Gaussian Embeddings for Collaborative Filtering

Ludovic Dos Santos
Sorbonne Universities, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606
ludovic.dossantos@lip6.fr

Benjamin Piwowarski
Sorbonne Universities, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606
benjamin.piwowarski@lip6.fr

Patrick Gallinari
Sorbonne Universities, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606
patrick.gallinari@lip6.fr

ABSTRACT

Most collaborative filtering systems, such as matrix factorization, use vector representations for items and users. Those representations are deterministic, and do not allow modeling the uncertainty of the learned representation, which can be useful when a user has a small number of rated items (cold start), or when there is conflicting information about the behavior of a user or the ratings of an item. In this paper, we leverage recent works in learning Gaussian embeddings for the recommendation task. We show that this model performs well on three representative collections (Yahoo, Yelp and MovieLens) and analyze learned representations.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

recommender system, gaussian embeddings, probabilistic representation

ACM Reference format:
Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. 2017. Gaussian Embeddings for Collaborative Filtering. In *Proceedings of SIGIR'17, August 7–11, 2017, Shinjuku, Tokyo, Japan, ,* 4 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080722

1 INTRODUCTION

Recommender systems help users find items they might like in large repositories, thus alleviating the information overload problem. Methods for recommending text items can be broadly classified into collaborative filtering (CF), content-based, and hybrid methods [6]. Among recommender systems, collaborative filtering systems are the most successful and widely used because they leverage past ratings from users and items, while content-based approaches typically build users (or items) profiles from items (or users) predefined representations. Among collaborative recommender systems we are interested in those that represent users and items as vectors in a common latent recommendation space. Matrix factorization derivatives are a typical example of such models. The main assumption is that the inner product between a user and an item representation is correlated with the rating the user would give to the item. The main interest of these models is their potential for integrating different sources of information [12] as well as their good performance, both in efficiency and effectiveness [6]. The most successful approaches are based on learning-to-rank approaches such as [5, 10].

Despite their successes, one limitation of those models is their lack of handling of uncertainty. Even when they are based on a probabilistic model, e.g. [5, 7], they only suppose a Gaussian prior over the user and item embeddings but still learn a deterministic representation. The prior is thus only used as a regularization term on user and item representations.

Uncertain representation can have various causes, either related to the lack of information (new users or items, or users that did not rate any movie of a given genre), or to contradictions between user or item ratings. To illustrate the latter, let us take a user that likes only a part of Kung Fu movies, but with no clear pattern, i.e. no subgenre. With usual approaches, such a user will have a component set to zero for the direction of the space corresponding to Kung Fu. With our proposed approach, we would still have a zero mean, but with a high variance. Our hypothesis is that, because of these two factors, namely cold start and conflicting information, training will result in learned representations with different confidences, and that this uncertainty is important for recommending.

Moreover, using a density rather than a fixed point has an important potential for developing new approaches to recommendation. First, instead of computing a score for each item, the model can compute a probability distribution, as well as the covariance between two different item scores. This can be interesting when trying to diversify result lists, since this information can be leveraged e.g. by Portfolio approaches [9]: The model could propose diversified lists with different degrees of risk. Secondly, such models are interesting because they can serve as a basis for integrating different sources of external information, and thus serve as a better bridge between content-based approaches and collaborative filtering ones. Thirdly, time could be modeled by supposing that the variance of the representation increases with time if no new information (i.e. user ratings) are provided.

Our model use Gaussian embeddings which have been recently and successfully used for learning word [8], knowledge graph [4] or nodes in a graph [2] embeddings. More precisely, each user or item representation corresponds to a Gaussian distribution where the mean and the variance are learned. Said otherwise, instead of a fixed point in space, a density represents each item or user. The variance term is a measure of uncertainty associated to the user/item representation.

In the following, we first describe the model, and then show experimentally that it performs very well compared to state-of-the-art approaches on three different datasets. Our contributions are (1) the use of a new type of representations for items and users in collaborative filtering; (2) extensive experimental work that shows

SIGIR'17, August 7–11, 2017, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3080722

the good performance of the model; (3) qualitative analysis to show how the variance component is used in the model.

