A Lightweight Method for Modeling Confidence in Recommendations with Learned Beta Distributions

Norman Knyazev
Radboud University
Nijmegen, The Netherlands
norman.knyazev@ru.nl

Harrie Oosterhuis
Radboud University
Nijmegen, The Netherlands
harrie.oosterhuis@ru.nl

ABSTRACT

Most recommender systems (RecSys) do not provide an indication of confidence in their decisions. Therefore, they do not distinguish between recommendations of which they are certain, and those where they are not. Existing confidence methods for RecSys are either inaccurate heuristics, conceptually complex or computationally very expensive. Consequently, real-world RecSys applications rarely adopt these methods, and thus, provide no confidence insights in their behavior.

In this work, we propose learned beta distributions (LBD) as a simple and practical recommendation method with an explicit measure of confidence. Our main insight is that beta distributions predict user preferences as probability distributions that naturally model confidence on a closed interval, yet can be implemented with the minimal model-complexity. Our results show that LBD maintains competitive accuracy to existing methods while also having a significantly stronger correlation between its accuracy and confidence. Furthermore, LBD has higher performance when applied to a high-precision targeted recommendation task.

Our work thus shows that confidence in RecSys is possible without sacrificing simplicity or accuracy, and without introducing heavy computational complexity. Thereby, we hope it enables better insight into real-world RecSys and opens the door for novel future applications.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Learning to rank*; • **Computing methodologies** → **Uncertainty quantification**.

KEYWORDS

Recommender Systems, Confidence, Uncertainty

ACM Reference Format:
Norman Knyazev and Harrie Oosterhuis. 2023. A Lightweight Method for Modeling Confidence in Recommendations with Learned Beta Distributions. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3604915.3608788

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09.
https://doi.org/10.1145/3604915.3608788

1 INTRODUCTION

In order to best serve their users, recommender systems (RecSys) generally model user preferences to predict which item recommendations they would like [40, 46]. Generally, these models are constructed with collaborative filtering, where to predict the preference of one user-item combination, one considers what similar users thought of that item [29]. The most fundamental of these methods is Matrix Factorization (MF) [26] which represents users and items with embeddings, i.e., numerical vectors, where a high similarity between embeddings leads to a positive preference prediction, and vice versa. Due to its effectiveness and simplicity, MF is still one of the most competitive and widely-used RecSys methods [39].

In spite of being a heavily studied core RecSys problem, the modeling of user preferences remains a very difficult task [40]. Part of this is due to the inherent uncertainty in user preferences: human behavior is complex and stochastic, and therefore, impossible to predict with perfect accuracy [3, 22]. As a result, it is thus inevitable that even the best RecSys models will make errors in their predictions. However, despite these expected errors, the most commonly used RecSys models do not give any sense of uncertainty in their predictions [27, 49]. In other words, these models do not distinguish between predictions with high confidence, of which they are certain, or those with low confidence, of which they are unsure. Consequently, current RecSys give no insight into when recommendations are made with reliable confidence or doubtful guesswork [14, 21].

Naturally, these shortcomings make the idea of confidence modeling for RecSys very appealing. A good measure of confidence could give insight into the behavior of a system, and give an indication of the extent of the error one can expect for a particular prediction. [32, 36]. Furthermore, confidence enables functionalities such as exploration: recognizing which unexplored items have the potential of a high preference [8]; or conservative safety: guaranteeing a certain level of user experience by only following high-confidence preferences [34]. In addition, uncertainty modeling could make RecSys more explainable [51], and enable more insightful studies of user preferences.

In response, previous work proposed RecSys with confidence by applying generative probabilistic models [42, 49]. To the best of our knowledge, all existing work assumes user ratings arise as samples from Gaussian probability distributions, where the ratings predicted by a MF model provide the means of each distribution [33]. Salakhutdinov and Mnih [42] then apply a Bayesian approach and infer a probability distribution over embeddings that captures the uncertainty in the data. But this inference has an impractically heavy computational cost, even when approximated, which seems

to have prevented widespread real-world adoption. As an alternative, Wang et al. [49] use static embeddings but vary the standard deviations of the Gaussians to model different levels of uncertainty. However, this approach is very limited in the confidence patterns it can express, i.e. it only lets uncertainty vary on user and item levels separately and not over specific user-item combinations. The lack of user-item interactions for confidence modeling is similarly a problem for Koren and Sill [27], who instead capture the uncertainty by learning user-specific thresholds to assign each region of a Gaussian-like density to a single rating value. The total density of each region then represents the probability of observing that rating. While this allows confidence to vary, this solution is limited in the patterns it can express and does not appear easily interpretable. As such, it appears the current RecSys field has no method that effectively models uncertainty in an elegant and interpretable fashion, whilst also having practical computational requirements.

In this work, we introduce the learned beta distributions (LBD) as a simple and lightweight RecSys method for a natural and explicit model measure of confidence. Our method transforms a standard beta distribution into a discrete rating distribution, based on only two parameters that can be modeled with the same model-complexity as MF. By utilizing close approximations of their gradients [41, 48], LBDs are optimized without high computational costs, thus making them very practical to work with. Furthermore, LBDs can vary their confidence over different user-item combinations, with enough model expressiveness to capture complex patterns in uncertainty. Our experimental results show LBD has performance competitive with MF and the confidence models of Wang et al. [49] and Koren and Sill [27], while having a considerably stronger correlation between its confidence and its accuracy across ratings.

With the introduction of LBD for RecSys with confidence, we thus show that confidence modeling in RecSys is possible without sacrificing accuracy or adding heavy computational complexity. While our study is limited to MF, the most foundational RecSys model [39], the approach underlying LBD is easily applicable or extendible to more complex RecSys models. For the RecSys field, we hope our practical approach enables better insight into real-world recommendations, and stimulates future work into novel applications of uncertainty models for RecSys.

2 RELATED WORK

Rating prediction is one of the core recommendation tasks in environments with significant amounts of explicit feedback [2, 19, 25, 26]. However, most widely-adopted models, such as MF only focus on point predictions [26, 33, 44, 45]. As a consequence, they are also unable to differentiate between cases where the prediction task is easy and the prediction is likely correct, and those where the error is likely to be higher.

Being able to quantify confidence in individual predictions has many potential uses. Showing confidence scores alongside recommendations is known to increase user satisfaction and platform trust [12, 31, 37]. Mesas and Bellogín [32] find that it may be possible to significantly increase precision by focusing only on the highest confidence predictions. Jeunen and Goethals [15] and Adomavicius et al. [2] use lower confidence bounds instead of estimated relevance values to improve policy evaluation and recommender performance. Conversely, upper confidence bounds can be used to recommend a more diverse range of items or to explore user preferences for items where they are poorly understood [5, 8].

Existing confidence estimation methods can be broadly divided into three categories: heuristic, bootstrapping and generative approaches. The algorithms in the first group extend point predictions with some predefined measures, for instance, the number of interactions for the user or the item [2, 4, 30]. However, whilst computationally light, the use of heuristics may also lead to poor generalization [30, 50]. Mazurowski [30] therefore proposes a bootstrapping approach that trains a recommender multiple times using different slices of the training data. Uncertainty in a given rating is then represented by the variability of the prediction across different models. However, training multiple models also means that this technique has a significant computational cost.

Generative methods instead view individual ratings as samples from a distribution, whose parameters have to be learned. In practice, this distribution is generally assumed to be Gaussian with the same variance parameter for all ratings [28, 33, 42, 49, 52]. However, further steps need to be taken to capture rating variability, as simple generative methods such as Probabilistic Matrix Factorization (PMF) have the same confidence for all ratings [33]. Bayesian Probabilistic Matrix Factorization (BPMF) [42] extends PMF by integrating over all possible user and item vectors via Gibbs sampling [7]. The empirical sampling variance of the embeddings then represents the model's confidence in the correctness of the learned parameters. However, this method of sampling is also expensive and the model thus difficult to apply in practice. Lim and Teh [28] instead use variational methods for quicker convergence, albeit at the cost of accuracy and a worse fit to the data [42].

Our approach is conceptually most similar to the following two approaches: Confidence-Aware Matrix Factorization (CMF) of Wang et al. [49] and OrdRec of Koren and Sill [27]. The former extends PMF by relaxing its assumption of shared variance. Instead, each rating is assumed to be distributed with its own variance that is a product of learned global, user and item terms. The authors also propose an analogous extension to BPMF; however, the computational complexity arising from sampling still remains. On the other hand, OrdRec [27] assumes a fixed-variance logistic distribution to model preference, but also learns a set of varying thresholds for each user. The probability density falling between a pair of neighboring thresholds is then mapped to a single rating value and represents the probability of observing that rating. This allows variance to vary per user, but does so in an oblique and hard to interpret manner. We differ from existing work by relying on beta distributions to capture uncertainty, as we argue that this is a more appropriate choice than Gaussian or logistic distributions to reflect the variation in user ratings.

