Improving Pairwise Learning for Item Recommendation from Implicit Feedback

Steffen Rendle*
University of Konstanz
78457 Konstanz, Germany
steffen.rendle@uni-konstanz.de

Christoph Freudenthaler
University of Konstanz
78457 Konstanz, Germany
chr.freudenthaler@gmail.com

ABSTRACT

Pairwise algorithms are popular for learning recommender systems from implicit feedback. For each user, or more generally context, they try to discriminate between a small set of selected items and the large set of remaining (irrelevant) items. Learning is typically based on stochastic gradient descent (SGD) with uniformly drawn pairs. In this work, we show that convergence of such SGD learning algorithms slows down considerably if the item popularity has a tailed distribution. We propose a non-uniform item sampler to overcome this problem. The proposed sampler is context-dependent and oversamples informative pairs to speed up convergence. An efficient implementation with constant amortized runtime costs is developed. Furthermore, it is shown how the proposed learning algorithm can be applied to a large class of recommender models. The properties of the new learning algorithm are studied empirically on two real-world recommender system problems. The experiments indicate that the proposed adaptive sampler improves the state-of-the art learning algorithm largely in convergence without negative effects on prediction quality or iteration runtime.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Item Recommendation; Recommender Systems; Matrix Factorization; Factorization Model

*Current affiliation: Google Inc.

WSDM'14, February 24–28, 2014, New York, New York, USA.

http://dx.doi.org/10.1145/2556195.2556248.

1. INTRODUCTION

Recommender systems have become an important feature in modern websites, e.g. in Amazon, Netflix, or Flickr. Click rates, revenues and other measures of success may be increased by the application of effective recommender systems. The difficult task is to identify relevant items even if they are generally unpopular. Recommender systems leverage available context such as user information, time, location, etc. to filter relevant items. Thereby, also items from the tails of the popularity distribution are successfully recommended.

In practice, often only implicit feedback is available to learn a recommender system. Examples for implicit feedback are clicks, watched movies, played songs, purchases or assigned tags. A characteristic of implicit feedback is that it is one-class, i.e. only positive observations are available. For example, a website logs that a user has bought an item on a specific day or chosen a tag for a specific resource. A common approach in recommender systems to deal with this data is to assume that everything that has not been selected is of less interest for the user in this situation. This idea results in a pairwise ranking loss that tries to discriminate between a small set of selected items and a very large set of all remaining items. Due to the very large number of pairs, learning algorithms are usually based on sampling pairs (uniformly) and applying stochastic gradient descent (SGD). This optimization framework is also known as Bayesian Personalized Ranking (BPR) [14]. Many recent published recommender systems use BPR for learning, including tensor factorization models for tag recommendation [15], relation extraction [17], sequential shopping recommender with taxonomies [8], focused matrix factorization for advertisement [7], hierarchical latent factor models [1] or co-factorization machines [5].

In this paper, it is shown that uniform sampling pairs results in slow convergence, especially if the pool of items is large and the overall item-popularity is tailed. Both properties are common to most real-world data sets. We argue that most SGD updates have no effect because the gradient vanishes. The reason is that a uniformly sampled negative item is very likely to be ranked correctly below a (random) observed item and thus the gradient of the pair is near 0. Empirically, we show that simple oversampling by global popularity is not sufficient to solve this problem. We propose a non-uniform sampling distribution that adapts both to the context and the current state of learning. An efficient sampling algorithm with constant amortized runtime complexity is developed to sample from the proposed distribution. Experiments on two real-world recommender applica-

tions indicate that the proposed adaptive sampler improves the state-of-the art learning algorithm largely in convergence without negative effects on prediction quality or iteration runtime. Finally, we show how to generalize the algorithm to a large class of factorization models.

2. PROBLEM STATEMENT

First the problem of recommending items from implicit feedback data is described. Then pairwise learning with BPR [14] is shortly recapitulated. The novel contribution of this section is to show that convergence of BPR algorithms slows down due to uniform sampling of negative items.

