error-based methods, EB-FunkSVD and EB-Linear, we use 2-way cross-validation to create the error matrix  $E$ . We conducted some preliminary experiments with more multi-modelling runs (10 and 20) and more CV-folds (5 and 10), but we obtained similar results to the ones for  $N = 5$  and for 2-way cross-validation.

BeMF trains multiple MF models, one per rating value. For fairness, we ensure that BeMF has a similar number of parameters to all the other models. For example, for the Netflix dataset, where there are 5 possible rating values, if the other models are using  $d = 50$ , then BeMF will learn five models with dimension 10 each.

**5.3.3 Rating prediction evaluation.** To evaluate rating predictions, we measure the RMSE between the ratings in the test set and the predicted ratings. To evaluate the uncertainty estimates, we use the following evaluation techniques that we presented in Section 4.2: Pearson $_\rho$ , Spearman $_\rho$ , UPI, RMSE-Uncertainty curves,  $\delta RMSE$  and  $EUC$ .

**5.3.4 Top- $n$  recommendation evaluation.** We compute recommendations for each of the 10,000 test users. For each of these users  $u$ , the candidate items  $C_u \subseteq I$  that can be recommended to  $u$  are all the items that  $u$  has not rated in either the training or validation sets. Then, for each user, we create a top- $n$ ,  $Z_u^n \subseteq C_u$ ,  $|Z_u^n| = n$ . The way we do this depends on whether we are evaluating RBR, UBF or PRR (Sections 4.3.1, 4.3.2 or 4.3.3, respectively). We measure the accuracy of a top- $n$  using MAP@ $n$  and Recall@ $n$ , averaged over all 10,000 users. An item  $i$  is considered relevant to a user  $u$  if its test set rating  $r_{ui} \geq \theta$  where  $\theta = 4$ . We evaluate the uncertainty estimates using the evaluation techniques that we presented in Section 4.3, namely  $UAC$  and  $URI$ .

## 6 RESULTS

In this section, we present the results. The subsections correspond to the ones in Section 4.

### 6.1 Expected correlations

We start by exploring correlations between the uncertainty estimates and each of: the user profile size ( $\#R_u$ ), item support ( $\#R_i$ ), user rating variance ( $Var(R_u)$ ) and item rating variance ( $Var(R_i)$ ).

We compute Spearman correlations on a set of 100,000 randomly sampled user-item pairs. In other words, we pick a random user and a random item, we estimate the uncertainty  $\rho_{ui}$  and, once we have done this for 100,000 such pairs, we correlate with the statistics mentioned in the previous paragraph. Selecting users and items at random avoids a bias: if we had instead selected users and items from  $R$  ( $r_{ui} \in R$ ), then we would bias these correlations to cases where the user has rated the item.

Model	RMSE	
	MovieLens	Netflix
Baseline	0.8400	0.8618
ENSEMBLE	<b>0.8294</b>	<b>0.8493</b>
CPMF	0.8447	0.8742
BeMF	1.0031	1.0270
OrdRec	0.8832	0.8666

Table 4. Rating prediction: RMSE. (Results for NEG-ITEM-SUPPORT, ITEM-VARIANCE, RESAMPLE, EB-FunkSVD and EB-Linear will be the same as the Baseline.) The best values (lowest) are highlighted in bold.

Figure 2 gives the results. The correlations between NEG-ITEM-SUPPORT and  $\#R_i$  and between ITEM-VARIANCE and  $Var(R_i)$  are trivially perfect and so they are not included.

In the MovieLens dataset, the models' correlation with user profile size, user variance and item variance are very low; EB-Linear and BeMF show strong negative correlation with item support, as expected; but there are counter-intuitive positive correlations with item support for ENSEMBLE, EB-FunkSVD and OrdRec. In the Netflix dataset, some expected positive correlations with user variance and item variance can be observed, with the most noticeable values being achieved by EB-Linear; in the case of item support, EB-Linear and BeMF show strong negative correlation, as expected, while RESAMPLE, ENSEMBLE and EB-FunkSVD show positive correlations to it, which is counter-intuitive.

Notwithstanding the heuristic nature of these expected correlations, the results are quite concerning. Although they are all supposed to be estimates of uncertainty, they behave differently from each other, and there is no uncertainty estimate that convincingly exhibits all the expected behaviours. We highlight, among these concerning results, how EB-FunkSVD and EB-Linear have an opposite correlation with item support. This is particularly odd because both methods are based on the errors of the same baseline, and yet their uncertainty estimates differ drastically. While some of the forthcoming results in the remainder of this work will show good performance for these methods, the correlations we have observed raise a flag to the credibility of uncertainty estimates based on prediction error.

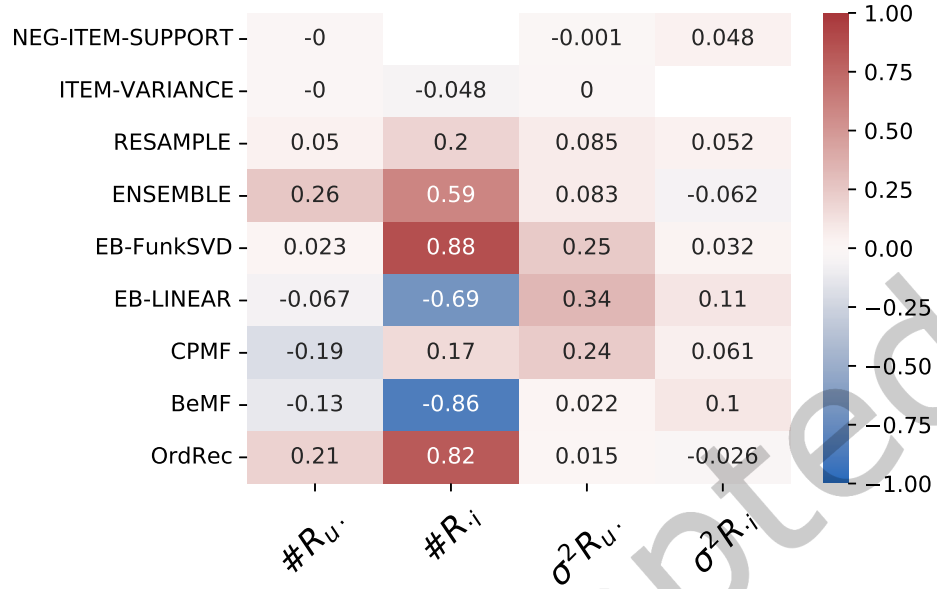
In the remaining subsections, we evaluate the uncertainty estimates using the methods that we reviewed in Section 4.