3 BACKGROUND: THE BETA DISTRIBUTION

The beta distribution is a continuous probability distribution over the closed interval $[0, 1]$ [18]. Because it can easily be extended to any finite interval and can represent a great variety of distributions, the beta distribution has been widely applied to many different settings [10]. Despite this wide applicability, the distribution only has two parameters $\alpha > 0$ and $\beta > 0$. Its probability density function

(PDF) is defined as:

$$f_B(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, \tag{1}$$

where the normalization factor, $B(\alpha,\beta) = \int_0^1 f_B(t;\alpha,\beta)dt$, is the complete regularized beta function that ensures that the PDF integrates to one. The mean of the distribution (μ) is determined by the ratio of α and the sum $\alpha+\beta$, where an α higher than the β means that the mean is closer to 1 than to 0, and vice versa. The variance of the distribution depends on both the sum of the parameters as well as their product:

$$\mathbb{E}[x] = \mu = \frac{\alpha}{\alpha+\beta}, \qquad \mathbb{V}[x] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}. \tag{2}$$

We note that generally the beta distribution is non-symmetric, except when $\alpha=\beta$. For instance, when $\alpha=\beta=1$ it becomes equal to the (symmetric) uniform distribution.

The beta distribution models the likelihood function of the parameter underlying a Bernoulli trial, where $\alpha-1$ is the number of successes and $\beta-1$ the number of failures. For an intuitive understanding, imagine $\alpha+\beta-2$ flips of a biased coin where $\alpha-1$ heads and $\beta-1$ tails are observed. If the coin had a probability x of heads, the probability of the above result would be proportional to $x^{\alpha-1}(1-x)^{\beta-1}$. We see that the PDF of the beta distribution is simply this probability for each x value, but normalized to produce a valid distribution.

Inspired by this intuition, one can alternatively parameterize the beta distribution by its mean $\mu \in (0,1)$ and the sample size $\nu \in (0,\infty)$. This corresponds to $\alpha = \nu\mu$ and $\beta = \nu(1-\mu)$ in the original parameterization, since $\mu = \alpha/(\alpha+\beta)$ and $\nu = \alpha+\beta$. Intuitively for our likelihood example, μ would correspond to the observed mean and ν reflects the number of observations of the Bernoulli trial: thereby, ν can also be seen as a measure of confidence.

The cumulative distribution function (CDF) of the beta distribution is not as straightforward:

$$F_B(x;\alpha,\beta) = I_x(\alpha,\beta) = \frac{\int_0^x f_B(t)dt}{B(\alpha,\beta)}, \tag{3}$$

where $I_x(\alpha,\beta)$ is the incomplete regularized beta function. Whilst no closed form for the CDF exists, it can be represented in terms of infinite power series or continued fractions [9, p. 292, 385-389]. Using these representations, close approximations of the CDF and its derivative w.r.t. α and β can be computed efficiently [41, 48] and implementations of these approximations are already available in popular software packages such as Tensorflow [1] and Jax [6].

Finally, while the beta distribution only covers the closed interval $[0,1]$, it can easily be used to create a distribution over any closed interval. Let $R^{\min}$ and $R^{\max}$ indicate the endpoints of a closed interval for the random variable $x' \in [R^{\min}, R^{\max}]$. We can transform the beta distribution to cover this interval by taking from it a sample $x \sim \text{beta}(\alpha,\beta)$ and transforming it: $x' = x(R^{\max}-R^{\min})+R^{\min}$. Thus for each point x' in the interval $[R^{\min}, R^{\max}]$, there is a corresponding x in $[0,1]$: $x = \frac{x'-R^{\min}}{R^{\max}-R^{\min}}$. As a result, the PDF of the new distribution is simply: $f'_B(x';\alpha,\beta) = f_B(x;\alpha,\beta)/(R^{\max}-R^{\min})$, and its CDF is: $F'_B(x';\alpha,\beta) = F_B(x;\alpha,\beta)$. The mean value undergoes the same transformation: $\mathbb{E}[x'] = \mathbb{E}[x](R^{\max}-R^{\min})+R^{\min}$, and the variance is multiplied by the size of the interval squared: $\mathbb{V}[x'] = \mathbb{V}[x](R^{\max}-R^{\min})^2$. Thereby, the beta distribution is not only useful to model ratios or probabilities, but also for modeling other values that are bound to other closed intervals.

4 METHOD: LEARNED BETA DISTRIBUTIONS FOR RATING PREDICTION

Our goal is to introduce a rating prediction model that quantifies the confidence it has in its predictions without heavy computational costs. In this section, we propose learned beta distributions as an expressive and conceptually simple method for rating prediction with an elegant measure of confidence. The remainder of this section is structured as follows: First, we detail how a beta distribution can be transformed into an optimizable distribution over discrete ratings (Section 4.1). Second, several approaches are proposed to model the parameters of a unique rating distribution for each user-item combination (Section 4.2). Third, two alternatives to modeling user and item biases into the parameters of the model are introduced (Section 4.3). Fourth, static and adaptive binning strategies for discretization are proposed (Section 4.4). Fifth, a summary of the complete approach is laid out, starting from user and item embeddings and ending in discrete rating distributions (Section 4.5). Finally, we compare LBD with the existing MF, CMF and OrdRec approaches in conceptual terms (Section 4.6).

4.1 From a beta distribution to an optimizable discrete rating distribution

In this subsection, we describe how a beta distribution can be transformed to model a discrete rating distribution, and subsequently optimized with a cross-entropy loss.

We assume that users can provide ratings on a discrete and linear scale: $R \in \{R_1, R_2, \ldots, R_n\}$, where n is the number of possible rating values and the difference between R_1 and R_2 is equal to that between R_2 and R_3, etc. To denote the minimum and maximum values, we use $R_1 = R^{\min}$ and $R_n = R^{\max}$, respectively.

Let α_{ij} and β_{ij} be the parameters for the distribution of rating R_{ij}, corresponding with user i and item j. We begin by transforming the associated beta distribution to the interval spanning the range of possible rating values: $[R^{\min}, R^{\max}]$. As discussed in Section 3, a sample from a beta distribution: $x \sim \text{beta}(\alpha_{ij},\beta_{ij})$ can be transformed accordingly: $x' = x(R^{\max}-R^{\min})+R^{\min}$.The resulting distribution over x' has the same shape as the beta distribution, but is transposed and stretched over the rating interval. The PDF is easily mapped back to the original distribution: $f'_B(x';\alpha,\beta) = f_B(x;\alpha,\beta)/(R^{\max}-R^{\min})$ and the same goes for the CDF: $F'_B(x';\alpha,\beta) = F_B(x;\alpha,\beta)$.

While our new distribution is on the correct interval, it still has to be discretized. This can be done straightforwardly by splitting the interval into bins, each representing the probability mass of one rating value. Let the $W_{ij,r} \in [0,1]$ parameter represent the bin width for rating r s.t. $\sum_{r=1}^{n} W_{ij,r} = 1$. The lower edge of rating R_r can then be defined for $r > 1$ as $E_r = E_{r-1} + (R^{\max}-R^{\min})W_{ij,r-1}$, and $E_1 = R^{\min}$ for the first edge. The probability of a rating is then simply the area under the PDF and inside the corresponding bin. We note that an equivalent discretization can be achieved by

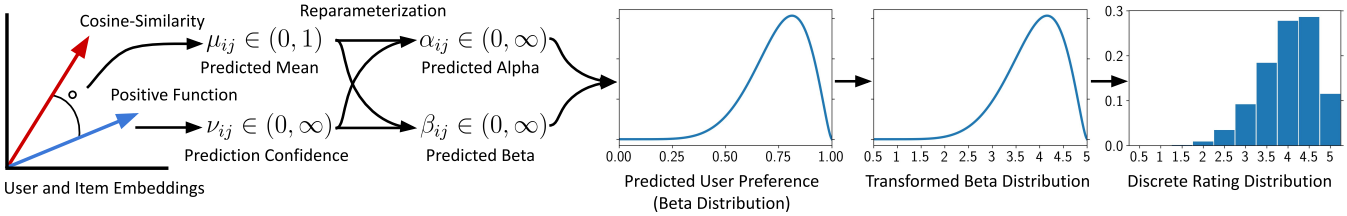

Figure 1: Visualization of the learned beta distributions (LBD) method. LBD starts with user and item embeddings, their cosine similarity determines the predicted mean and a positive function its confidence. These values are reparameterized to the input of a beta distribution, which models a prediction of user preference with explicit uncertainty. This distribution is transformed and discretized to produce the predicted discrete rating distribution. The entire LBD method is differentiable and optimized with a cross-entropy loss.