2 MODEL

Our model is learned through a pairwise learning-to-rank criteria that was first proposed by Rendle et al. [5] for collaborative filtering (Bayesian Personalized Ranking, BPR). Formally, they optimize the maximum a posteriori of the training dataset $\mathcal{D}$, i.e. $p(\Theta|\mathcal{D}) \propto p(\mathcal{D}|\Theta)\,p(\Theta)$ where $\mathcal{D}$ represents the set of all ordered triples (u,i,j) – the user $u \in \mathcal{U}$ prefers item $i \in \mathcal{I}$ to item $j \in \mathcal{I}$, and Θ, the model parameters. The factor $p(\Theta)$ corresponds to the prior on the item and user representations (a Gaussian prior in [5]). Then, using the standard hypothesis of the independence of samples given the model parameters, we have

$$p(\mathcal{D}|\Theta) = \prod_{(u,i,j)\in\mathcal{D}} p(i >_u j|\Theta) \tag{1}$$

In [5], the probability that user u prefers item i to item j, noted by $i >_u j$, is given by the sigmoid function of the difference of the inner products, that is:

$$p(i >_u j|\Theta) = \sigma\left(x_i \cdot x_u + b_i - x_j \cdot x_u - b_j\right) \tag{2}$$

where b_i (resp. b_j) is the bias for item i (resp. j).

The Gaussian Embeddings Ranking Model. In this work, while we start with Eq. (1), the different variables $x_\bullet$ ($\bullet$ is either a user u or an item j) are random variables, denoted $\mathbf{X}_\bullet$, and not elements of a vector space. We can hence estimate directly probability $p(i >_u j|\Theta)$ of the inner product $\mathbf{X}_u \cdot \mathbf{X}_i$ being higher than $\mathbf{X}_u \cdot \mathbf{X}_i$.

We start by supposing that user and item representations follow a normal distribution: for any user u and item i,

$$\mathbf{X}_i \sim \mathcal{N}(\mu_i, \Sigma_i) \text{ and } \mathbf{X}_u \sim \mathcal{N}(\mu_u, \Sigma_u)$$

We suppose that $\Sigma_\bullet = \mathrm{diag}(\sigma_{\bullet 1}, \ldots, \sigma_{\bullet N})$ is a diagonal covariance matrix to limit the complexity of the model (N is the dimension of the latent space): in practice, we have $2N$ parameters for each user, and $2N+1$ for each item (+1 because of the bias). The variance is associated with each element of the canonical basis of the vector space. In practice, we hypothesize that this helps the model to learn meaningful dimensions, since it forces the use of the canonical latent space basis.

Since $\mathbf{X}_i$ and $\mathbf{X}_u$ are random variables, their inner product is also a random variable, and we do not need to use the sigmoid function of Eq. (2) as for BPR to compute the probability that user u prefers i to j. We can instead directly use the random variables defined above, which leads to:

$$\begin{aligned} p(i >_u j|\Theta) &= p\left(\mathbf{X}_i \cdot \mathbf{X}_u + b_i > \mathbf{X}_j \cdot \mathbf{X}_u + b_j|\Theta\right) \\ &= p\Big(\underbrace{\mathbf{X}_u \cdot \left(\mathbf{X}_j - \mathbf{X}_i\right)}_{\mathbf{Z}_{uij}} < b_i - b_j\Big|\Theta\Big) \end{aligned}$$

where b_i and b_j are the item biases[1]. To compute the above equation, we use the following two simplifications:

(1) Item representations are independent given the model parameters. This implies that the difference $\mathbf{X}_j - \mathbf{X}_i$ follows a normal distribution $\mathcal{N}\left(\mu_j - \mu_i, \Sigma_i + \Sigma_j\right)$
(2) We approximate the inner product distribution by a Gaussian using moment matching (mean and variance), using the results from [1] who defined the moments of the inner product of two Gaussian random variables; with our notations, the two first moments of $\mathbf{Z}_{uij}$ are defined as:

$$\mathbb{E}\left[\mathbf{Z}_{uij}\right] = \mu_u^\top(\mu_j - \mu_i) \tag{3}$$

$$\begin{aligned} \mathrm{Var}\left[\mathbf{Z}_{uij}\right] &= 2\mu_u^\top\left(\Sigma_i + \Sigma_j\right)\mu_u + \left(\mu_j - \mu_i\right)^\top \Sigma_u(\mu_j - \mu_i) \\ &\quad + \mathrm{tr}\left(\Sigma_u\left(\Sigma_j + \Sigma_i\right)\right) \\ &= \sum_k \left(\sigma_{ik} + \sigma_{jk}\right)\left(2\mu_{uk}^2 + \sigma_{uk}\right) + \sigma_{uk}\left(\mu_{jk} - \mu_{ik}\right)^2 \end{aligned} \tag{4}$$

The above assumptions allow us to write:

$$p(i >_u j|\Theta) \approx \int_{-\infty}^{b_i - b_j} \mathcal{N}\left(x; \mathbb{E}\left[\mathbf{Z}_{uij}\right], \mathrm{Var}\left[\mathbf{Z}_{uij}\right]\right) \mathrm{d}x \tag{5}$$

which is the Normal cumulative distribution function. This function can be easily differentiated with respect to the $\mu_{\bullet k}$ and $\sigma_{\bullet k}$ ($\bullet$ is a user or an item) since it can be rewritten using the error function (ERF), which is trivially derivable.

From Equations (5) and (4) and ignoring the biases, we can see that the difference between the inner products $\mu_u \cdot \mu_i$ and $\mu_u \cdot \mu_j$ is controlled by the variance of the user and the item (especially for the components of the means that have a high magnitude) – the larger, the closer to 0.5 the probability. This is the main difference with other matrix factorization-based approaches.

Finally, we have to define the prior distribution over the parameters Θ, i.e. the means and variances. As we consider only a diagonal covariance matrix, we can consider an independent prior for each mean and variance, i.e. for any $\mu_{\bullet k}$ and $\Sigma_{\bullet kk}$. In this case, using a normal-gamma prior is a natural choice since it is the conjugate distribution of a normal. More specifically, we suppose that

$$\left(\mu_{\bullet k}, \Sigma_{\bullet kk}^{-1}\right) \sim \mathrm{NormalGamma}(\nu, \lambda, \alpha, \beta) \tag{6}$$

with $\nu = 0$, $\lambda = 1$, $\alpha = 2$ and $\beta = 2$ – these parameters do not change much the solution, and were chosen so that user and item representations are not too much constrained. In the absence of data, this would set, for each component, the mean to 0 and the variance to 1, which corresponds to the mode of the normal-gamma distribution.

Loss function. Plugging in Eq. (6) and (1) gives

$$\begin{aligned} \mathcal{L} &= \log p(\Theta) + \sum_{(u,i,j)\in D_S} \log p(i >_u j|\Theta) \\ &\overset{+}{=} \sum_{\bullet,i}\left[\left(\frac{1}{2} - \alpha\right)\log\left(\Sigma_{\bullet i}\right) - \frac{2\beta + \lambda\mu_{\bullet i}^2}{2\Sigma_{\bullet i}}\right] - c\sum_i b_i^2 \\ &\quad + \sum_{(u,i,j)\in\mathcal{D}} \log p(i >_u j|\Theta) \end{aligned}$$

where $\bullet$ is a user or an item, $\overset{+}{=}$ means equal up to a constant, and $p(i >_u j|\Theta)$ is given by Eqs. (5), (3) and (4), and where we suppose a normal prior for the biases (to avoid overfitting). The parameters

[1] In this work, we did not consider them to be random variables, but they could be

Dataset	Users			Items			Ratings		
	10	20	50	10	20	50	10	20	50
Yahoo!	3,386	1,645	286	1,000	1,000	995	158,868	99,915	32,759
MovieLens	743	643	448	1,336	1,330	1,307	94,491	91,053	80,643
Yelp	13,775	8,828	3,388	44,877	39,355	27,878	980,528	790,859	466,522

Table 1: Datasets statistics by number of training examples (10, 20 or 50 ratings by user).