## 6.2 Rating prediction

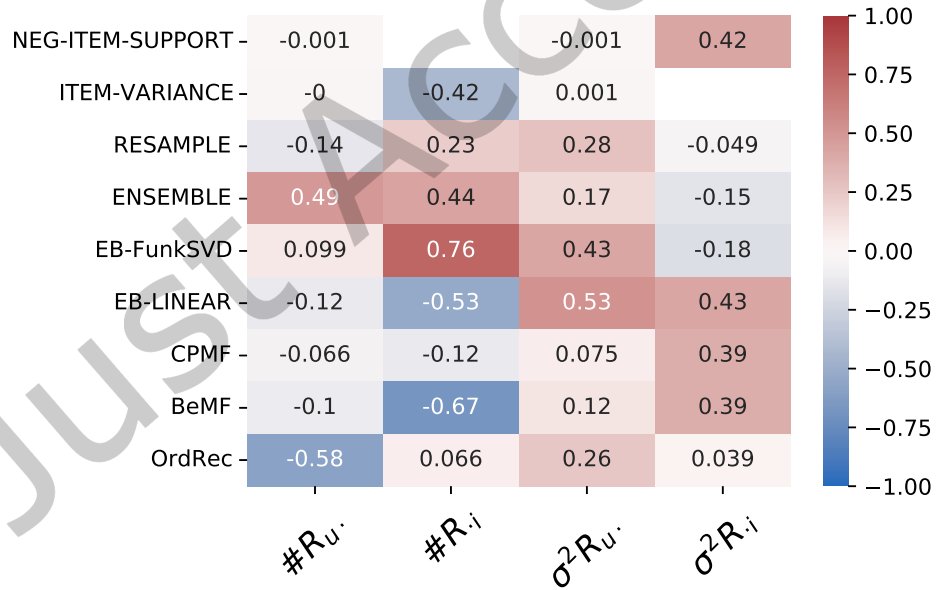
We begin our evaluation of rating prediction by comparing the prediction error of the different models, leaving uncertainty aside. Results are shown in Table 4. Several methods (NEG-ITEM-SUPPORT, ITEM-VARIANCE and RESAMPLE) are not explicitly listed in this Table. This is because, for rating prediction itself, they use the baseline model, and therefore their RMSE is the same as that of the baseline.

It is worth mentioning that, as explained in Section 3, CPMF, BeMF and OrdRec do not directly optimise RMSE, while the baseline model and the models within the ensemble do. In light of this, CPMF shows good performance for both datasets, while ENSEMBLE is able to outperform the baseline in both cases (keeping in mind that low values are better). OrdRec has a slightly worse performance in the MovieLens dataset, while being closer to the other models in the Netflix dataset. BeMF has the worst performance in both datasets.

To assess the quality of the uncertainty estimates from a prediction error point of view, we employ the methods described in Section 4.2. The RMSE-uncertainty curves are given in Figure 3. On the whole, we see what we would expect: as uncertainty grows so does error. In both datasets, EB-Linear seems to be the curve that shows this most strongly. In the Netflix dataset, OrdRec and EB-FunkSVD also exhibit this behaviour quite strongly.



(a) MovieLens dataset.



(b) Netflix dataset.

Fig. 2. Correlations between uncertainty estimates and dataset statistics. We expect negative correlation with  $\#R_{u\cdot}$  and  $\#R_{\cdot i}$ ; we expect positive correlation with  $\text{Var}(R_{u\cdot})$  and  $\text{Var}(R_{\cdot i})$ .

	Pearson $_{\rho}$	Spearman $_{\rho}$	$\delta RMSE$	UPI	EUC
NEG-ITEM-SUPPORT	0.0218	0.0391	0.5064	0.1370	0.5353
ITEM-VARIANCE	0.1266	0.0958	0.3034	0.6029	0.5674
RESAMPLE	0.1187	0.1196	0.3243	0.5205	0.5788
ENSEMBLE	0.0327	0.0215	-0.0865	0.0875	0.5051
EB-FunkSVD	0.1788	0.2008	0.0310	0.5500	0.6245
EB-Linear	<b>0.3463</b>	<b>0.2928</b>	<b>0.9675</b>	<b>1.6851</b>	<b>0.6982</b>
CPMF	0.0449	0.0863	0.3009	0.2444	0.5632
BeMF	0.1114	0.1189	0.3022	0.3204	0.5859
OrdRec	0.2260	0.2526	0.5552	0.6924	0.6731

Table 5. Rating prediction uncertainty evaluation: MovieLens dataset. The best values (highest) are highlighted in bold.

	Pearson $_{\rho}$	Spearman $_{\rho}$	$\delta RMSE$	UPI	EUC
NEG-ITEM-SUPPORT	0.0111	0.0277	0.1408	<b>0.0233</b>	<b>0.5170</b>
ITEM-VARIANCE	0.1340	0.1089	0.3897	0.5075	0.5742
RESAMPLE	0.0872	0.0753	0.2567	0.3632	0.5483
ENSEMBLE	0.0405	0.0262	0.0527	0.1761	0.5152
EB-FunkSVD	0.2161	0.1884	0.4370	0.8133	0.6218
EB-Linear	<b>0.2832</b>	0.2398	<b>0.7382</b>	<b>1.0839</b>	0.6587
CPMF	0.0752	0.0684	0.2174	0.2362	0.5424
BeMF	0.1585	0.1755	0.5319	0.3726	0.5949
OrdRec	0.2788	<b>0.2674</b>	0.6843	0.7673	<b>0.6669</b>

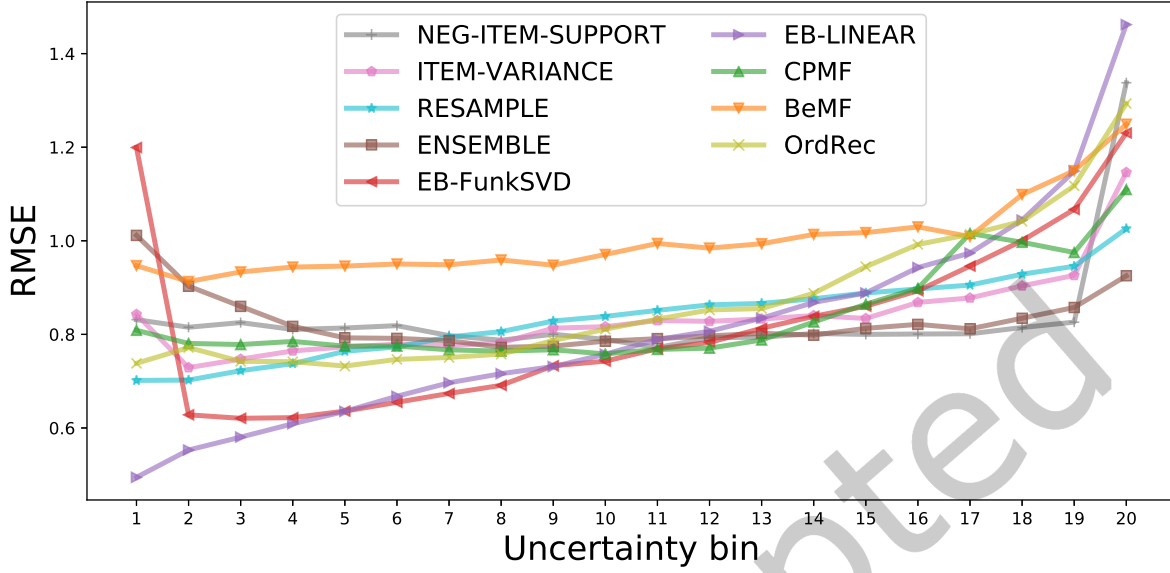
Table 6. Rating prediction uncertainty evaluation: Netflix dataset. The best values (highest) are highlighted in bold.