binning the original beta distribution, the bin edges here are simply transformed inversely: $e_r = (E_r - R^{\min})/(R^{\max} - R^{\min})$. Therefore, it is also straightforward to express the probabilities in terms of the CDF of the underlying beta distribution:

$$\begin{aligned} P(R_r; \alpha_{ij}, \beta_{ij}) &= F'_B(E_{r+1}; \alpha_{ij}, \beta_{ij}) - F'_B(E_r; \alpha_{ij}, \beta_{ij}) \\ &= F_B(e_{r+1}; \alpha_{ij}, \beta_{ij}) - F_B(e_r; \alpha_{ij}, \beta_{ij}). \end{aligned} \tag{4}$$

Importantly, our transformation from the beta CDF (F_B) to the rating probability $P(R_r; \alpha_{ij}, \beta_{ij})$ is completely differentiable. Furthermore, the gradients w.r.t. α_{ij} and β_{ij} can be computed using the close approximations of previous work [41, 48]. In other words, our discrete rating probability is fully differentiable to its input parameters. As a result, we can easily use it to optimize a cross-entropy loss:

$$\mathcal{L}(\alpha, \beta) = \frac{1}{Z} \sum_i \sum_j \log P(R_{ij}; \alpha_{ij}, \beta_{ij}). \tag{5}$$

Thus, we have shown that beta distributions can be transformed into discrete rating distributions in a straightforward and elegant manner. Since the result is fully differentiable, it is easily optimizable; in particular, we propose to maximize the log-likelihood of observed rating data by minimizing a cross-entropy loss.

4.2 Modeling means and confidences

We have introduced a method for producing a discrete rating distribution for a rating R_{ij} from parameters α_{ij}, β_{ij} and W_{ij}. In this subsection, we propose a simple way of modeling α_{ij} and β_{ij} based on the interactions between user and item embeddings.

To start, we make use of the alternative parameterization described in Section 3 and introduce the mean $\mu_{ij} \in (0, 1)$ and confidence $\nu_{ij} \in (0, \infty)$. These map back to the original parameterization through: $\alpha_{ij} = \mu_{ij}\nu_{ij}$ and $\beta_{ij} = (1 - \mu_{ij})\nu_{ij}$. Furthermore, inspired by the simplicity of MF, we introduce a non-zero learned embedding for each user $U_i \in \mathbb{R}^D$ and for each item $V_j \in \mathbb{R}^D$. Our first insight is that because $\mu_{ij} \in (0, 1)$, it can be modeled elegantly with the cosine similarity:

$$\mu_{ij} = \frac{1}{2} + \frac{1}{2}\text{Cosine-similarity}(U_i, V_j) = \frac{1}{2} + \frac{1}{2}\frac{U_i^\intercal V_j}{||U_i||\,||V_j||}. \tag{6}$$

Thereby, a higher similarity between the directions of the embeddings results in a higher mean of the predicted rating distribution. Our second insight is that any differentiable transformation from the embeddings to a positive scalar can be used for the confidence parameter. Whilst many transformations are possible, we propose the following three:

$$\nu_{ij}^{\text{norm}} = ||U_i||\,||V_j||, \quad \nu_{ij}^{\text{sum}} = ||U_i + V_j||, \quad \nu_{ij}^{\text{dot}} = |U_i^\intercal V_j|. \tag{7}$$

With ν_{ij}^{norm}, the confidence parameter is equal to the product of the L_2 norms of the embeddings. Thus users or items with larger embeddings will result in more confident predictions. Alternatively, one could interpret this as each user and item having a unique confidence term, expressed in the distance of their embedding vector from the origin. The simplicity of ν_{ij}^{norm} makes it easy to understand, but it therefore also lacks expressiveness and cannot capture complex patterns that vary over specific user-item interactions.

The L_2 norm of the summation of the embeddings is used to model confidence in ν_{ij}^{sum}. Here both the direction and the distance of the embeddings determine the resulting confidence. As a result, this model can capture more complex interactions between users and items, but is also intuitively more difficult to understand.

Lastly, ν_{ij}^{dot} uses the absolute value of the dot product of the embeddings. Thus, confidence is also determined by both the direction and the distance of the embedding vectors. However, because the absolute value is used, both high similarity and dissimilarity result in high confidence predictions. The dot product highly correlates with the cosine similarity, which could also mean there is a high correlation between μ_{ij} and ν_{ij}^{dot}. It is unclear whether such a correlation is desirable in practice.

To summarize, we have introduced a novel approach to modeling a beta distribution by basing its input parameters on the interaction between user and item embeddings. In particular, we propose to model mean ratings by cosine similarity and the confidence by one of several possible transformations. Thereby, our approach has the same model-complexity as MF, since both only use user and item embeddings. But whereas MF only uses them to model pointwise ratings, LBD models predictions as probability distributions.

4.3 Modeling user and item biases

An important feature of standard MF models is that they can also include bias weights per user and item [17, 24, 26]. These bias weights can increase all the ratings of a user or an item, for instance, to capture that certain users tend to give higher ratings than the average user. Because adding these weights is very effective for MF, we propose adding them to our LBD model as well.

The addition of bias weights to the beta distribution is straightforward. We introduce the optimizable scalars $a \in \mathbb{R}$ and $b \in \mathbb{R}$, and denote a_0 for a global weight, a_i for a user weight and a_j for an item weight, and the analogous weights b_0, b_i and b_j. These bias weights are then simply added to the weights predicted with the embeddings:

$$\alpha'_{ij} = \max(a_0 + a_i + a_j + \alpha_{ij}, \epsilon), \quad \beta'_{ij} = \max(b_0 + b_i + b_j + \beta_{ij}, \epsilon), \quad (8)$$

where ϵ is a very small positive quantity to ensure that $\alpha'_{ij} > 0$ and $\beta'_{ij} > 0$.

While the addition may appear simple, it also has a Bayesian interpretation, as the normalized product of beta distributions can also be produced by summing their parameters accordingly:

$$f_B(a_0 + 1, b_0 + 1)\, f_B(a_i + 1, b_i + 1)\, f_B(a_j + 1, b_j + 1)\, f_B(\alpha_{ij}, \beta_{ij}) \propto f_B(a_0 + a_i + a_j + \alpha_{ij}, b_0 + b_i + b_j + \beta_{ij}). \quad (9)$$

Therefore, we can interpret $f_B(a_0 + 1, b_0 + 1)$ as a global prior distribution, $f_B(a_i + 1, b_i + 1)$ as the likelihood function given that user i is involved, the same for $f_B(a_j + 1, b_j + 1)$ and item j, and finally, $f_B(\alpha_{ij}, \beta_{ij})$ as the likelihood function for the interaction between i and j. While this Bayesian interpretation is valid, we note that our optimization only considers the final distribution: therefore, it does not directly optimize the accuracy of the prior and likelihood decomposition.

Additionally, we also propose bias weights in terms of the alternative parameterization. We introduce the optimizable scalars $u \in (0, 1)$ and $v \in (0, \infty)$, where u_0 and v_0 represent global weights and u_i, u_j, v_i and v_j user and item specific weights. The weights v are used as multipliers on the confidences ν: thus higher v result in more confident predictions, and vice versa. For the u weights, we create a piecewise linear mapping that places $\mu'_{ij} = 0.5$ when $\mu_{ij} = u_0 u_i u_j$: thus high u values result in a lower predicted mean, and vice versa. Formally, the bias weights are applied as follows:

$$\nu'_{ij} = v_0\, v_i\, v_j\, \nu_{ij}, \quad \mu'_{ij} = \begin{cases} \dfrac{\mu_{ij}}{2\, u_0\, u_i\, u_j} & \text{if } \mu_{ij} \geq u_0\, u_i\, u_j, \\ \dfrac{1}{2} + \dfrac{\mu_{ij} - u_0\, u_i\, u_j}{2(1 - u_0\, u_i\, u_j)} & \text{otherwise.} \end{cases} \quad (10)$$

Since our proposed applications of bias weights are fully differentiable, they effortlessly integrate into our LBD model.

4.4 Modeling static and adaptive discretization strategies

We have described how user and item embeddings and biases can model the parameters of a beta distribution. As discussed in Section 4.1, LBD creates a discrete rating distribution through binning according to the bin width parameters W. We propose a static binning strategy for modeling W: LBD-S; and an adaptive binning strategy: LBD-A.

The static strategy of LBD-S straightforwardly makes each bin equisized: $\forall i, j, r : W_{ij,r} = \frac{1}{n}$. In terms of model-complexity, this is the most efficient choice as none of the W parameters have to be learned. Implicitly this strategy assumes a linear relation between preference and ratings, i.e., the uniform distribution with $\alpha = 1$ and $\beta = 1$ will translate to a uniform discrete distribution. However, this assumption may not be correct, e.g., in the Movielens dataset [11], the whole ratings ($\{1, 2, 3, 4, 5\}$) are much more frequent than the half ratings ($\{0.5, 1.5, 2.5, 3.5, 4.5\}$). This difference in frequency seems to be an artifact of the rating interface, which leads users to choose whole-number ratings over the others, and is unlikely to reflect an actual clustering in the underlying preferences [16, 35]. Importantly, such patterns are difficult to capture for LBD-S, since it assumes that probability mass is smoothly spread over all rating values.