2.1 Ranking from Implicit Feedback

Let $S \subset C \times I$ be a set of observed actions, where C is a set of context and I a set of items. For instance, C could be a set of users, I a set of movies and S states which movies a user has watched, i.e. user-movie pairs. More complex examples can also be handled, e.g. C might hold additional variables like location, mood, time, sequences or attributes; also I might be described by additional variables, e.g. attributes or taxonomies (see section 4.2).

The task of item recommendation is to find a ranking $\hat{r}$ of items for each context. We formulate this by a bijective[1] function $\hat{r} : I \times C \to \{1, \dots, |I|\}$, where $\hat{r}(i|c)$ is the rank of item i for a given context c. The ranking function is usually modeled by a scoring function $\hat{y}(i|c)$ which is itself parametrized by a set of model parameters Θ. E.g. matrix factorization (MF) is a common choice for the scoring model $\hat{y}$ if i and c are categorical variables[2]. Ranking with any scoring model can be done by computing scores for all items (given a context c) and sorting the items by scores (for each context). The formal link between the rank $\hat{r}$ and the scoring function $\hat{y}$ can be defined as

$$\hat{r}(i|c) := |\{j : \hat{y}(j|c) \geq \hat{y}(i|c)\}|. \tag{1}$$

E.g. for the top item (i.e. rank 1) there is only one item (the item itself) that is as large or larger. For the item at rank 2 there are only 2 items where the score is as large or larger, etc.

The ranking itself is uniquely (up to ties) specified by the values of the model parameters Θ – through the scoring function $\hat{y}$.

2.2 Pairwise Learning from Implicit Feedback

The values of the model parameters Θ are learned from the implicit feedback data S. A popular approach to learn the model parameters Θ is based on pairwise learning. The idea is to discriminate for each context $c \in C$ between the selected items $I^+(c) := \{i : (i, c) \in S\}$ and the remaining items $I \setminus I^+(c)$. An item i is preferred over an item j under a context c (notation $i \succ_c j$), if and only if i but not j was selected under context c:

$$i \succ_c j \Leftrightarrow i \in I^+(c) \wedge j \in I \setminus I^+(c). \tag{2}$$

The set of all pairwise preferences $D_S \subseteq C \times I \times I$ can be defined as

$$(c, i, j) \in D_S :\Leftrightarrow i \in I^+(c) \wedge j \in I \setminus I^+(c). \tag{3}$$

[1]Bijective in I.

[2]Note that MF is just an example and the following presentation is not restricted to MF.

The link from pairwise preference to the model/ scoring function $\hat{y}$ is established by

$$p(i \succ_c j) := \sigma(\hat{y}(i|c) - \hat{y}(j|c)) \tag{4}$$

where $\sigma(x) = 1/(1 + \exp(-x))$. The goal is to maximize the likelihood of correctly ordering the preferences

$$\underset{\Theta}{\operatorname{argmax}} \prod_{(c,i,j) \in D_S} p(i \succ_c j), \tag{5}$$

which is equivalent to minimizing the negative log likelihood (NLL)

$$\mathrm{NLL} := - \sum_{(c,i,j) \in D_S} \ln \sigma(\hat{y}(c, i) - \hat{y}(c, j)). \tag{6}$$

SGD Learning.

The gradient of an arbitrary model parameter $\theta \in \Theta$ is

$$\frac{\partial \mathrm{NLL}}{\partial \theta} = \sum_{(c,i,j) \in D_S} (1 - \sigma(\hat{y}(c, i) - \hat{y}(c, j))) \frac{\partial (\hat{y}(c, i) - \hat{y}(c, j))}{\partial \theta}. \tag{7}$$

As the number of pairs $|D_S|$ is very large[3], learning algorithms typically are based on stochastic gradient descent (SGD). A pair $(c, i, j) \in D_S$ is sampled uniformly and a stochastic gradient descent step is performed:

$$\theta \leftarrow \theta - \eta \underbrace{(1 - \sigma(\hat{y}(c, i) - \hat{y}(c, j)))}_{=:\Delta_{c,i,j}} \frac{\partial}{\partial \theta} (\hat{y}(c, i) - \hat{y}(c, j)). \tag{8}$$

where η is the learning rate and has to be chosen small enough to ensure that the step is done in the right direction – i.e. the gradient is only (approximately) correct within a small region around θ. Note the difference between implicit feedback $S \subseteq C \times I$ and training pairs $D_S \subseteq C \times I \times I$.