However, as stated before, these curves are relatively hard to interpret, and for this reason we refrain from drawing strong conclusions from them and turn our attention to the various other evaluation metrics that we presented in Section 4.2. The values for these metrics are shown in Tables 5 and 6.

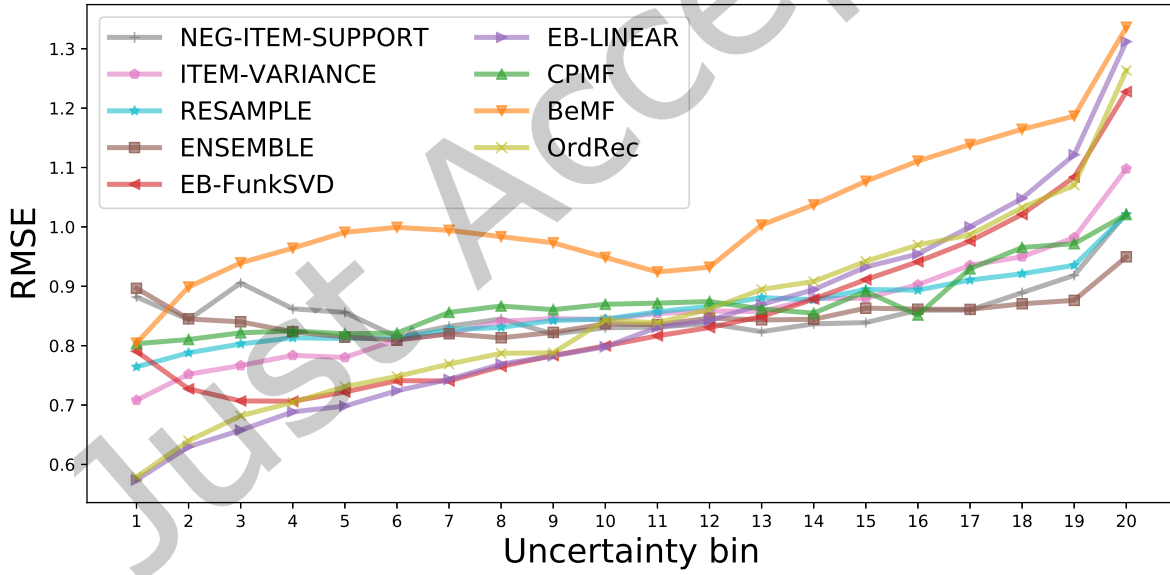
The best overall results are obtained by EB-Linear and OrdRec, while the worst results come from ENSEMBLE. In general, the different metrics in the Table seem quite correlated. That is, the performances of the models have nearly the same ordering across all metrics and both datasets. We find these results to be reassuring, since all the metrics are designed to measure the strength of the relationship between uncertainty and rating prediction error. On this, a few observations can be made. EB-Linear’s uncertainty seems to correlate linearly with error since their Pearson correlation is stronger, while OrdRec’s might correlate non-linearly given the higher Spearman correlation. Moreover, EB-Linear performed strongest according to UPI, while the results for other metrics, such as EUC, are closer, with even a slight advantage for OrdRec in the Netflix dataset.

### 6.3 Top- $n$ recommendation

The results in the previous section evaluate the measures of uncertainty through the prism of prediction error. But, as we noted in Section 2.2, we care more about the quality of the top- $n$  recommendations than the prediction error. Hence, using the ideas from Section 4.3, we here compare the uncertainty estimates in a variety of top- $n$  recommendation scenarios.



(a) MovieLens dataset.



(b) Netflix dataset.

Fig. 3. RMSE-uncertainty curves.



Model	MovieLens		Netflix	
	URI	UAC	URI	UAC
NEG-ITEM-SUPPORT	0.5070	0.0135	0.6747	-0.1292
ITEM-VARIANCE	-0.0590	0.0316	-0.1561	0.2121
RESAMPLE	-0.2371	0.1138	-0.3554	0.1470
ENSEMBLE	-0.3399	0.2125	-0.4491	0.4141
FunkSVD-CV	-0.0867	-0.0218	-0.2825	0.1733
Bias-CV	-0.1548	-0.0489	-0.2983	0.0037
CPMF	0.5196	-0.0242	<b>0.6944</b>	<b>-0.2905</b>
BeMF	<b>0.9527</b>	<b>-0.1619</b>	0.6043	-0.0492
OrdRec	0.2833	-0.0283	0.3808	-0.2143

Table 7. Rating-based ranking uncertainty evaluation: URI and UAC on MovieLens and Netflix datasets. The best values (highest on URI and lowest on UAC) are highlighted in bold.

**6.3.1 Rating-based ranking.** In rating-based ranking (RBR), the top- $n$  are the  $n$  candidate items with the highest predicted ratings. We begin our evaluation by comparing the accuracy of the different models, leaving uncertainty aside. Figure 4 shows the MAP@ $n$  and Recall@ $n$ , averaged over all test users, for different values for  $n \in 1, 2, \dots, 10$ .

The most evident result in the Figure is the very low performance of BeMF in these metrics. This is to be expected. It happens because BeMF is only able to predict rating scale values, that is,  $\hat{r}_{ui} \in S$ ; unlike the other models, it cannot predict a rating that lies between two values on the scale. Therefore, many candidates will receive the same predicted rating – they ‘tie’. The creation of a top- $n$  from such an ordering involves a lot of random tie-breaking. This is presumably why BeMF is not evaluated for RBR in [33]. Apart from BeMF, the remaining models perform similarly, which is reassuring. But there are exceptions: OrdRec has worse MAP for the MovieLens dataset; CPMF is notably worse for the Netflix dataset; and both OrdRec and CPMF have poor recall on the Netflix dataset.

Turning our attention now to the evaluation of the uncertainty estimates, in the RBR scenario our evaluation tools are URI and AUC. The results are shown in Table 7. Remember that higher (positive) URI values denote better performance, while the opposite is the case for UAC. We first notice that the two metrics mostly agree. For both of them, BeMF showed better performance in the MovieLens dataset (although we know that its results are affected by the random choices in the cases of tied ratings), while CPMF was best in the Netflix dataset. On the negative side, RESAMPLE and ENSEMBLE showed poor performance, together with EB-FunkSVD and EB-Linear, which was one of the best performers on the rating prediction task. Therefore, we see a clear disconnect between the models’ ability to estimate uncertainty on the rating prediction and top- $n$  recommendation tasks.

**6.3.2 Uncertainty-based filtering.** In uncertainty-based filtering (UBF), we prevent uncertain candidates (ones whose uncertainty estimate exceeds  $\tau$ ) from being included in the top- $n$ . In the results that we report here, we progressively exclude bigger fractions of the most uncertain candidates, reducing the overall uncertainty of the recommendation lists. More specifically, for each uncertainty estimate, we obtain a distribution using 100,000 randomly-selected user-item pairs and we obtain the 20%, 40%, 60% and 80% percentiles of these distributions as cut-offs for uncertainty. We have chosen this percentile approach due to the different ranges of values that the uncertainty estimates have. Figure 5 shows the MAP@10 and Mean Predicted Rating@10 of the recommendation lists for these different cut-offs. The same Figure also shows the recommendation coverage with each cut applied and a MAP@10 value that considers only users to whom a full 10 recommendations can be given.