$\Theta = \{\Sigma_\bullet, \mu_\bullet\}_{\bullet \in \mathcal{U} \cup \mathcal{I}}$ are optimized with stochastic gradient update, picking a random triple at each step, and updating the correspond means and variances (for user u, and items i and j).

Ordering items. We could have used pairwise comparison from our model since the relation $i >_u j \iff p(i >_u j|\Theta) > 0.5$ defines a total order over items, but it does not make any difference in the ranking if $p(i >_u j|\Theta)$ equals 0.51 or 0.99, and this difference could be due to the sole difference between variances. To avoid this problem, we order items by their probability of having a positive score, i.e. $s_{ui} = p(\mathbf{X}_u \cdot \mathbf{X}_i + b_i > 0)$[2]. This ordering preserves the variance information in contrast to the pairwise comparison – a score with a high variance will be associated with a score s_{ui} close to 0.5.

3 EXPERIMENTS

We experimented with three different datasets extracted respectively from the *Yahoo! Music ratings for User Selected and Randomly Selected songs, version 1.0* [3], the MovieLens 100k dataset [4] and the Yelp Dataset Challenge [5]. The **Yahoo!** dataset contains ratings for songs. The original rating data includes 15,400 users, and 1,000 songs for approximately 350,000 ratings. The **MovieLens** dataset contains ratings for movies. The original rating data includes 943 users, and 1,682 movies for approximately 100,000 ratings. The **Yelp** dataset contains ratings for businesses. The original rating data includes 1,029,432 users, and 144,072 businesses for approximately 4.1 million ratings.

We experimented with the following models, which are all based on a learning-to-rank criterium (except for MP): **(MP)** Most popular ranking where items are ranked based on how often they have been seen in the past – while simple, this baseline is always competitive. There is no hyper-parameter to optimize for this procedure; **(SM)** Soft margin (hinge) ranking loss, using stochastic gradient descent [11] and the implementation from [3] optimizing all possible hyper-parameters; **(BPRMF)** Bayesian Personalized Ranking [5] on which our model cost function is based, using the implementation from [3] and optimizing all possible hyper-parameters; **(CofiRank)** A state-of-the art [10] ranking algorithm for recommender systems that optimizes an upper bound of nDCG[6]. We chose the hyper-parameters based on [10]; **GER** Our model named Gaussian Embeddings Ranking. The number of dimensions was set among 5, 10, 20, 50, 100, 200, 500 and 1000.

[2]We use the same moment matching approximation to compute the inner product distribution.

[3]This dataset is available through the Yahoo! Webscope data sharing program http://webscope.sandbox.yahoo.com/.

[4]Available at http://grouplens.org/datasets/movielens/100k/.

[5]See https://www.yelp.com/dataset_challenge for more information.

[6]https://github.com/markusweimer/cofirank

Train Size →	10			20			50		
Model ↓	N@1	N@5	N@10	N@1	N@5	N@10	N@1	N@5	N@10
Yahoo!									
MP	53.0	59.1	67.3	52.5	58.3	66.4	53.6	57.8	64.0
BPRMF	52.8	59.0	67.2	52.2	58.3	66.4	52.2	57.7	63.5
SM	50.9	56.7	65.4	49.7	55.6	64.2	49.9	54.1	60.3
CR	**53.5**	**60.3**	68.2	**57.8**	**61.7**	**68.9**	**56.0**	**60.0**	**65.6**
GER	**53.5**	**60.3**	**68.3**	53.8	60.7	68.2	54.3	59.6	65.3
MovieLens									
MP	66.0	64.7	65.8	68.4	65.3	66.3	69.1	67.4	67.5
BPRMF	66.1	64.6	65.7	66.3	64.3	65.8	66.9	65.0	66.2
SM	55.9	57.5	60.3	58.3	59.6	61.6	58.6	60.4	62.5
CR	69.0	67.3	68.6	69.7	68.5	69.5	71.4	69.4	70.6
GER	**70.3**	**67.7**	**70.0**	**72.0**	**69.5**	**71.1**	**72.5**	**71.3**	**71.5**
Yelp									
MP	40.7	41.5	46.9	39.5	39.9	44.7	37.4	37.6	41.4
BPRMF	40.8	41.3	46.8	39.6	39.8	44.6	37.3	37.2	40.9
SM	37.3	38.3	44.4	35.8	36.9	41.9	33.4	34.1	38.0
CR	47.1	46.9	51.1	46.5	46.6	50.4	46.2	45.8	48.6
GER	**55.2**	**52.2**	**56.2**	**57.4**	**53.5**	**56.4**	**58.2**	**53.7**	**55.3**

Table 2: Collaborative Ranking results. nDCG values (×100) at different truncation levels are shown within the main columns, which are split based on the amount of training ratings.