Sampling a preference $(c, i, j) \in D_S$ uniformly can be done without explicitly storing D_S, by first sampling $(c, i) \in S$ and then sampling a negative item $j \in I \setminus I^+(c)$. See figure 2 for the full algorithm. The whole framework of a pairwise loss between selected and all remaining items, as well as the SGD algorithm using uniform sampling was proposed for item recommendation in [14] under the name *Bayesian Personalized Ranking* (BPR).

2.3 Issues in Tailed Item Distributions

Even though BPR has been successfully applied in numerous recommender applications and for a variety of models, in the following it is shown that convergence slows down considerably if the pool of items I is large and the item popularity is non-uniform distributed.

Gradient Magnitude.

Learning model parameters with BPR is done by looping over eq. (8). As it can be seen, each gradient step has a multiplicative scalar

$$\Delta_{c,i,j} := (1 - \sigma(\hat{y}(c, i) - \hat{y}(c, j))) = (1 - p(i \succ_c j)). \tag{9}$$

[3]Approximately $\mathcal{O}(|S|\,|I|)$ because the number of selected items $I^+(c)$ in a context c is much smaller than the number of all items I.

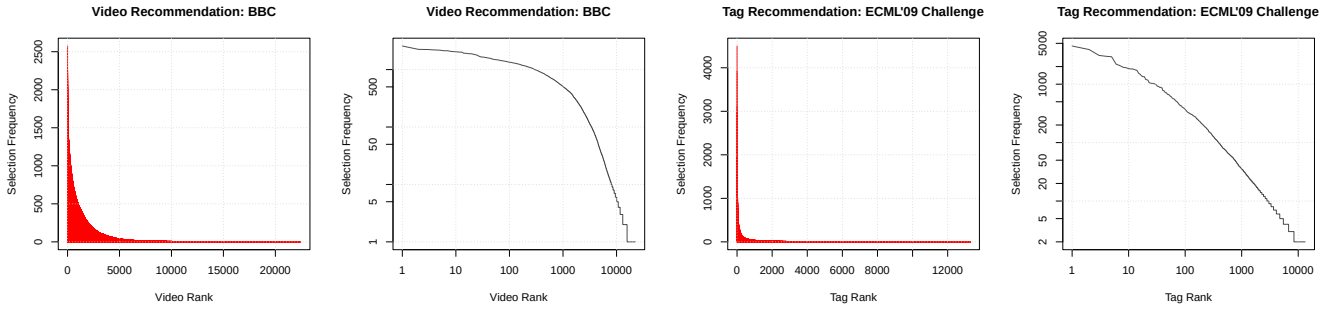

Figure 1: Item popularity in two example datasets (see section 5.1) plotted in raw and log-log scales. Popularity in both datasets is tailed.

1: **procedure** LEARNBPR(η, S)
2: Randomly initialize Θ
3: **repeat**
4: Draw $(c, i) \in S$ uniformly
5: Draw $j \in I \setminus I^+(c)$ uniformly
6: **for** $\theta \in \Theta$ **do**
7: Update θ with eq. (8).
8: **end for**
9: **until** convergence
10: **return** Θ
11: **end procedure**

Figure 2: BPR – a stochastic gradient descent algorithm that learns to discriminate observed selections $(c, i) \in S$ from all remaining items $j \in I \setminus I^+(c)$.

This quantity depends on how the scoring model (using the current model parameters Θ) would discriminate between the positive item i and the negative item j under context c. The quantity $\Delta_{c,i,j}$ is obviously a probability and is close to 0 if i is correctly assigned a larger score than j. It is close to 1 if j is falsely assigned a larger score than i. This means, $\Delta_{c,i,j}$ can be read as how much influence a pairwise preference (c, i, j) has for improving Θ. If $\Delta_{c,i,j}$ is close to 0, nothing can be learned from the case (c, i, j) because its gradient vanishes, i.e. θ is not changed by the update step (eq. 8). Thus, in the remainder $\Delta_{c,i,j}$ is called the *gradient magnitude* of a sampled case (c, i, j). Note that $\Delta_{c,i,j}$ depends on the model parameters Θ and thus $\Delta_{c,i,j}$ changes while learning.