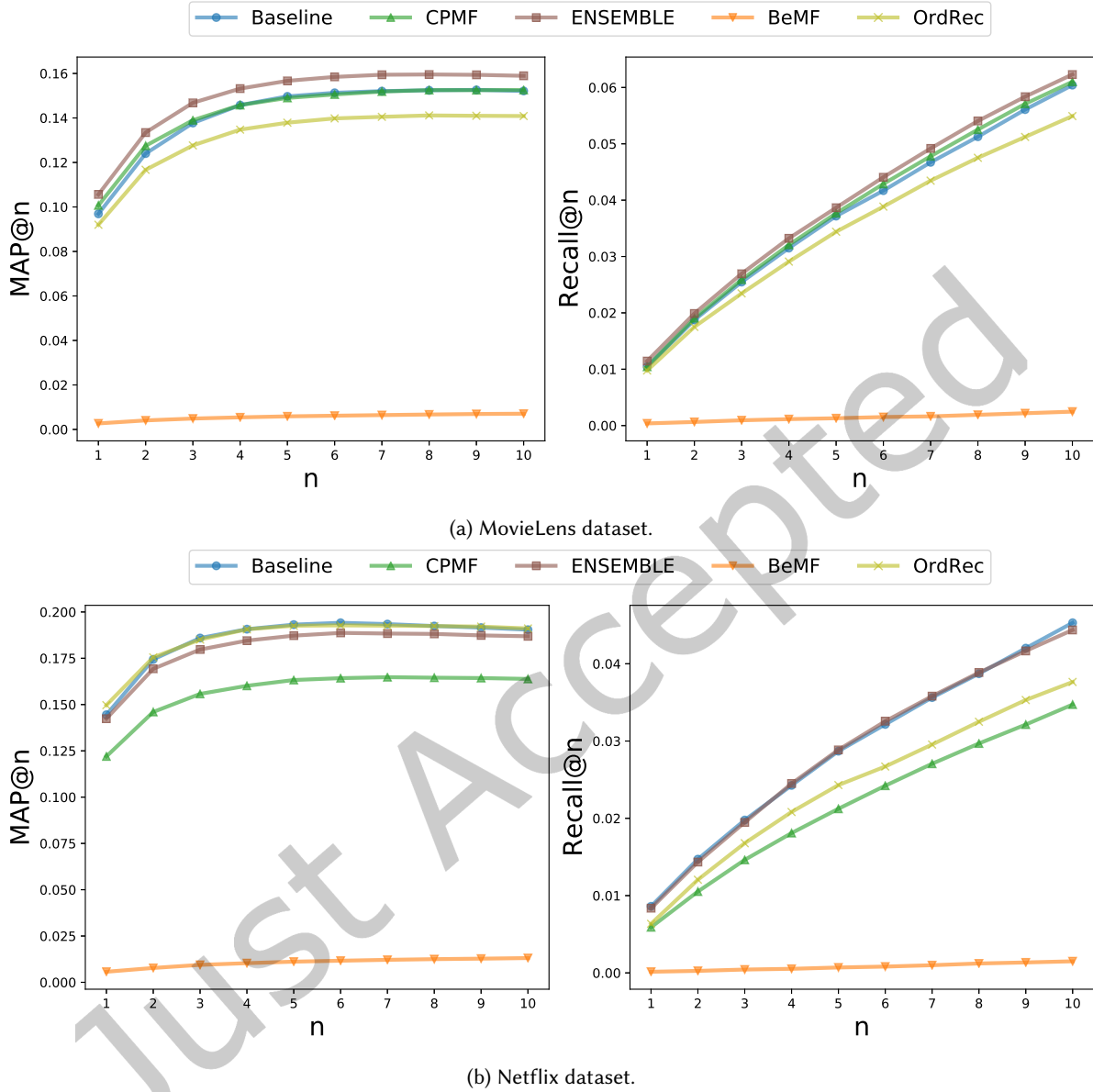


Fig. 4. RBR: MAP@n and Recall@n for  $n \in 1, 2, \dots, 10$ . (Results for NEG-ITEM-SUPPORT, ITEM-VARIANCE and RESAMPLE will be the same as the Baseline.)

The only uncertainty estimates whose MAP increases as uncertain recommendations are excluded are CPMF, BeMF and, in the case of the Netflix dataset only, NEG-ITEM-SUPPORT. Many of the other uncertainty estimates harm precision as uncertain recommendations are excluded. These results are similar to those we observed with URI in the previous section. An exception is OrdRec, which had high URI but did not perform well here. We also