We followed mostly the experimental procedure of [10]. More precisely, the dataset has been preprocessed as follows. For each dataset, and for each user, a fixed amount (10, 20, or 50) of items are kept to train the model, 10 for the validation set and the rest for testing. Users with fewer than 30, 40, 70 ratings were removed to ensure that we at least 10 ratings were available for testing. We also removed items that were not rated by at least 5 users. The statistics of the training datasets are shown in Table 1. The validation set is used to select the best hyper-parameters for GER[7], SM and BPRMF. Finally, for evaluation, we re-rank the judged items using the evaluated model, and use a metric for ranked lists of items, namely nDCG at ranks 1, 5 and 10. The reported results are the average of 5 different experiments, with 5 different train/validation/test splits.

Results are reported in Table 2. There are roughly three groups of models: (1) Soft Margin (SM) that performed the worst; (2) BPRMF and most popular (MP); (3) GER and CofiRank (CR). Most popular (MP) performs well because of the chosen experimental evaluation where only items *seen* by the user are evaluated and ranked.

Our model (GER) outperforms the others on the MovieLens and Yelp dataset. It is usually above CofiRank (nDCG difference between 0.01 and 0.08), with the exception of the Yahoo dataset for train sizes 20 and 50 (nDCG difference between 0.001 and 0.02). Overall, GER is performing very well on three datasets with different characteristics in terms of rating behavior and number of ratings.

[7]We observed that a dimension of 50 generally gives good results independently of the dataset, while c (bias regularization) $\approx 10^{-5}$ give good results on MovieLens and Yelp, but needs to be higher ($\approx 10^{-3}$) on Yahoo to achieve good results.

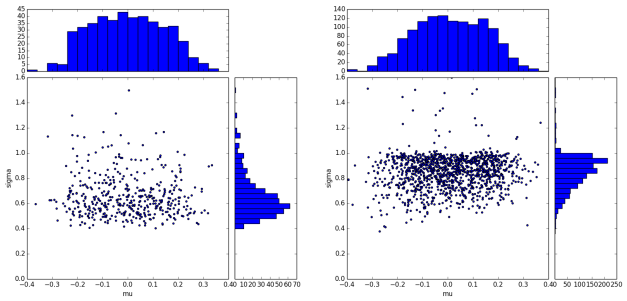

Figure 1: User (left) and item (right) learned first component representation plots for T50 and a representation dimension of 50.

4 ANALYSIS

We now turn to the analysis of results, based on the study of the learned representations for the Yahoo dataset. We used the hyper-parameters that were selected on the validation set, and looked at the set of mean-variance couples, i.e. the $(\mu_{\bullet k}, \Sigma_{\bullet k})$ for $k \in \{1, \ldots, N\}$ and $\bullet \in \mathcal{U} \cup \mathcal{I}$.

In Figure 1, we plotted the different couples $(\mu_{\bullet k}, \Sigma_{\bullet k})$, as well as the histogram of mean and variance. We can see that the model exploits the different ranges of mean (-0.3 to 0.3) and, more importantly, variance (0.4 to 1.2) around the priors. This was satisfying since it was not obvious that the variance component would be used by the model, i.e. that it would deviate from its prior.

To have a deeper insight on the use of the variance, in Figure 2, we plotted violin (density) plots of the variance of users and items, depending on the latent space dimension (5 to 1000), and on the training size (10 to 50). We can observe that when the amount of training data increases, the median of the $\sigma_{\bullet k}$ decreases while the variance of the $\sigma_{\bullet k}$ increases. This corresponds to our hypotheses: the former shows that more evidence reduces the uncertainty on the representation, while the latter shows that more training data induces more conflicting ratings and hence an increase in some of the variances – this is confirmed by the case of item representations that receive more ratings. Finally, we can observe that one the dimension increases too much, most of the density of the $\sigma_{\bullet k}$ is centered around the prior – which could imply that most of the dimensions were simply not used.