Tailed Item Distributions.

In recommender systems, item popularity is typically non-uniform distributed and some items are in general more popular than others. Figure 1 shows two item popularity distributions for a movie and a social tagging dataset – see section 5.1 for details about the datasets. Both figures show that most items are rarely selected overall[4].

As discussed before, SGD algorithms cannot learn from samples (c, i, j) where the magnitude $\Delta_{c,i,j}$ is close to zero. $\Delta_{c,i,j}$ is supposed to be small when i is correctly ranked over j – i.e. $\Delta_{c,i,j}$ is smaller the larger the difference $\hat{y}(c, i) - \hat{y}(c, j)$ is. Because (c, i) is a positive observation (and overall positive observations are tailed distributed as in fig. 1) and j is uniformly drawn, it is very likely that the model score $\hat{y}(c, i)$ is also larger than $\hat{y}(c, j)$ and thus the gradient magnitude is small. Figure 3 shows the probability that a sample (c, i, j) has a gradient magnitude smaller than 0.01, 0.1 and 0.5 respectively. It can be seen that already after a few training epochs (one epoch includes $10 \cdot |S|$ many single BPR updates), almost all samples have very small magnitudes and therefore most of the samples are useless in the SGD algorithm – i.e. an update (eq. 8) does not change the value. It should be noted that this does not mean that the loss is inappropriate, but that the *good* samples have just not been seen by the algorithm yet. E.g. under a given context, a positive item might be estimated on rank 10 instead of 1, which means there are 9 very informative samples while the vast majority of the remaining |I| - 10 items are mainly non-informative. If $|I|$ is large, uniform sampling is very likely to need many iterations before finding some of these 9 samples and in the meantime the algorithm spends a lot of time applying updates on useless samples.

In the remainder of this work, non-uniform samplers are proposed that exchange the negative item sampler in the BPR algorithm (see fig. 2, line 5) to overcome this issue.

3. IMPROVED ITEM SAMPLING

In this section, non-uniform samplers for negative items are proposed to speed up convergence. The sampling distribution for negative items will be denoted as $p(j|c)$. First, sampling according to the global item popularity is presented as a baseline. Then an adaptive, context-aware sampling distribution that follows the belief of the ranking model $\hat{y}$ is proposed.

3.1 Static & Global Sampling

In section 2.3, the uniform sampling assumption ($p(j|c) \propto 1$) of pairwise learning was identified as the main reason of slow convergence of SGD learning. This resulted in mostly uninformative pairs which are trivial to rank in the right order.

A simple approach to increase the fraction of difficult pairs is to oversample popular items. The empirical popularity/ selection frequency (see fig. 1) can be used directly to define the sampling distribution

$$p(j|c) \propto |\{(c', j') \in S : j = j'\}|. \quad (10)$$

[4]However, keep in mind that recommender systems target at personalizing, i.e. overall unpopular items might be preferred by some users and should be ranked high.