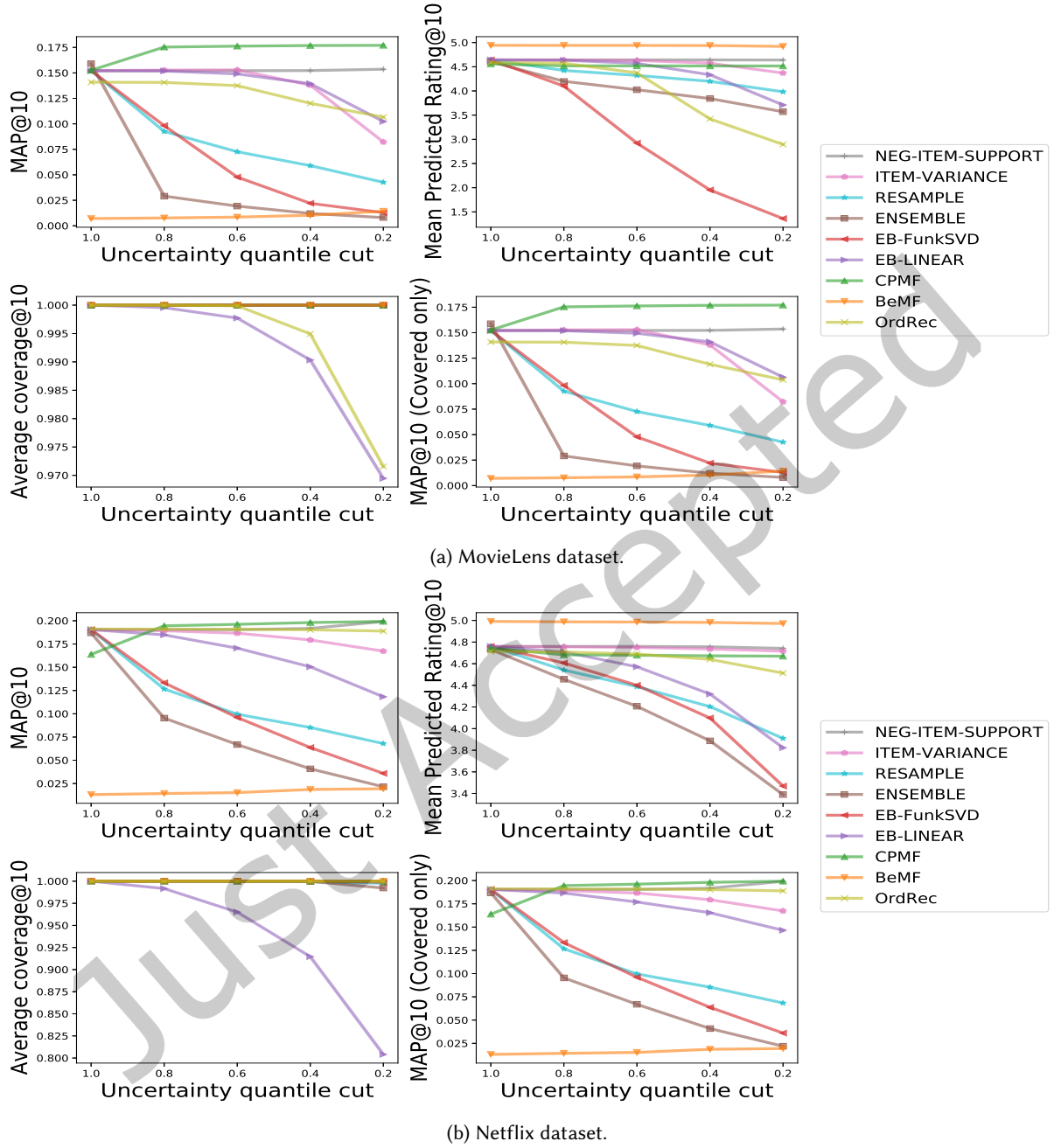


Fig. 5. UBF: MAP, Mean Predicted Rating, Coverage and the MAP for users for whom a full set of recommendations can be made. This is for top-10 recommendations with increasing  $\tau$  to filter different quantiles.

see that the decrease in MAP is, in general, related to the decrease in average relevance as stricter thresholds are applied. That is, the five models with the largest average relevance decrease (ENSEMBLE, EB-FunkSVD, RESAMPLE, ITEM-VARIANCE and OrdRec), are the same ones with the largest drops in MAP. On the other hand, with some uncertainty metrics, such as CPMF and BeMF, it is possible to apply cuts which reduce average relevance only slightly and increase MAP.

Empirically, UBF does not harm coverage. For most of the uncertainty estimates, we can still make 10 recommendations, even as filtering is made stricter. The exceptions are EB-Linear and OrdRec (and the latter only for the MovieLens dataset), and then, even in the worst case (EB-Linear at the 80% quantile cut-off), the model is still able to provide 80% of the requested recommendations. Therefore, the MAP@10 values for users whose coverage is unharmed are very similar to the original ones.

**6.3.3 Probability-of-relevance ranking.** In probability-of-relevance ranking (PRR), we generate recommendations by ranking items according to the probability that their rating is greater than or equal to the relevance threshold,  $\theta$ , which for these datasets is 4. As we discussed in Section 2.2.3, this idea was proposed for BeMF. But it applies directly also to CPMF and OrdRec. The remaining (non-probabilistic) models, appear to be unsuitable to PRR. Nevertheless, in order not to exclude all of them from the comparisons, we can use some heuristic extensions. The ENSEMBLE trains several models, which can be thought of as an  $N$ -dimensional sample of the theoretical population of the rating estimator under all the possible random initialisations. This setup is reminiscent of the central limit theorem. Then, assuming normality, we can use:

$$\mathbb{P}(r_{ui} \geq \theta) = \mathbb{P}\left(\mathcal{N}\left(\frac{1}{N} \sum_{k=1}^N \Phi^{(k)}(u, i), \frac{\rho_{ui}}{\sqrt{N}}\right) \geq \theta\right) \quad (37)$$

Furthermore, the error-based models predict rating prediction errors, that is  $\rho_{ui} = E[|\hat{r}_{ui} - r_{ui}|]$ . Therefore,  $\rho_{ui}$  is an estimate of the ratings' standard deviation. Now, assuming  $r_{ui}$  to follow a Normal distribution with average  $\Phi(u, i)$  and standard deviation  $\rho_{ui}$ , it becomes possible to estimate  $\mathbb{P}(r_{ui} \geq \theta)$  as:

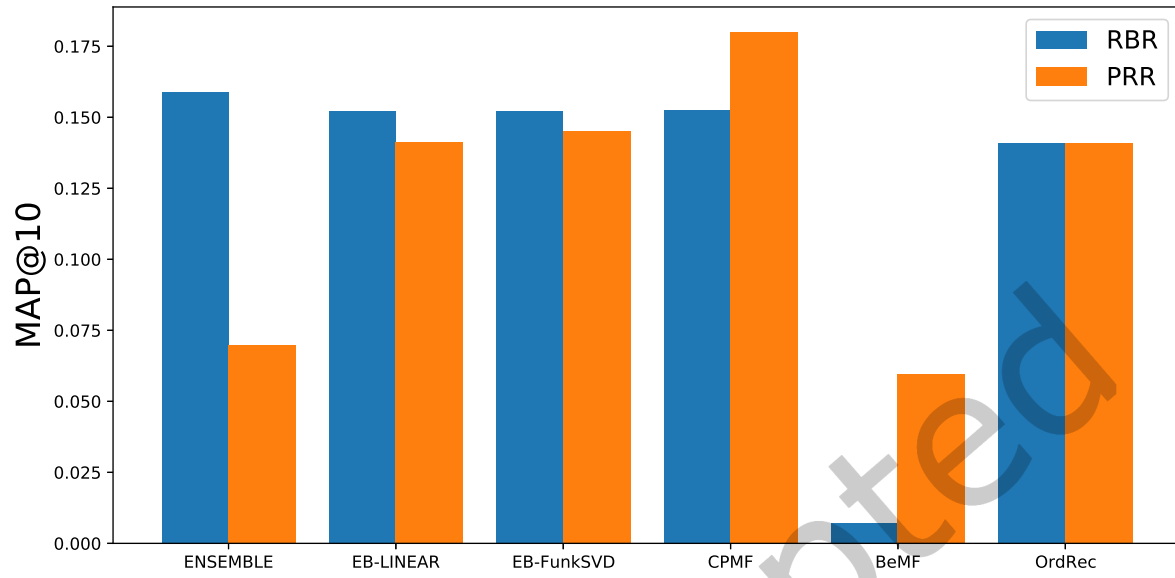
$$\mathbb{P}(r_{ui} \geq \theta) = \mathbb{P}(\mathcal{N}(\Phi(u, i), \rho_{ui}) \geq \theta) \quad (38)$$

With such extensions, we can now compare ENSEMBLE, EB-FunkSVD, EB-Linear, CPMF, BeMF and OrdRec for the task of PRR. As discussed in Section 4.3.3, we evaluate whether PRR, which uses the uncertainty estimates for ranking, produces more accurate recommendation lists than RBR, which ignores the uncertainty estimates. This comparison is shown using MAP@10 in Figure 6.