In contrast, the adaptive binning strategy varies the bin widths per rating, user and item, and thereby, it can better capture frequency differences between certain rating values. Inspired by the discretization of Koren and Sill [27], we introduce the unnormalized bin widths $w_{ij,r} = \exp(\theta^r_i + \theta^r_j)$ with the optimizable parameters θ^r_i and θ^r_j for each user i, item j and rating value r. The bin widths are then simply obtained by normalizing: $W_{ij,r} = w_{ij,r} / \sum_{r'=1}^{n} w_{ij,r'}$. By increasing the bin width of certain rating values relative to others, these rating values can become more probable without changing the underlying beta distribution. Thereby, LBD-A can capture non-smooth patterns that LBD-S cannot, e.g., that half ratings are less probable than whole ratings, or whether a certain user is more likely to translate a specific level of preference to a higher rating than most other users [16]. Importantly, the increased expressiveness of LBD-A over LBD-S comes at the cost of only a few additional parameters for each user and item.

4.5 Overview of the LBD method

Now that all the components of the learned beta distributions (LBD) method have been introduced, we summarize how these components come together to form a single approach. The visualization in Figure 1 accompanies this subsection.

Similar to MF, LBD has an optimizable embedding vector for each user and item, the cosine similarity between these vectors determines the mean prediction (μ), and a positive function is used to get a positive scalar indicating the confidence in the prediction (ν). The mean and confidence are then mapped to α and β values and bias terms are also added. The resulting values serve as input parameters to a beta distribution. This beta distribution is the prediction of user preferences with explicit uncertainty; preferences are modeled as ratios in the interval $[0, 1]$ and the beta PDF (f_B) indicates how probable each possible value is predicted to be. This distribution is then transformed to match the range of possible rating values and discretized to give exact probabilities per rating. Discretization can be done either using equally sized bins or by adjusting each region's width $w_{ij,r}$ based on the user and item terms (θ). Because each step between the embeddings and the final discrete rating distribution is differentiable, one can compute the gradient of a rating probability w.r.t. the underlying embeddings. We utilize this differentiability to optimize the model with a cross-entropy loss, so that the predicted discrete rating probabilities maximize the likelihood of observed rating data.

Thereby, our LBD method predicts user ratings with explicit uncertainty in its predictions. Importantly, it does this with the same model-complexity as MF, and unlike previous methods that utilize Gaussian distributions [42, 49], LBD can have non-symmetric variance that can capture user-item interactions, never predicts ratings outside of the valid rating range, can be described using just

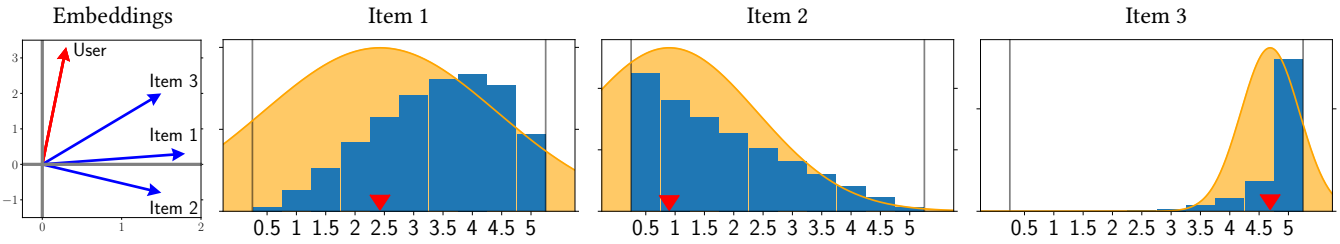

Figure 2: Example comparison of how MF, CMF and LBD make predictions for a user and three different items. The prediction of MF is pointwise (red triangle); that of CMF is a Gaussian distribution (orange continuous probability density function); and that of LBD is a discrete rating prediction based on an underlying beta distribution (blue discrete probability mass function).

two parameters and does not involve heavy computational costs in its optimization and prediction.

4.6 Comparison with existing methods

Before LBD is compared with the existing MF [26], CMF [49] and OrdRec [27] methods in experimental terms in Section 6, we first consider key conceptual differences between the methods. To accompany the discussion, Figure 2 visualizes the predictions of MF, CMF and LBD for an example scenario with one user and three items.

To start, MF differs from CMF, OrdRec and LBD in that its pointwise predictions are without any sense of confidence, and accordingly, they are presented as points on the x-axis. As a result, while CMF and LBD show high uncertainty about their predictions for item 1 and 2, MF gives no indication that it may be less reliable there.

Next, we compare CMF and LBD, as both give predictions in the form of a distribution: CMF as a Gaussian distribution and LBD as a discrete probability distribution based on an underlying beta distribution. We argue that the usage of a beta distribution by LBD provides three key advantages over Gaussians for rating prediction: (i) Beta distributions can be bounded to only give probability inside the possible range of ratings, whereas Gaussians are unbounded. This problem is illustrated across all items in Figure 2 as the PDF of CMF gives much probability outside the valid range of ratings. (ii) Beta distributions can have non-symmetric variance, while Gaussians cannot. For instance, LBD thinks the lowest rating of item 2 is most probable, and puts the remainder of its probability on the higher ratings. In contrast, CMF also thinks a low rating is most likely, but has to give equal probability to ratings below and above it, including ratings out of the valid range. (iii) Lastly, LBD can be fully parameterized with minimal model-complexity, since it only requires an interaction between two embeddings to compute all its parameters. Conversely, this is not possible for a Gaussian because its mean is unbounded, and as a result, CMF needs additional parameters for its confidence model (not visualized in Figure 2).

Finally, our visualization in Figure 2 does not include OrdRec because its differentiating characteristics are difficult to display. OrdRec uses the logistic distribution, which is very similar to the Gaussian distribution but with heavier tails. However, unlike CMF and similar to LBD, it also applies discretization through the same adaptive binning strategy as described in Section 4.4. Thus unlike CMF, OrdRec can transform the shape of its rating distribution through adaptive binning, enabling it to have asymmetric variance. Nevertheless, the adaptive binning strategy is relatively limited in its expressiveness (cf. $w_{ij,r}$ and v_{ij}). Furthermore, it is also unclear whether the symmetric logistic function, with its infinite coverage and heavy tails, is appropriate to model user preferences. In contrast, LBD captures both a measure of preference and confidence within its embeddings, and can already express asymmetric preference distributions before discretization is applied. The remainder of this paper will investigate whether the theoretical advantages of LBD can be verified experimentally and lead to more effective modeling of uncertainty.

5 EXPERIMENTAL SETUP

The experiments carried out for this paper are aimed at answering the following four research questions:

- **RQ1** What modeling of the parameters of LBD results in the highest predictive performance for rating prediction?
- **RQ2** Does our LBD approach provide rating prediction performance that is competitive with MF, CMF and OrdRec?
- **RQ3** Is there a stronger correlation between the confidence and accuracy of predictions by LBD than for CMF and OrdRec?
- **RQ4** Does the confidence modeling of LBD translate to higher performance compared to MF, CMF and OrdRec in a high-precision targeted recommendation task?

Correspondingly, the goals behind these research question are: (i) to find out what version of LBD is the most accurate at rating prediction; (ii) to evaluate whether the confidence modeling of LBD sacrifices competitive accuracy; (iii) to verify whether LBD has a better indication of confidence than the existing confidence-aware methods; and (iv) to investigate how the confidence modeling of LBD could be beneficially leveraged in a practical downstream task.

5.1 Dataset

We base our experiments on the MovieLens 10M dataset [11], where ratings from 69878 users are spread over 10677 items on a 0.5 point scale between 0.5 and 5. In order to perform cross-validation, we randomly split the user ratings into 10 folds. Each experiment is repeated 10 times, so that each fold is used for evaluation once, while the remaining 9 folds are further split into 95% train and 5% validation data. To avoid irrelevant cold-start problems in evaluation, we further ensure that the evaluation set never contains ratings involving users or items that are not in the training set.

5.2 Baselines, metrics and parameter tuning

We compare our LBD method with three baselines of similar model-complexity: MF [26, 33], CMF [49] and OrdRec [27]. MF is a foundational RecSys method and remains a very strong rating prediction baseline when tuned well [38, 39]. Furthermore, it does not have any confidence modeling, and thereby, allows us to evaluate whether the confidence modeling of LBD prevents it from providing competitive rating prediction accuracy. CMF performs confidence modeling through Gaussian distributions with varying variance. Lastly, OrdRec models confidence by partitioning a logistic distribution into non-overlapping regions, each representing the probability mass assigned to a possible rating value.