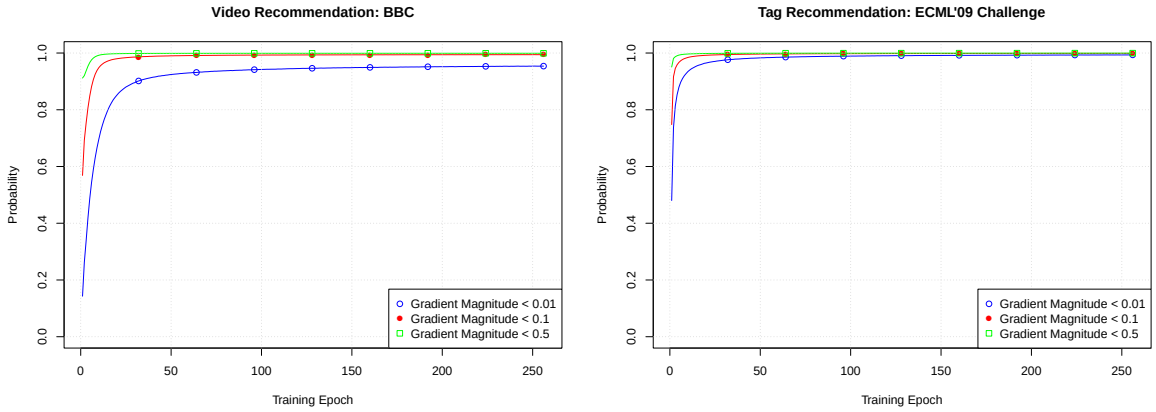

Figure 3: Gradient magnitude (eq. 9) measures how much influence ([0,1] interval) a training case has on the current learning process. The three curves depict how many magnitudes are smaller than 0.01, 0.1 and 0.5 when uniform sampling (BPR) is used. After only a few training epochs almost all magnitudes are smaller than 0.1 and most are smaller than 0.01. Training cases with near zero magnitudes slow down learning because the corresponding SGD update step won't change the model parameters but requires computational time. See fig. 6 for the development of the average gradient and section 5.1 for details on the experimental setup.

A sampler from this distribution is trivial to implement: an observation (c', j) is drawn uniformly from the observations S, then c' is discarded and j is used as the negative item.

Alternatively a parametric sampling can be used. Typically, the empirical distribution follows approximately an analytical law, e.g. a Geometric or Zipf distribution (see fig. 1). E.g. for the Geometric distribution[5]

$$p(j|c) = \gamma\,(1-\gamma)^{r(j)}, \quad \gamma \in (0,1) \tag{11}$$

or equivalently in another notation/ parametrization

$$p(j|c) \propto \exp(-r(j)/\lambda), \quad \lambda \in \mathbb{R}^+, \tag{12}$$

where $r(j)$ is the rank of item j according to global popularity ranking. The expected rank in the distribution of eq. (12) is identified by the parameter λ.

Whether to choose the empirical distribution (eq. 10) or its parametric counterpart results in approximately the same results. In practice, the empirical distribution might be easier to implement and we use it in the evaluation. However, the more complex models proposed later use a parametric approach and thus, it facilitates readability to discuss the parametric counterpart already here.

Properties.

The presented sampling distribution has two important properties:

1. **Global**: The item distribution is independent of the context, i.e. for different context, the distribution is the same.
2. **Static**: The distribution does not change while model parameters are learned.

[5]For readability, throughout the paper all ideas are illustrated with the Geometric distribution. It is straightforward to exchange the Geometric distribution (eqs. 12, 14, 21) by another analytic distribution.

Algorithm.

Because the distribution is global and static, sampling from the parametric item distribution (eq. 12) is simple: First, a rank r is sampled from the Geometric distribution[6] in $\mathcal{O}(1)$. Second, the item ranked on position r of the global popularity list[7] is returned in $\mathcal{O}(1)$.

In total, sampling from eq. (12) can be done without increasing the computational complexity of the original BPR algorithm.

Discussion.

The motivation for developing a sampler is to select for a given context c an item j such that the pair (i, j) is informative at the current state of learning (i.e. for the current values of Θ). The proposed popularity based sampler does not reflect this in two aspects: (1) It is static and thus does not take into account that the estimated rank $\hat{r}(j|c)$ of an item j changes during learning. E.g. an item might be ranked high in the beginning but after several steps of learning it is ranked low. (2) The sampler is global and does not reflect, that it depends on the context how informative an item is. E.g. one item might be interesting for one group of persons but not for another one. Both aspects can also be seen in the gradient magnitude $\Delta_{c,i,j}$, which depends on c and changes during learning.

3.2 Adaptive & Context-dependent Sampling

In the following a fine-grained sampler is developed that adapts both to the context and the current belief (i.e. to $\hat{y}$ through Θ) of the model. Analogously to the global item popularity (eq. 10) one could define a static context-specific popularity distribution

$$p(j|c) \propto |\{(c', j') \in S : c = c', j = j'\}| = \delta((c, j) \in S) \tag{13}$$

[6]With the parametrization of eq. (12), an exponential sampler can be used: $r \sim \lfloor \mathrm{Exp}(1/\lambda) \rfloor$.