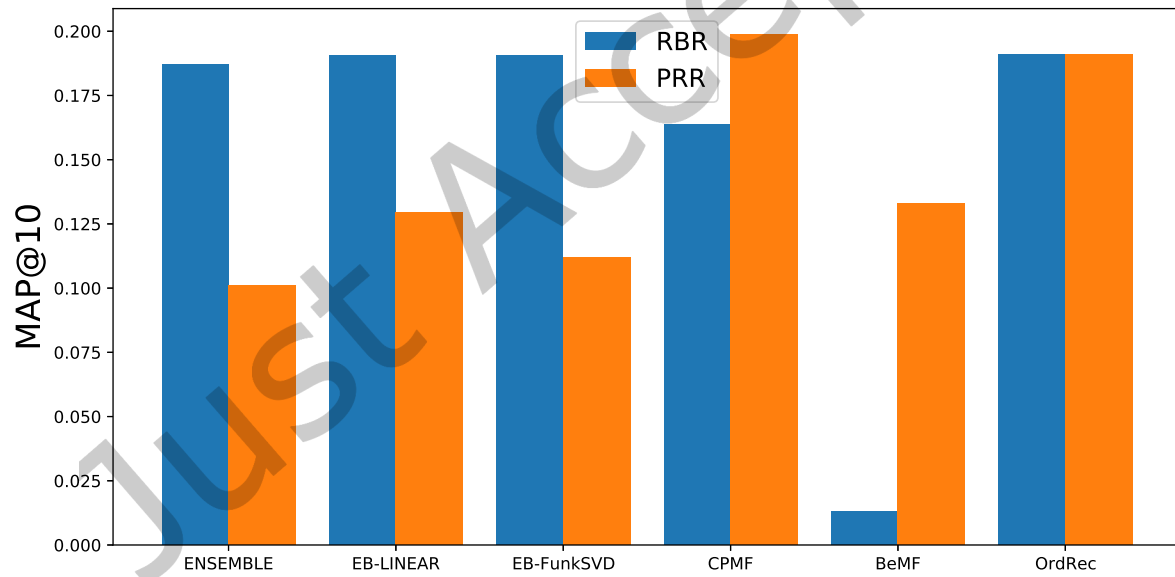
In the Figure, we see that only CPMF and BeMF have higher MAP@10 when their recommendations are selected using PRR compared with selection by RBR. OrdRec's MAP@10 is the same for both. For all the other uncertainty estimates, MAP@10 is harmed. This may be because the way that we extended PRR (from BeMF) to these other models is not well-motivated enough, but it also reflects the poor performance of these models.

Finally, as discussed in Section 2.2.3, the quantity  $1 - \mathbb{P}(r_{ui} \geq \theta)$  can be considered as an uncertainty estimate. To evaluate the usefulness of these estimates, we employ UAC, remembering that *URI* is not applicable in this case due to the correlation between the ranking probabilities and the uncertainties. The results are shown in Table 8. The best performances are achieved by BeMF in the MovieLens dataset, and OrdRec in the Netflix dataset. For CPMF, this uncertainty estimate is uncorrelated with MAP. Unfortunately, for ENSEMBLE, EB-FunkSVD and EB-Linear, the uncertainties are positively correlated with MAP.

This concludes our results. To summarise, we have shown that there is no model that performs better than all the others across all the metrics that we have explored. Nevertheless, we have shown CPMF to be the best model for uncertainty-aware ranking (i.e. for uncertainty-based filtering and probability-of-relevance ranking). On the other hand, our new EB-Linear model had the best uncertainty estimates for most of our rating prediction results, while performing very poorly on the top- $n$ . This highlights a disconnect between the rating prediction task and



(a) MovieLens dataset.



(b) Netflix dataset.

Fig. 6. PRR versus RBR: MAP@10.

Model	MovieLens	Netflix
ENSEMBLE	0.2979	0.2787
EB-FunkSVD	-0.0193	0.2296
EB-Linear	0.0075	0.1180
CPMF	-0.0334	-0.0140
BeMF	<b>-0.2077</b>	-0.1509
OrdRec	-0.0237	<b>-0.2048</b>

Table 8. PRR uncertainty estimates: UAC on MovieLens and Netflix datasets. The best values (lowest) are highlighted in bold.

the top- $N$  recommendation task in the context of uncertainty estimation. Moreover, we saw that the uncertainty estimates behave very differently from each other. This, together with the variations in the performance of the uncertainty estimates across different metrics, raises a major concern about the credibility of some of the uncertainty estimates. Finally, given that top- $n$  recommendations is more important than rating prediction, we believe CPMF to be the most promising of all the models.

## 7 DISCUSSION

So far, we described and empirically evaluated several ways of estimating rating prediction uncertainty. The results show the performance of each uncertainty estimate according to different evaluation criteria. Although we have already raised some general concerns, the results allows us to conduct a further discussion regarding each uncertainty estimate. Furthermore, there are aspects that may be particular to each estimate that may not be explicitly demonstrated by the results. This section discusses these issues and critiques each of the uncertainty estimates.

**NEG-ITEM-SUPPORT:** Characterising the uncertainty of a rating prediction by the number of ratings of the target item has a solid foundation. Its high URI confirms that, when recommended, well-supported items tend to be more relevant. However, its strong performance on this metric might be attributable to the various biases known to exist in the datasets used for offline RS evaluation [12]. Furthermore, as we have already noted, one concern about NEG-ITEM-SUPPORT is that this estimate of uncertainty is not personalised, which does not seem reasonable; we would expect uncertainty to depend on the relationship between an item and the user profile. Another unreasonable property is its model-independence; we would expect uncertainty to depend in part on the quality of the rating prediction model. Another potential issue is that, if used for UBF, NEG-ITEM-SUPPORT will reduce the item candidates to a subset of the most popular ones, greatly decreasing the system's ability to recommend novel, yet relevant, items.

**ITEM-VARIANCE:** Disagreement between users would certainly seem to be a source of uncertainty. In line with this, the positive results achieved by ITEM-VARIANCE in the rating prediction uncertainty evaluation show that the predicted ratings for items about which users disagree tend to be less accurate. On the other hand, we did not find ITEM-VARIANCE to be useful in the top- $n$  recommendation case. In addition to this, and similarly to NEG-ITEM-SUPPORT, ITEM-VARIANCE is also model-independent and not a personalised metric.

**RESAMPLE:** We found that the variation in predictions that ensues from perturbations during model training does correlate with the error in the final predictions, as evidenced by the positive values achieved by RESAMPLE in the rating prediction uncertainty evaluation metrics. On the other hand, this method obtained negative URI, meaning that, in the task of top- $n$  recommendation, quantifying uncertainty by this method is not only ineffective but actually harmful. One possible explanation for this is that recommended

items are the ones with the highest ratings, which means they fall far from the average rating case and are therefore more likely to get noisy predictions.

**ENSEMBLE:** We found ENSEMBLE to have similar, or even worse, results than RESAMPLE in regards to uncertainty evaluation. In fact, given the similar motivation of both, we can conclude that multi-modelling strategies are not useful for uncertainty quantification in the case of top- $n$  recommendation. Nevertheless, we highlight that, in the MovieLens dataset, ENSEMBLE had the best MAP and Recall in the RBR task, although the improvement relative to the baseline is small.

**EB-FunkSVD:** We expect uncertainty to be a quantity related to rating prediction error. The cross-validation methods build on this idea directly: they use regression to predict error, and treat these predicted errors as estimates of uncertainty. Unsurprisingly then, EB-FunkSVD achieves good results in the rating prediction uncertainty evaluation. Unfortunately, when it comes to top- $n$  recommendations, it performs poorly. This may be because EB-FunkSVD minimises the average absolute error, but the recommended items are those with the highest ratings, which are outliers relative to the average, and therefore, might not receive good uncertainty predictions.