For the comparison with the baselines, we use the two best LBD parameterizations identified for **RQ1**. Both use α and β bias terms with embeddings of size 512 to jointly model preference and confidence, where the latter is represented via v^{sum}. We consider both static binning (LBD-S) and adaptive binning (LBD-A).

To evaluate various aspects of recommendation performance, we use a variety of metrics for regression (RMSE, MAE), classification (accuracy, average log-likelihood) and ranking (NDCG@3, NDCG@10). For classification, we count a predicted rating to be correct if it is closer to the true rating than to any other value in $\{0.5, 1, \ldots, 5\}$. For the ranking task, we use models' predicted ratings to rerank each user's test set items with known ratings, taking the average over all users. Item relevance is *not* binarized as we aim to achieve the best ordering across all ranks. For regression and ranking, predictions are made using each model's estimate of the rating mean, whilst accuracy is evaluated using the mode of the distribution. Although different point estimates, e.g., quantiles, may be useful for other tasks, we observed that the above estimates resulted in the highest performance for all models.

The hyperparameters of each model were determined by an extensive random search. All models were trained with embedding sizes: 32, 64, 128, 256 and 512, for up to 50 epochs with minibatch size of 8192 using the Adam optimizer [20] with learning rates in the range $[10^{-6}, 10^{-2}]$ and early stopping on validation RMSE with a patience of 10 and a $5 \cdot 10^{-4}$ tolerance. We also applied L_2 regularization on either all embeddings equally or proportional to their frequency in the training data. The final parameters per model were chosen according to the best achieved RMSE on the validation set of the first data fold. MF's embeddings were used to initialize CMF [49]. Furthermore, [27] only evaluate a version of their model whose adaptive binning is determined solely on the user (OrdRec-U). We also include OrdRec with binning based on both the user and item via the exponent from Section 4.4 (OrdRec-UI). And while the authors allow for any model to represent the logistic mean, we use OrdRec on top of MF to allow for a direct comparison with other confidence models.

To evaluate the effectiveness of the confidence modeling, we measure the correlation between confidence and prediction error using both linear correlation (Pearson's r) as well as rank correlation (Kendall's τ). The former reflects the association between specific error and confidence values, whereas the latter quantifies the effectiveness of using model confidence to separate low and high-error predictions. Furthermore, we evaluate the relationship qualitatively by clustering interactions into 1000 bins based on the predicted variance of individual ratings. We use rating variance as it is a confidence measure available for all models, and group interactions using equally spaced bins to identify the relationship between accuracy and specific confidence values. However, due to this grouping strategy, we may also observe patterns for a small number of interactions spread over many bins that do not generalize to the data as a whole. Therefore, we also group interactions into bins containing the same number of interactions of monotonically increasing variance, allowing us to evaluate the general direction of the association but not for specific confidence values.

5.3 High-precision targeted recommendation

Finally, inspired by work on prediction confidence in other domains [12, 30, 32], we want to evaluate whether the confidence modeling of LBD can be effectively exploited in downstream tasks.

We simulate a high-precision task where only a limited number of targeted users are provided with recommendations. Specifically, each method is only allowed to choose N different users, for each of whom it makes one personalized item recommendation, while the remaining users receive no recommendations. For evaluation purposes, the methods may only choose user-item combinations that are present in the test set, so that we can evaluate performance based on the number of successful recommendations, which are taken to be those with rating of at least 4.5. For each model, we report the proportion of users who received a successful recommendation (Precision@1) over different values of N, as well as the relative improvement over the confidence-unaware MF.

This task is analogous to a notification setting where recommendations are mildly intrusive interruptions. For the best user experience in this setting, only highly relevant recommendations should be made and the number of recommendations per user should be limited. We expect that confidence modeling allows for a higher precision in selecting highly relevant items, since it can better identify which high rating predictions are most likely correct.

6 RESULTS AND DISCUSSION

6.1 Parameterization of the LBD model

We begin with addressing the first research question (**RQ1**) by investigating *what method of modeling the LBD parameters results in the highest predictive performance.* To that end, we consider Table 1 which shows the average performance of various LBD configurations over 10 test folds. The configurations are combinations of different confidence functions and (α, β) and (ν, μ) biases proposed in Sections 4.2 and 4.3. In addition, we also evaluate a model with no bias terms, a model that uses separate embeddings to model preference and confidence (denoted with 256, 256) and LBD-A with adaptively discretized bins (denoted by 512 + 10, for the ten additional learned binning parameters).

We see that the choice of parameterization can have a substantial impact on performance: overall, using v^{sum} with α, β biases alongside adaptive bins leads to best performance, while using v^{dot} is clearly the worst choice. Interestingly, the above model with adaptive bins shows the best performance across all non-ranking metrics, as well as second best performance on ranking, with particularly salient improvements in terms of accuracy and log-likelihood. Moreover, the same model but without dynamic bins achieves the

Table 1: Performance in terms of regression, classification and ranking metrics for various LBD configurations (see Sections 4.2-4.4). Reported results are averages from 10-fold cross-validation, with standard deviations in parentheses. Two embedding sizes denote separate embeddings for preference and confidence. 512 + 10 **denotes LBD with the adaptive binning strategy applied (LBD-A). Highest performance per metric is denoted in bold.**

Size	$\nu^{(\cdot)}$	Bias	RMSE	MAE	Accuracy	Ave. Log-L.	NDCG@3	NDCG@10
512	*sum*	-	0.7932 (0.0005)	0.6130 (0.0004)	0.3016 (0.0005)	-1.769 (0.001)	0.9330 (0.004)	0.9568 (0.0002)
512	*sum*	μ, ν	0.7864 (0.0005)	0.6058 (0.0003)	0.3092 (0.0003)	-1.760 (0.001)	0.9339 (0.0003)	0.9568 (0.0001)
512	*sum*	α, β	0.7761 (0.0005)	0.5895 (0.0005)	0.3139 (0.0003)	-1.755 (0.003)	**0.9357** (0.0003)	**0.9583** (0.0001)
512	*norm*	α, β	0.7852 (0.0007)	0.5942 (0.0005)	0.3140 (0.0005)	-1.818 (0.004)	0.9336 (0.0002)	0.9572 (0.0002)
512	*dot*	α, β	0.8387 (0.0005)	0.6391 (0.0006)	0.2972 (0.0005)	-1.865 (0.005)	0.9330 (0.0004)	0.9495 (0.0001)
256, 256	*sum*	α, β	0.7850 (0.0006)	0.5964 (0.0006)	0.3133 (0.0006)	-1.774 (0.002)	0.9331 (0.0003)	0.9570 (0.0001)
512 + 10	*sum*	α, β	**0.7759** (0.0004)	**0.5875** (0.0004)	**0.4356** (0.0006)	**-1.432** (0.002)	0.9351 (0.0003)	0.9579 (0.0001)

Table 2: Model performance on regression, classification and ranking tasks. Results are averages from 10-fold cross-validation, with standard deviations in parentheses. Highest performance per metric is denoted in bold. Significant improvement of an LBD model over *all* baseline models is denoted by $\triangle$, significant improvement of a baseline over *any* LBD version is denoted by $\triangledown$ (separate one-sided Wilcoxon signed-rank tests with matched folds, $p < 0.001$).

Model	RMSE	MAE	Accuracy	Ave. Log-L.	NDCG@3	NDCG@10
MF (128)	0.7802 (0.0006)	0.5984 (0.0005)	0.2923 (0.0005)	-2.004 (0.002)	0.9334 (0.0002)	0.9570 (0.0001)
MF (512)	0.7883 (0.0006)	0.6022 (0.0005)	0.2934 (0.0004)	-2.022 (0.002)	0.9321 (0.0002)	0.9560 (0.0001)
CMF (128)	0.7760 (0.0007)	0.5936 (0.0004)	0.3226 (0.0004) $^{\triangledown}$	-1.777 (0.001)	0.9338 (0.0002)	0.9573 (0.0001)
CMF (512)	0.7820 (0.0005)	0.5967 (0.0003)	0.2849 (0.0015)	-1.801 (0.001)	0.9329 (0.0003)	0.9567 (0.0001)
OrdRec-U (512)	0.7821 (0.0006)	0.6043 (0.0004)	0.2322 (0.0007)	-1.881 (0.001)	0.9343 (0.0003)	0.9575 (0.0001)
OrdRec-UI (512)	0.7765 (0.0006)	0.5896 (0.0005)	0.4187 (0.0006) $^{\triangledown}$	-1.569 (0.001) $^{\triangledown}$	0.9349 (0.0003)	0.9578 (0.0001)
LBD-S (512)	0.7761 (0.0005)	0.5895 (0.0005)	0.3139 (0.0003)	-1.755 (0.003)	**0.9357** (0.0003) $^{\triangle}$	**0.9583** (0.0001) $^{\triangle}$
LBD-A (512)	**0.7759** (0.0004)	**0.5875** (0.0004) $^{\triangle}$	**0.4356** (0.0006) $^{\triangle}$	**-1.432** (0.002) $^{\triangle}$	0.9351 (0.0003)	0.9579 (0.0001)

best performance for ranking metrics as well as the second highest performance on regression metrics and log-likelihood. This thus suggests that these two parameterizations are best suited to rating prediction, regardless of the specific task.