[7]This (static) list can be precomputed once.

However, this distribution has two drawbacks: (1) it is defined on very few samples, i.e. for a given context c, there exists only a small subset $I^+(c)$ of selected items, where typically none is selected multiple times. This would make the item distribution a step function where the negative items are indistinguishable. (2) it does not take the current belief (i.e. Θ) into account.

Instead we propose to use the scoring function $\hat{y}$ to define the sampling distribution. Intuitively, when a negative item j for a given observation $(c,i) \in S$ should be sampled, the closer j is to the top (the smaller the rank $\hat{r}(j|c)$), the more informative is j. This can also be seen in the gradient magnitude $\Delta_{c,i,j}$: if (c,i) is given, we should choose j s.th. $\hat{y}(j|c)$ is *large* because it will increase $\Delta_{c,i,j}$. Instead of using the notion of a *large* score, it is better to formalize a small predicted rank $\hat{r}(j|c)$ because largeness of scores is relative w.r.t. other items but ranks are an absolute value. This allows to formulate an adaptive and context-aware sampling distribution

$$p(j|c) \propto \exp(-\hat{r}(j|c)/\lambda), \quad \lambda \in \mathbb{R}^+. \tag{14}$$

Properties.

The item distribution (eq. 14) depends on $\hat{r}(j|c)$. Remind that $\hat{r}(j|c)$ is the rank of item j among all items I using the score model $\hat{y}(j|c)$ for ordering items. Thus the proposed sampler is:

1. **Context-dependent**: The sampling probability depends on the context because $\hat{r}(j|c)$ is the estimated rank of item j for a given context c. The more the model $\hat{y}$ can distinguish different context, the more context-aware is the ranking and thus the sampler.

2. **Adaptive**: The sampler changes while model parameters Θ are learned because changes in Θ result consecutively in changes in the scoring model $\hat{y}$, the ranking $\hat{r}$ and sampler.

Trivial Algorithm.

The sampler for the negative item j given a positive observation $(c,i) \in S$ can be implemented by: First, sample a rank r from the Geometric distribution in $\mathcal{O}(1)$. Second, return the item j which is currently ranked on the r-th position, i.e. find j s.th. $\hat{r}(j|c) = r$ or $j = \hat{r}^{-1}(r|c)$. A trivial implementation of the second step would be to compute $\hat{y}(j|c)$ for all $j \in I$, then sort the items j by their score and finally return the item at place r. This algorithm has a complexity of $\mathcal{O}(|I| \cdot T_{\text{pred}} + |I| \log(|I|))$ where T_{pred} is the time for predicting a score. Note that this sampler should replace line 5 in algorithm 2 and would increase the runtime of BPR by a factor of $\mathcal{O}(|I| \cdot T_{\text{pred}} + |I| \log(|I|))$. This is clearly not feasible in practice.

4. EFFICIENT SAMPLING ALGORITHM

In the following it is shown how approximative sampling from eq. (14) can be implemented efficiently in amortized constant time for a broad class of factorization models. First the basic idea is shown for matrix factorization and then a generalization to factorization machines [13] is shown.

4.1 Matrix Factorization (MF)

Assume that the context C and items I are represented by categorical variables, i.e. $C = \{c_1, c_2, \ldots\}$ and $I = \{i_1, i_2, \ldots\}$ respectively. For example, each context c could correspond to a user in a personalization setting. Let the scoring model $\hat{y}$ be a matrix factorization (MF)

$$\hat{y}(l|c) := \sum_{f=1}^{k} v_{c,f}\, v_{l,f}, \quad V \in \mathbb{R}^{(C \cup I) \times k}. \tag{15}$$

where $k \in \mathbb{N}$ is the latent dimension and the factors V are the model parameters Θ. Scoring one item with MF (eq. 15) is clearly in $\mathcal{O}(k) =: T_{\text{pred}}$.

In the following, a fast adaptive and context-dependent sampling algorithm is derived which approximates the sampler from eq. (14) in amortized constant time. The idea is to formalize eq. (14) as a mixture of ranking distributions over normalized factors. The mixing probability is derived from a normalized version of the MF scoring function eq. (15). Note that the final sampler works with any MF model and no transformation has to be performed explicitly however the transformation is necessary to derive the algorithm.