**EB-Linear:** Our critique of EB-FunkSVD also applies to EB-Linear. In this case, the results are even more evident: the rating prediction uncertainty evaluation results are even better than those of EB-FunkSVD, but its URI results are even worse. Therefore, we conclude that the simple linear model can accurately recognize the user-item interactions where rating prediction imprecision is most likely to occur, but cannot recognize which recommendations are more or less likely to be correct. To solve this issue, one possibility to be explored in future work is to restrict the error estimation learning only to those interactions which get predicted with the highest ratings.

**CPMF:** CPMF extends the popular PMF model by incorporating variance parameters. Whereas other ways of measuring uncertainty are largely heuristic in nature, CPMF has a great theoretical foundation. Not only that, but our experiments show CPMF to be a successful approach, being the only model able to benefit from the uncertainty-aware recommendation methods (UBF and PRR). Nevertheless, its performance in the, admittedly less important, rating prediction uncertainty evaluation was not impressive. Of course, this may not be a weakness of CPMF; it may, instead, raise a flag about the disconnect between prediction error experiments and top- $n$  recommendation experiments. Of the two, a good performance at top- $n$  recommendation is more important than performance in rating prediction.

**BeMF:** In its original formulation, this model has several particularities that contrast with the others herein. For instance, the rating prediction function of BeMF differs from those of OrdRec and CPMF by focusing on the mode of the rating distributions instead of their average and dispersion. We are grateful to the authors of this method, since their work greatly helped our definitions of UBF and PRR. Nevertheless, the results achieved by their model have been largely underwhelming; although the uncertainty evaluation metrics have been positive in both the rating prediction and the recommendation case, the poor MAP obtained in both RBR and PRR makes it really hard to justify its use compared, for example, with CPMF, especially since BeMF has much higher complexity.

**OrdRec:** OrdRec offers a collaborative filtering algorithm that is suitable to ordinal feedback data. Although this was its main goal, its authors recognized that its multinomial formulation allows the model to also provide uncertainty estimates. In Section 3.5, we raised a concern regarding OrdRec's uncertainty estimates at the limits of the rating scale, which may cause issues. In particular, recommended items are usually those whose predicted ratings fall close to the rating scale upper boundary, and therefore, these items will often receive very low uncertainty estimates, which might not only not reflect reality, but also harms some of the evaluation metrics employed herein (e.g. URI). Furthermore, we have empirically shown that RBR and PRR produce similar results with OrdRec. In light of this, we believe that these uncertainty estimates add small value when compared to other models.

## 8 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

*Conclusions.* Uncertainty in RS is a very important, and yet quite under-explored, topic. In this paper, we have shown how it is possible to accurately quantify the uncertainty in rating predictions using several different methods, with EB-Linear being the most precise. This result can motivate the use of uncertainty estimates in a variety of tasks in the RS field, e.g. active learning, co-training and reinforcement learning. We have also shown that it is possible to improve recommendation accuracy by changing the way a RS ranks items, with the incorporation of uncertainty estimates. In this case, either through filtering (UBF) or probabilistic ranking (PRR), we showed CPMF to be the best contender. In addition, we showed that other models, e.g. EB-FunkSVD and EB-Linear, have a poor performance on estimating uncertainty on top- $n$  recommendations, which makes it hard to defend their use in practical systems despite their success on rating prediction uncertainty estimation.

*Limitations.* Even though the collection of uncertainty estimators explored in this work is the largest yet and the empirical analysis is substantial, we recognize the empirical analysis has some limitations.

First, both datasets that we have used come from the same domain (movies). Movie recommendation has historically been one of the most studied applications of recommender systems, but there are many others, such as shopping, social media, etc. While we expect the behaviour of the uncertainty estimators to generalize to other domains, there is clear value in exploring how different rating patterns will affect the uncertainty estimates.

Second, our empirical analysis uses only a single train/test split. This is a consequence of our decision to opt for chronological train/test splits. We chose, in other words, to preserve the temporal nature of the data, which is intrinsic to recommender systems. This decision does, however, mean that our results lack the kind of statistical significance analysis that could be obtained from using multiple train/test splits. While we expect our results to be robust, due to the large size of our datasets, there is clear value to exploring this further.

Finally, in our empirical analysis, for those uncertainty estimation methods that need a separate rating predictor, we used FunkSVD (Eq. 1) with mean-squared rating prediction error as its loss function (Eq. 2). There may be value in exploring other loss functions (e.g. ranking-based losses). We believe our choice was appropriate because it gave a reasonably fair comparison across the systems; the exploration of uncertainty does not rely on finding recommenders with the very highest accuracy; and, in any case, ranking-based loss becomes more relevant when dealing with implicit feedback data, which is out of the scope of this paper and the main focus of future work (see below).

*Future work.* One avenue for future work is, of course, to address the limitations described above. But, there are at least two further additional avenues for exploration.

First, this work is deliberately limited to the case of explicit feedback; more specifically, we focus on uncertainty estimation for rating prediction. An obvious avenue for future work is to investigate uncertainty estimation for RSs that use implicit feedback (based on user behaviour). We are aware of only a few papers that explore uncertainty estimation in RS that use implicit feedback, e.g. [17, 23, 31, 40]. We are already developing our own models for this case, e.g. [14], building on the foundation that this paper provides.

Second, while our results give strong support to the usefulness of uncertainty estimates in RS, we believe that uncertainty quantification might also be the key to improving beyond-accuracy recommendation goals, e.g. novelty, diversity and serendipity [24]. For example, a serendipitous recommendation can be one with high relevance yet high uncertainty. Nevertheless, the difficulties of measuring serendipity impose great challenges to our ability to test this hypothesis. For this reason, one avenue for future research is to conduct user trials, in which users would provide feedback not only on their item preferences, but also on their surprise regarding recommendations that have different levels of uncertainty.



## ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant No. 18/CRT/6223, which is co-funded under the European Regional Development Fund. We are grateful to Diego Carraro for his assistance with the preparation of this paper.

## ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant No. 18/CRT/6223, which is co-funded under the European Regional Development Fund. We are grateful to Diego Carraro for his assistance with the preparation of this paper.