Based on the observed performance differences, it appears that the adaptive binning allows the model to better capture the rating distribution, as evidenced by the associated size of the improvement in the fit to the rating data. We also hypothesize that ν^{sum} is better at confidence modeling than ν^{norm} due to its ability to capture interactions between specific users and items. On the other hand, we speculate that the dot product performs the worst due to its high correlation with cosine similarity, which could in turn produce a high correlation between predicted means and corresponding confidences. Furthermore, our results also indicate that using separate embeddings to model means and confidences does not boost performance, suggesting that a single set of user and item embeddings are enough to capture both preference and confidence patterns. Lastly, it is also clear that including bias terms can lead to better performance across all metrics, as expected from their popularity for MF-based models. However, we are unsure why bias weights in α, β terms are more effective than in μ, ν terms.

In conclusion, to answer **RQ1**: we find that the parameterization of LBD can have a substantial effect on its performance. *Our results indicate that the confidence function ν^{sum} with α and β bias terms, both with and without dynamic binning, lead to the highest overall performance across all our metrics.*

6.2 Recommendation performance

Now that we have established the best configurations of the LBD model, we can address our second research question (**RQ2**) and consider *whether the predictive performance of LBD is competitive with that of MF, CMF and OrdRec.* For this question, we use the results in Table 2 which show the performance of all models with different embedding sizes over a variety of performance metrics.

Overall, we generally see no enormous variation in the performance of the models, as most are in a similar range. Whilst LBD-S achieves lower accuracy compared to CMF and OrdRec-UI, its regression performance is comparable to that of the above models, and its ranking performance is significantly better than for any other model. Furthermore, LBD-A, whilst showing a smaller improvement on NDCG, actually achieves the best performance across all other metrics, significantly improving over the baselines in MAE as well as both classification metrics. Importantly, the above results thus show that LBD has performance that is competitive with MF, CMF and OrdRec.

As such, we answer **RQ2** in the affirmative: *the performance of LBD is competitive with MF, CMF and OrdRec across all our measured metrics.* Furthermore, it appears that LBD could even provide small significant improvements for most metrics. We note that our aim is not to improve these performance metrics, but to evaluate whether confidence modeling has a negative effect on predictive performance. These conclusions thus strongly imply that confidence modeling with LBD allows us to simultaneously model both

Table 3: Correlations between the predicted variance and MAE of baselines and LBD. Results are averages from 10-fold cross-validation, with standard deviations in parentheses. Highest correlation per type is denoted in bold. Significant improvement of LBD over *all* baselines is denoted by Δ (separate one-sided Wilcoxon signed-rank tests with matched folds, $p < 0.001$).

	CMF	OrdRec-U	OrdRec-UI	LBD-S	LBD-A
Pearson's r (Linear Correlation)	0.079 (0.001)	0.157 (0.001)	0.175 (0.001)	0.304 (0.002)$^{\Delta}$	**0.329** (0.002)$^{\Delta}$
Kendall's τ (Rank Correlation)	0.043 (0.002)	0.094 (0.001)	0.112 (0.001)	0.194 (0.001)$^{\Delta}$	**0.216** (0.001)$^{\Delta}$

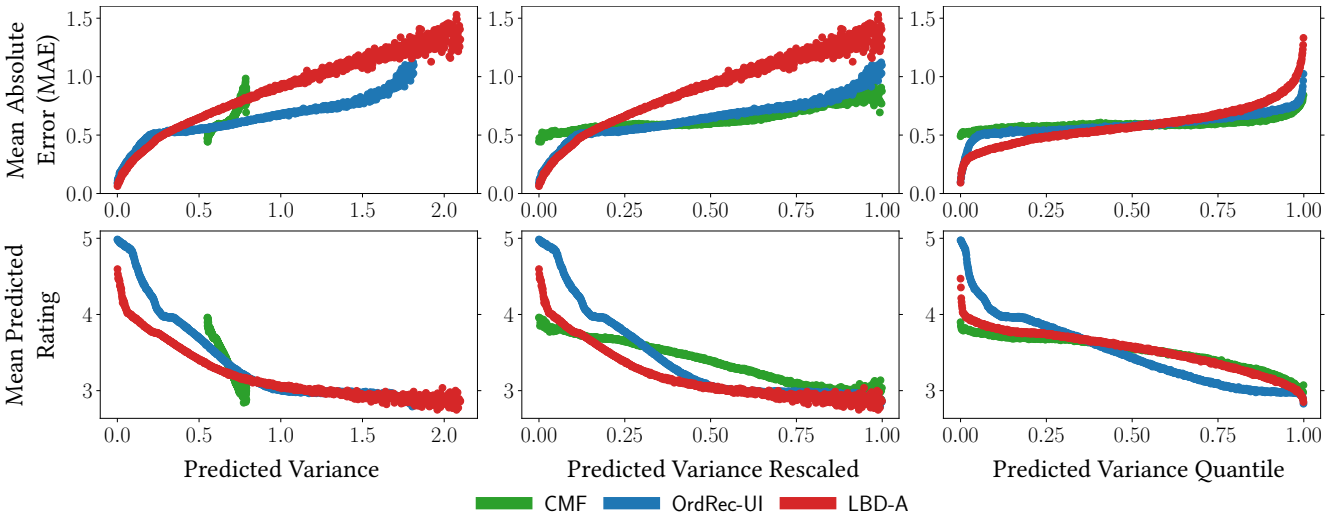

Figure 3: Variation of mean absolute error (MAE) (top) and average predicted rating values (bottom) for different values of predicted variance. Interactions grouped by the predicted variance of each model into 1000 clusters, discarding 0.1% of highest predicted variance outliers. Points represent mean response value of each cluster. Left: interactions grouped into equispaced bins spanning each model's observed predicted variance. Center: grouping as on the left, but variance rescaled to [0,1]. Right: each bin contains 0.1% of interactions, ordered by predicted variance.

preference and confidence without sacrificing predictive performance, at least not in terms of our regression, classification and ranking metrics.

6.3 Evaluating prediction confidence

Now that we have investigated the predictive performance of LBD, we turn our attention to its key feature —its confidence modeling— and address the third research question (**RQ3**) by evaluating *whether there is a stronger correlation between the confidence and accuracy of predictions for LBD than for CMF and OrdRec*. We note that no comparison with MF can be made, since its pointwise predictions do not include any indication of confidence, uncertainty or variance. Table 3 shows the strength of linear and rank correlation between MAE and predicted variance, representing the association between the prediction errors and confidence. We use predicted variance to quantify confidence, since low variance denotes a prediction with low uncertainty and thus high confidence, and vice versa.

From Table 3, it is immediately clear that there are substantial differences across models in terms of both types of correlation. Whilst OrdRec-UI's correlation is the highest among the baselines, LBD-S and LBD-A demonstrate by far the strongest association between their predicted variance and the size of the error. In fact, compared to OrdRec-UI, both LBD models demonstrate a relative increase in both correlation coefficients of at least 70%, with LBD-A's correlation being almost twice as strong as that of OrdRec. That LBD-S achives such a large improvement over OrdRec-UI indicates that modeling user preferences with just a beta distribution and no adaptive binning better captures uncertainty in user ratings compared to the non-linear mapping of OrdRec. Overall, the above results thus suggest that LBD is considerably better at separating low-error predictions from those with a higher error.

For a more qualitative understanding of the above differences, Figure 3 shows how MAE and the average predicted rating vary over interactions grouped by predicted variance. Furthermore, as the predicted variance of distinct models lies over different ranges (see left column of Figure 3), we also show the correlation with predicted variance rescaled to the range $[0, 1]$ (center column), and variance quantiles (right column). We omit OrdRec-U and LBD-S from the visualization to reduce visual clutter.

We start by considering the top-left of Figure 3, which shows how the predicted variance of CMF, LBD-A and OrdRec-UI correlates with their accuracy, with higher variance corresponding to greater MAE for all. Rescaling each model's variance for a more direct comparison, as done in the top-center of Figure 3, shows that there is a larger difference in terms of MAE between the lowest and highest variance predictions of LBD-A and OrdRec-UI compared to CMF. This suggests that LBD-A and OrdRec-UI are more efficient at separating high and low-error predictions.