Rank-Invariant Normalization.

First, a transformation $\hat{y}^*$ of $\hat{y}$ is defined

$$\hat{y}^*(l|c) := \sum_{f=1}^{k} p(f|c)\ \operatorname{sgn}(v_{c,f})\, v^*_{l,f} \tag{16}$$

where $p(f|c)$ is the probability function

$$p(f|c) :\propto |v_{c,f}|\, \sigma_f \tag{17}$$

and $v^*_{l,f}$ are standardized item factors

$$v^*_{l,f} = \frac{v_{l,f} - \mu_f}{\sigma_f} \tag{18}$$

with the empirical mean and variance over all item factors

$$\mu_f = E(v_{\cdot,f}), \quad \sigma_f^2 = \operatorname{Var}(v_{\cdot,f}). \tag{19}$$

LEMMA 4.1 (RANK INVARIANCE). *Ranking $\hat{r}^*$ generated from scoring $\hat{y}^*$ results in the same ranking as $\hat{r}$ from $\hat{y}$.*

PROOF. First, the scoring function can be rewritten as:

$$\begin{aligned}
\hat{y}(l|c) &= \sum_{f=1}^{k} v_{c,f}\, v_{l,f} = \sum_{f=1}^{k} |v_{c,f}|\ \operatorname{sgn}(v_{c,f})\, (\sigma_f\, v^*_{l,f} + \mu_f) \\
&= \sum_{f=1}^{k} |v_{c,f}|\ \operatorname{sgn}(v_{c,f})\, \sigma_f\, v^*_{l,f} + \sum_{f=1}^{k} |v_{c,f}|\ \operatorname{sgn}(v_{c,f})\, \mu_f \\
&= \hat{y}^*(l|c) + \underbrace{\sum_{f=1}^{k} |v_{c,f}|\ \operatorname{sgn}(v_{c,f})\, \mu_f}_{=:b(c)}
\end{aligned}$$

The additional term $b(c)$ is independent of the item l. Thus $\hat{y}(l|c)$ is a linear transformation[8] of $\hat{y}^*(l|c)$. In general, linear transformations are rank-invariant because

$$\begin{aligned}
\hat{y}(i|c) \geq \hat{y}(j|c) &\Leftrightarrow a(c)\, \hat{y}(i|c) \geq a(c)\, \hat{y}(j|c) \\
&\Leftrightarrow a(c)\, \hat{y}(i|c) + b(c) \geq a(c)\, \hat{y}(j|c) + b(c) \\
&\Leftrightarrow \hat{y}^*(i|c) \geq \hat{y}^*(j|c)
\end{aligned}$$

[8]I.e. there exist a positive constant $a(c)$ and an arbitrary constant $b(c)$, s.th. $\hat{y}(l|c) = a(c)\, \hat{y}^*(l|c) + b(c)$.

From this follows the lemma. □

This means, if we are interested in the ranks generated by $\hat{y}$, we can also work with $\hat{y}^*$. Even though $\hat{y} \neq \hat{y}^*$, the generated rankings are equal, $\hat{r} = \hat{r}^*$.

Rank Mixture.

The representation $\hat{y}^*$ has the advantage that $p(f|c)$ can be read as a mixing probability over standardized item factors. I.e. the larger $p(f|c)$, the more important the dimension f for the specific context c. This allows to define the sampling distribution as the mixture:

$$p(j|c) := \sum_{f=1}^{k} p(f|c)\, p(j|c, f) \tag{20}$$

Due to the standardization of $v^*_{\cdot,f}$, it is reasonable to define $p(j|c, f)$ analogously to eq. (14)

$$p(j|c, f) \propto \exp(-\hat{r}^*(j|c, f)/\lambda), \tag{21}$$

where the ranking $\hat{r}^*(j|c, f)$ is generated from the context-*and* factor-dependent scoring function $\hat{y}^*(j|c, f)$. Following eq. (16), this scoring function can be defined as

$$\hat{y}^*(l|c, f) := \operatorname{sgn}(v_{c,f})\, v^*_{l,f}. \tag{22}$$

We can get rid of the standardization of $v^*_{l,f}$ and use a rank-invariant[9] but simpler function

$$\hat{y}(l|c, f) := \operatorname{sgn}(v_{c,f})\, v_{l,f}. \tag{23}$$

Note that $\hat{y}(l|c, f)$ depends on the original parameters V and not on their normalization. The scoring function $\hat{y}(l|c, f)$ has a very simple relation to its ranking $\hat{r}(l|c, f)$: the item on rank r has the r-th largest factor $v_{l,f}$ – if $\operatorname{sgn}(v_{c,f})$ is positive otherwise it has the r-th largest negative factor.