## REFERENCES

- [1] Gediminas Adomavicius, Sreeharsha Kamireddy, and YoungOk Kwon. 2007. Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance. In *Procs. of the 17th Workshop on Information Technology and Systems*. 152–157.
- [2] X. Amatriain, J.M. Pujol, and N. Oliver. 2009. I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems. In *Procs. of the Seventeenth Conference on User Modeling, Adaptation, and Personalization*. 247–258.
- [3] Andrea Barraza-Urbina. 2017. The Exploration-Exploitation Trade-off in Interactive Recommender Systems. In *Procs. of the 11th ACM Conference on Recommender Systems*. 431–435.
- [4] Edmon Begoli, Tanmoy Bhattacharya, and Dimitri Kusnezov. 2019. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence* 1, 1 (2019), 20–23.
- [5] Mark Belford, Brian Mac Namee, and Derek Greene. 2018. Stability of Topic Modeling via Matrix Factorization. *Expert Systems with Applications* 91 (2018), 159–169.
- [6] James Bennett and Stan Lanning. 2007. The Netflix Prize. In *Procs. of KDD Cup and Workshop*. 3–6.
- [7] Cesare Bernardis, Maurizio Ferrari Dacrema, and Paolo Cremonesi. 2019. Estimating Confidence of Individual User Predictions in Item-based Recommender Systems. In *Procs. of the 27th ACM Conference on User Modeling, Adaptation and Personalization*. 149–156.
- [8] David Block. 1998. Exploring interpretations of questionnaire items. *System* 26, 3 (1998), 403–425.
- [9] Jesus Bobadilla, A. Gutiérrez, F. Ortega, and B. Zhu. 2018. Reliability Quality Measures for Recommender Systems. *Information Sciences* 442 (2018), 145–157.
- [10] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Ganarski. 2013. Risk-Aware Recommender Systems. In *Procs. of the International Conference on Neural Information Processing*. 57–65.
- [11] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* 12, 4 (2002), 331–370.
- [12] Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. 2018. How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility. In *Procs. of the 12th ACM Conference on Recommender Systems*. 224–232.
- [13] Sergio Cleger-Tamayo, Juan M Fernández-Luna, Juan F Huete, and Nava Tintarev. 2013. Being confident about the quality of the predictions in recommender systems. In *European Conference on Information Retrieval*. Springer, 411–422.
- [14] Victor Coscrato and Derek Bridge. 2022. Recommendation Uncertainty in Implicit Feedback Recommender Systems. In *Procs. of the 30th Irish Conference on Artificial Intelligence and Cognitive Science*. Springer.
- [15] Arthur F. da Costa, Marcelo G. Manzato, and Ricardo J. G. B. Campello. 2018. CoRec: A Co-Training Approach for Recommender Systems. In *Procs. of the 33rd Annual ACM Symposium on Applied Computing*. 696–703.
- [16] Edoardo D’Amico, Giovanni Gabbolini, Cesare Bernardis, and Paolo Cremonesi. 2022. Analyzing and improving stability of matrix factorization for recommender systems. In *Journal of Intelligent Information Systems*.
- [17] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. 2017. Gaussian embeddings for collaborative filtering. In *Procs. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1065–1068.
- [18] Simon Funk. 2006. Netflix Update: Try This at Home. <https://sifter.org/simon/journal/20061211.html>.
- [19] Barbara Hammer and Thomas Villmann. 2007. How to process uncertainty in machine learning?. In *European Symposium on Artificial Neural Networks*. 79–90.
- [20] F Maxwell Harper and Joseph A Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.
- [21] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.

- [22] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning* 110, 3 (2021), 457–506.
- [23] Junyang Jiang, Deqing Yang, Yanghua Xiao, and Chenlu Shen. 2020. Convolutional gaussian embeddings for personalized recommendation with uncertainty. *arXiv preprint arXiv:2006.10932* (2020).
- [24] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1 (2016).
- [25] Arnd Kohrs and Bernard Mérialdo. 2001. Improving collaborative filtering for new-users by smart object selection. In *Procs. of the International Conference on Media Futures (ICME)*.
- [26] Yehuda Koren and Joe Sill. 2011. OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions. In *Procs. of the 5th ACM Conference on Recommender Systems*. 117–124.
- [27] Benjamin M Marlin. 2003. Modeling User Rating Profiles for Collaborative Filtering. *Procs. of the 16th International Conference on Neural Information Processing Systems* (2003), 627–634.
- [28] Maciej A. Mazurowski. 2013. Estimating Confidence of Individual Rating Predictions in Collaborative Filtering Recommender Systems. *Expert Systems with Applications* 40, 10 (2013), 3847–3857.
- [29] Sean M. McNee, Shyong K. Lam, Catherine Guetzlaff, Joseph A. Konstan, and John Riedl. 2003. Confidence Displays and Training in Recommender Systems. In *Procs. of IFIP INTERACT03: Human-Computer Interaction*, Vol. 3. 176–183.
- [30] Andriy Mnih and Russ R. Salakhutdinov. 2008. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems*. 1257–1264.
- [31] Krishna Prasad Neupane, Ervine Zheng, and Qi Yu. 2021. MetaEDL: Meta Evidential Learning For Uncertainty-Aware Cold-Start Recommendations. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1258–1263.
- [32] Michael P. O’Mahony, Neil J. Hurley, and Guénolé CM Silvestre. 2006. Detecting Noise in Recommender System Databases. In *Procs. of the 11th International Conference on Intelligent User Interfaces*. 109–115.
- [33] Fernando Ortega, Raúl Lara-Cabrera, Ángel González-Prieto, and Jesús Bobadilla. 2021. Providing reliability in Recommender Systems through Bernoulli Matrix Factorization. *Information Sciences* 553 (2021), 110–128.
- [34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Procs. of the NIPS 2017 Workshop on Autodiff*.
- [35] Francisco J Peña, Diarmuid O’Reilly-Morgan, Elias Z Tragos, Neil Hurley, Erika Duriakova, Barry Smyth, and Aonghus Lawlor. 2020. Combining Rating and Review Data by Initializing Latent Factor Models with Topic Models for Top-N Recommendation. In *Procs. of the 14th ACM Conference on Recommender Systems*. 438–443.
- [36] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to the Recommender Systems Handbook. In *Recommender Systems Handbook*. Springer, 1–35.
- [37] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo. In *Procs. of the 25th International Conference on Machine Learning*. 880–887.
- [38] Guy Shani and Asela Gunawardana. 2011. Evaluating Recommendation Systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer, 257–297.
- [39] Bin Wang, Jie Lu, Zheng Yan, Huaishao Luo, Tianrui Li, Yu Zheng, and Guangquan Zhang. 2019. Deep Uncertainty Quantification: A Machine Learning Approach for Weather Forecasting. In *Procs. of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2087–2095.
- [40] Chenxu Wang, Fuli Feng, Yang Zhang, Qifan Wang, Xunhan Hu, and Xiangnan He. 2022. Rethinking Missing Data: Aleatoric Uncertainty-Aware Recommendation. *arXiv preprint arXiv:2209.11679* (2022).
- [41] Chao Wang, Qi Liu, Runze Wu, Enhong Chen, Chuanren Liu, Xunpeng Huang, and Zhenya Huang. 2018. Confidence-Aware Matrix Factorization for Recommender Systems. In *Procs. of the Thirty-Second AAAI Conference on Artificial Intelligence*. 434–442.
- [42] Azene Zenebe and Anthony F. Norcio. 2009. Representation, Similarity Measures and Aggregation Methods Using Fuzzy Sets for Content-Based Recommender Systems. *Fuzzy Sets and Systems* 160, 1 (2009), 76–94.
- [43] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. 2014. Addressing Cold Start in Recommender Systems: A Semi-Supervised Co-Training Algorithm. In *Procs. of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. 73–82.
- [44] Bo Zhu, Fernando Ortega, Jesús Bobadilla, and Abraham Gutiérrez. 2018. Assigning reliability values to recommendations using matrix factorization. *Journal of Computational Science* 26 (2018), 165–177.