Table 4: Model performance on the high-precision targeted recommendation task for distinct numbers of targeted users (N, half-powers of 10). Results are Precision@1 averages from 10-fold cross-validation, with standard deviations in parentheses. Highest performance for each value of N is denoted in bold. Significant improvement of an LBD model over *all* baseline models is denoted by $^{\Delta}$, no significant improvement of any of the baselines over either LBD model (separate one-sided Wilcoxon signed-rank tests with matched folds, $p < 0.001$).

Model	Precision@1 for top N users					
	$N = 100$	$N = 320$	$N = 1000$	$N = 3200$	$N = 10000$	$N = 32000$
MF	0.885 (0.034)	0.893 (0.019)	0.889 (0.011)	0.886 (0.004)	0.841 (0.004)	0.726 (0.002)
CMF	0.913 (0.030)	0.907 (0.016)	0.904 (0.013)	0.895 (0.005)	0.853 (0.005)	0.733 (0.002)
OrdRec-UI	0.962 (0.016)	0.949 (0.007)	0.932 (0.006)	0.910 (0.003)	0.863 (0.003)	0.747 (0.002)
LBD-S	0.947 (0.019)	0.944 (0.013)	0.932 (0.009)	0.908 (0.006)	0.861 (0.004)	0.747 (0.003)
LBD-A	**0.971** (0.012)	**0.962** (0.011)$^{\Delta}$	**0.949** (0.005)$^{\Delta}$	**0.925** (0.004)$^{\Delta}$	**0.877** (0.004)$^{\Delta}$	**0.751** (0.003)$^{\Delta}$

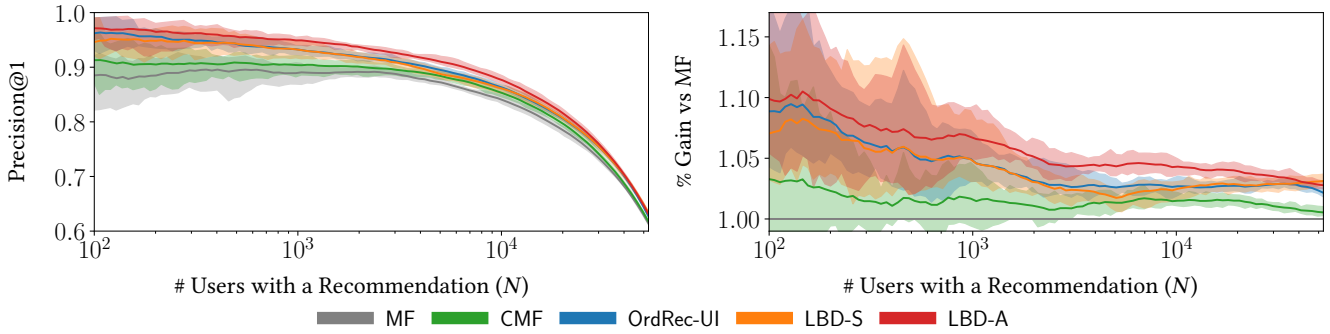

Figure 4: Model performance on the high-precision targeted recommendation task for distinct numbers of targeted users (N) with MF, CMF, OrdRec and LBD methods. Left: Percentage of users for whom a 4.5+ star recommendation is made (Precision@1). Right: Relative Precision@1 improvement over Matrix Factorization. Results are averages over 10 folds. Shaded regions denote minimum and maximum across folds.

Interestingly, from top-center of Figure 3, one might also (albeit erroneously) conclude that, because MAE of OrdRec-UI and LBD-A is comparable when their predicted variance is low and otherwise generally lower for OrdRec-UI, that must mean that OrdRec-UI's confidence modeling is better. We can see that this is however not true by inspecting the top-right of Figure 3, which presents the same results but with predictions ordered in quantiles of variance. There we see that using the predicted variance of LBD-A achieves a better separation between high and low MAE predictions when compared to OrdRec-UI (and CMF). OrdRec does appear to achieve a similar error to LBD-A on the interactions with the very lowest predicted variance, whilst also having the highest error in the right tail. However, the majority of the interactions, which are in the center, all have only minor differences in their MAE. LBD-A, on the other hand, whilst being particularly efficient in the tails, is also better at separating all of the other interactions. As such, for a comparable level of predictive performance (Section 6.2), LBD is better at predicting when it is more or less accurate, i.e. it has a substantially better model of confidence.

To get further insight into the different confidence models, the bottom row of Figure 3 also displays how the average predicted rating varies over the predicted variances. We see that all three models have more confidence in high value ratings, with this trend being the strongest for OrdRec. Potentially, this is an indication that it is easier to predict the positive preferences of a user than their neutral or negative preference. However, for the LBD model, we speculate that the usage of the v^{sum} confidence function could also produce this trend. It assigns higher confidence when embeddings have a similar direction, and similar directions also result in higher ratings due to the usage of cosine similarity. Due to the scope of this work, we have to leave a further investigation into the implications of these trends for future work.

Altogether, we answer **RQ3** strongly in the affirmative: *LBD provides a significantly and considerably stronger correlation between the confidence and accuracy of predictions than CMF or OrdRec.* Consequently, LBD is substantially better at separating accurate predictions from those with moderate or high prediction errors.

6.4 High-precision targeted recommendation

Finally, in order to investigate whether the better confidence modeling of LBD could translate into meaningful improvements for downstream tasks, we turn to **RQ4** and ask *whether LBD can reach higher performance in a high-precision recommendation task.*

For this question, we simulate a notification task where each user can only be recommended a single item, and due to the mild intrusiveness of a notification, only N total users will receive a recommendation. Therefore, this task could be divided into two stages: first to recognize what is the best item to recommend for each user, and second, to determine which N recommendations of the first stage are most likely to result in a positive reaction. We

expect that an accurate confidence model is better suited for both stages of this task.

We divide our simulation in these stages: first, for each user we select the test set item that has the highest predicted probability of a 4.5+ star rating (i.e., for models not relying on binning: the probability of being at least 4.25). For MF, this is also equivalent to simply using its predicted rating value [33]. We then use the above estimates to order the resulting user-item combinations. Then, for a given N, we calculate what percentage of the top-N user-item combinations truly did receive a 4.5+ star rating in the test set (Precision@1), i.e., the number of users who are happy with their notification. Note that N does not exceed the true number of users who have a 4.5+ star test set rating.

Figure 4 shows the results for this experiment in terms of Precision@1 for various values of N as well as models' relative improvement in the metric over MF. From the figure we see that all methods can identify a highly relevant item for most users, especially when N is small, with the task becoming progressively more difficult as N increases. Surprisingly, our results only show minor differences between CMF and MF, with the relative improvement of CMF limited to below 2% for most values of N. In contrast, OrdRec-UI, LBD-S and LBD-A start off with a substantially higher precision@1 for $N = 100$ and consistently outperform CMF even as N increases and users with lower confidence are included. Model performance for a selection of user numbers is also shown in Table 4. We can see that the performance of LBD-S is similar to that of OrdRec-UI, with frequently overlapping means and no significant improvement of one model over other. It appears that OrdRec-UI's weaker confidence modeling may still be quite efficient at identifying low-error high-rating items. LBD-A, on the other hand, consistently achieves the highest performance, with particularly noticeable improvement over OrdRec-UI of approximately 1.5% (absolute differences) for $1000 \leq N \leq 10{,}000$. As the improvements of LBD-A are also significant for all listed values $N > 100$, we conclude that LBD-A is meaningfully better at this high-precision targeted recommendation task, likely due to its better confidence modeling capabilities.

As such, we also answer **RQ4** in the affirmative: *LBD, and particularly LBD-A, brings a meaningful performance improvement for the high-precision targeted recommendation task over MF, CMF and OrdRec.* Combined with the conclusion from **RQ3**, our results strongly indicate that the confidence model of LBD is both a better indication of its prediction accuracy and that this higher accuracy can be translated to meaningful benefits for downstream tasks.

7 CONCLUSION AND FUTURE WORK

In this work, we investigated whether confidence modeling for RecSys is possible while maintaining high predictive accuracy and without introducing high model-complexity or heavy computational costs. To this end, we proposed learned beta distributions (LBD) as a simple RecSys with an explicit model of confidence while having the same model-complexity as MF.

Our experimental results show that LBD has competitive predictive performance with MF, CMF and OrdRec, where no decreases were observed for any metric. Furthermore, unlike CMF and OrdRec, we found that the confidence of LBD correlates strongly with its accuracy, allowing it to better separate predictions on their accuracy. Finally, we found that LBD's superior confidence modeling translates to a meaningfully higher precision in a high-precision targeted recommendation task, compared to the other methods.

We conclude that with the introduction of LBD, we have shown that accurate recommendation can be combined with accurate confidence modeling and low model-complexity. Furthermore, our findings indicate that such confidence models can lead to meaningful improvements in downstream tasks: LBD thereby provides a very promising potential for novel future applications.