5 CONCLUSION

In this paper, we proposed a collaborative recommender system where user and items correspond to distributions in a latent space, rather than to a deterministic point for other representation-based models, allowing to cope with the inherent uncertainty due to lack of information, or to conflicting information. We show on three datasets that our model outperforms state-of-the-art works. We analyzed qualitatively the results showing that the variance distribution was correlated with space dimension and train size.

Moreover, this type of model has various advantages that we have not exploited yet: (1) the integration of different sources of information, exploiting the uncertainty of each source; (2) the use of the variance for handling time (the representation becomes more and more uncertain if no new information is provided); and (3), the diversification of proposed rankings (e.g. Portfolio theory can be used to produce rankings with various risks).

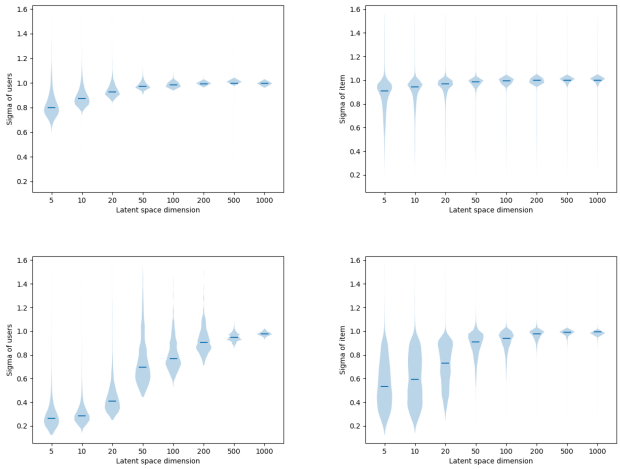

Figure 2: Violin (density) plots for the variance of users (left) and items (right) for latent space dimensions T10 (top) and T50 (bottom).

Acknowledgments. This work has been partially supported by: FUI PULSAR (BPI France, Région Ile de France) and LOCUST (ANR-15-CE23-0027-01).

REFERENCES

[1] Gerald G Brown and Herbert C Rutemiller. 1977. Means and Variances of Stochastic Vector Products with Applications to Random Linear Models. *Management Science* 24, 2 (October 1977).
[2] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. 2016. Multilabel Classification on Heterogeneous Graphs with Gaussian Embeddings. In *ECML*. DOI: http://dx.doi.org/10.1007/978-3-319-46227-1_38
[3] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A Free Recommender System Library. In *RecSys*.
[4] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. *Learning to Represent Knowledge Graphs with Gaussian Embedding*. DOI: http://dx.doi.org/10.1145/2806416.2806502
[5] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.
[6] Francesco Ricci, Lior Rokach, Bracha Shapira, and Kantor Paul B. 2011. *Recommender Systems Handbook*. Vol. 532. DOI: http://dx.doi.org/10.1007/978-0-387-85820-3
[7] R Salakhutdinov and A Mnih. 2007. Probabilistic Matrix Factorization. In *Proceedings of Advances in Neural Information Processing Systems*, Vol. 20. 1257–1264.
[8] Luke Vilnis and Andrew McCallum. 2014. Word Representations via Gaussian Embedding. In *ICLR*.
[9] Jun Wang and Jianhan Zhu. 2009. Portfolio theory of information retrieval. In *SIGIR*. ACM Press. DOI: http://dx.doi.org/10.1145/1571941.1571963
[10] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. 2007. CofiRank Maximum Margin Matrix Factorization for Collaborative Ranking. In *Advances in Neural Information Processing Systems*. 1–3.
[11] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. 2008. Improving maximum margin matrix factorization. *Machine Learning* 72, 3 (2008), 263–276.
[12] Fuzheng Zhang, Nicholas Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. DOI: http://dx.doi.org/10.1145/2939672.2939673