Future work could investigate whether the confidence model of LBD can be used to better understand user preferences, or the uncertainty contained in datasets. While this work focused on matching the model-complexity of MF, the concept of LBD is easily applicable or extendable to more complex RecSys models. Similarly, while we tackled rating-prediction, LBD could be extended to confidence modeling for predictions of implicit feedback signals, e.g., clicks, purchases or watchtime. Additionally, it would be of interest to better understand how confidence modeling of LBD interacts with various cognitive and statistical biases, and the approaches used to correct them [13, 23, 43, 47]. Finally, LBD is designed to be practical for real-world settings; therefore, we hope practitioners will consider LBD and its many potential applications.

ACKNOWLEDGMENTS

This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. EINF-4538.

Code and data

To facilitate the reproducibility of the reported results, this work only made use of publicly available data and our experimental implementation is publicly available at https://github.com/NKNY/confidencerecsys2023.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.

[2] Gediminas Adomavicius, Sreeharsha Kamireddy, and YoungOk Kwon. 2007. Towards More Confident Recommendations: Improving Recommender Systems Using Filtering Approach Based on Rating Variance. (2007), 6.

[3] Vito Walter Anelli, Amra Delić, Gabriele Sottocornola, Jessie Smith, Nazareno Andrade, Luca Belli, Michael Bronstein, Akshay Gupta, Sofia Ira Ktena, Alexandre Lung-Yut-Fong, et al. 2020. RecSys 2020 challenge workshop: engagement prediction on Twitter's home timeline. In *Fourteenth ACM Conference on Recommender Systems*. 623–627.

[4] Cesare Bernardis, Maurizio Ferrari Dacrema, and Paolo Cremonesi. 2019. Estimating Confidence of Individual User Predictions in Item-based Recommender Systems. In *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization*. Association for Computing Machinery, 149–156.

[5] Djallel Bouneffouf, Amel Bouzeghoub, and Alda Lopes Ganarski. 2013. Risk-Aware Recommender Systems. In *Neural Information Processing*, Minho Lee, Akira Hirose, Zeng-Guang Hou, and Rhee Man Kil (Eds.). Springer, 57–65.

[6] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs.*

[7] George Casella and Edward I. George. 1992. Explaining the Gibbs Sampler. *The American Statistician* 46, 3 (1992), 167–174.

[8] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. 2011. Contextual Bandits with Linear Payoff Functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 208–214.

[9] Annie Cuyt, Vigdis Brevik Petersen, Brigitte Verdonk, Haakon Waadeland, and William B. Jones (Eds.). 2008. *Handbook of Continued Fractions for Special Functions*. Springer.

[10] Arjun K Gupta and Saralees Nadarajah. 2004. *Handbook of beta distribution and its applications*. CRC press.

[11] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[12] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining Collaborative Filtering Recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*. ACM, 241–250.

[13] Jin Huang, Harrie Oosterhuis, and Maarten De Rijke. 2022. It Is Different When Items Are Older: Debiasing Recommendations When Selection Bias and User Preferences Are Dynamic. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. ACM, 381–389.

[14] 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.

[15] Olivier Jeunen and Bart Goethals. 2021. Pessimistic Reward Models for Off-Policy Learning in Recommendation. In *Fifteenth ACM Conference on Recommender Systems*. 63–74.

[16] Rong Jin, Luo Si, ChengXiang Zhai, and Jamie Callan. 2003. Collaborative Filtering with Decoupled Models for Preferences and Ratings. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*. ACM, 309–316.

[17] Christopher C Johnson. 2014. Logistic matrix factorization for implicit feedback data. *Advances in Neural Information Processing Systems* 27, 78 (2014), 1–9.

[18] Norman L Johnson, Samuel Kotz, and N Balakrishnan. 1994. Beta distributions. *Continuous univariate distributions. 2nd ed. New York, NY: John Wiley and Sons* (1994), 221–235.

[19] Zahid Younas Khan, Zhendong Niu, Sulis Sandiwarno, and Rukundo Prince. 2021. Deep Learning Techniques for Rating Prediction: A Survey of the State-of-the-Art. *Artificial Intelligence Review* 54, 1 (2021), 95–135.

[20] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs]

[21] Michael Kläs and Anna Maria Vollmer. 2018. Uncertainty in machine learning applications: A practice-driven classification of uncertainty. In *International conference on computer safety, reliability, and security*. Springer, 431–438.

[22] Bart P Knijnenburg, Martijn C Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the user experience of recommender systems. *User modeling and user-adapted interaction* 22, 4 (2012), 441–504.

[23] Norman Knyazev and Harrie Oosterhuis. 2022. The Bandwagon Effect: Not Just Another Bias. In *Proceedings of the 2022 ACM SIGIR International Conference on the Theory of Information Retrieval*.

[24] Yehuda Koren. 2009. Collaborative Filtering with Temporal Dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 447–456.

[25] Yehuda Koren and Robert Bell. 2011. Advances in Collaborative Filtering. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, 145–186.

[26] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[27] Yehuda Koren and Joe Sill. 2011. OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions. In *Proceedings of the Fifth ACM Conference on Recommender Systems*. ACM, 117–124.

[28] Yew Jin Lim and Yee Whye Teh. 2007. Variational Bayesian Approach to Movie Rating Prediction. (2007).

[29] Benjamin Marlin. 2004. *Collaborative filtering: A machine learning perspective*. University of Toronto Toronto.

[30] Maciej A. Mazurowski. 2013. Estimating Confidence of Individual Rating Predictions in Collaborative Filtering Recommender Systems. *Expert Systems with Applications* 40, 10 (2013), 3847–3857.

[31] Sean McNee, Shyong Lam, Catherine Guetzlaff, Joseph Konstan, and John Riedl. 2003. Confidence Displays and Training in Recommender Systems.

[32] Rus M. Mesas and Alejandro Bellogín. 2020. Exploiting Recommendation Confidence in Decision-Aware Recommender Systems. *Journal of Intelligent Information Systems* 54, 1 (2020), 45–78.

[33] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems*, Vol. 20. Curran Associates, Inc.

[34] Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. 2020. Confidence-aware learning for deep neural networks. In *international conference on machine learning*. PMLR, 7034–7044.

[35] Ladislav Peska and Stepan Balcar. 2022. The Effect of Feedback Granularity on Recommender Systems Performance. In *Sixteenth ACM Conference on Recommender Systems*. ACM, 586–591.

[36] Amy Rechkemmer and Ming Yin. 2022. When Confidence Meets Accuracy: Exploring the Effects of Multiple Performance Indicators on Trust in Machine Learning Models. In *CHI Conference on Human Factors in Computing Systems*. 1–14.

[37] James Reilly, Barry Smyth, Lorraine McGinty, and Kevin McCarthy. 2005. Critiquing with Confidence. In *Case-Based Reasoning Research and Development*, Héctor Muñoz-Ávila and Francesco Ricci (Eds.). Springer, 436–450.

[38] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *Fourteenth ACM Conference on Recommender Systems*. ACM, 240–248.

[39] Steffen Rendle, Li Zhang, and Yehuda Koren. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395* (2019).

[40] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender Systems: Introduction and Challenges. In *Recommender Systems Handbook*. Springer, 1–34.

[41] James Robinson-Cox and Robert Boik. 1998. Derivatives of the Incomplete Beta Function. *Journal of Statistical Software* 03 (1998).

[42] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*. Association for Computing Machinery, 880–887.

[43] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations As Treatments: Debiasing Learning and Evaluation. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. 1670–1679.

[44] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[45] Upendra Shardanand and Pattie Maes. 1995. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '95*. ACM Press, 210–217.

[46] Harald Steck. 2013. Evaluation of Recommendations: Rating-Prediction and Ranking. In *Proceedings of the Seventh ACM Conference on Recommender Systems*. 213–220.

[47] Adith Swaminathan and Thorsten Joachims. 2015. The Self-Normalized Estimator for Counterfactual Learning. In *Advances in Neural Information Processing Systems*, Vol. 28. 3231–3239.

[48] I.J. Thompson and A.R. Barnett. 1986. Coulomb and Bessel Functions of Complex Arguments and Order. *J. Comput. Phys.* 64, 2 (1986), 490–509.

[49] Chao Wang, Qi Liu, Runze Wu, Enhong Chen, Chuanren Liu, Xunpeng Huang, and Zhenya Huang. 2018. Confidence-Aware Matrix Factorization for Recommender Systems. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (2018).

[50] Mingyue Zhang, Xunhua Guo, and Guoqing Chen. 2016. Prediction Uncertainty in Collaborative Filtering: Enhancing Personalized Online Product Ranking. *Decision Support Systems* 83 (2016), 10–21.

[51] Yongfeng Zhang, Xu Chen, et al. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.

[52] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. 2015. Bayesian CP Factorization of Incomplete Tensors with Automatic Rank Determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 9 (2015), 1751–1763.