Sampling from Rank Mixture.

The formalization of the sampling distribution as a mixture model (eq. 20), results in a simple sampling algorithm for negative items:

1. Sample a rank r from a Geometric distribution.
2. Sample a factor dimension f from $p(f|c)$ (eq. 17).
3. Sort items according to $v_{\cdot,f}$, which is equivalent to an inverse ranking function $\hat{r}^{-1} : \mathbb{N} \times \{1, \ldots, k\} \to I$.
4. Return the item j on position r in the sorted list, i.e. $\hat{r}^{-1}(r|f)$ if $\operatorname{sgn}(v_{c,f}) = 1$, or $\hat{r}^{-1}(|I| - r + 1|f)$ if $\operatorname{sgn}(v_{c,f}) = -1$.

Steps 1 and 4 can be performed in $\mathcal{O}(1)$, step 2 including the computation of $p(f|c)$ in $\mathcal{O}(k) = T_{\text{pred}}$. The only computational intensive step is 3, where the factors are sorted in $O(|I| \log |I|)$.

In order to further reduce the complexity, we propose to precompute the ordering $\hat{r}^{-1}(\cdot|f)$ for each of the k factor dimensions and recompute it every couple of stochastic updates. After a single gradient step, the ranking $\hat{r}^{-1}(\cdot|f)$ changes only little and many update steps are necessary to change the precomputed ranking considerably. We propose to recompute the k rankings every $|I| \log |I|$ iterations which gave also good results in the evaluation. The described precomputation strategy has an amortized runtime of $\mathcal{O}(k)$ because every $|I| \log |I|$ iterations there is an effort of $\mathcal{O}(k\, |I| \log |I|)$. Moreover, the precomputation takes $k|I|$ additional memory for storing all $\hat{r}^{-1}(\cdot|f)$.

In total, the sampling algorithm has an amortized runtime of $\mathcal{O}(k)$ for drawing an item which is the same as the costs for a single gradient step of a MF model ($= T_{\text{pred}}$). As there is one sample for each gradient step, the computational complexity of the original SGD algorithm does not increase. Figure 4 sketches pseudocode of the improved learning algorithm.

```
1: procedure LearnAdaptiveOversampling(η, S)
2:     Randomly initialize Θ, q = 0
3:     repeat
4:         q ← q + 1
5:         if q % |I| log |I| = 0 then ▷ every |I| log |I| draws
6:             for f ∈ {1, ..., k} do
7:                 Compute r̂^{-1}(·|f)            ▷ O(|I| log |I|)
8:                 Compute σ_f^2 and μ_f           ▷ O(|I|)
9:             end for
10:        end if
11:        Draw (c, i) ∈ S uniformly
12:        Draw r from p(r) ∝ exp(−r/λ)            ▷ O(1)
13:        Draw f from p(f|c) ∝ |v_{c,f}| σ_f      ▷ O(k)
14:        j ← { r^{-1}(r|f),          if sgn(v_{c,f}) = 1
               { r^{-1}(|I| − r + 1|f), else        ▷ O(1)
15:        for θ ∈ Θ do
16:            Update θ with eq. (8).
17:        end for
18:    until convergence
19:    return Θ
20: end procedure
```

Figure 4: BPR with adaptive and context-dependent oversampling of negative items for matrix factorization.

4.2 Complex Factorization Models

In this section, it is shown how to modify the efficient algorithm in fig. 4 to learn the generic factorization machine (FM) model[10].

Let $\mathbf{x}_i \in \mathbb{R}^{p_I}$ be an arbitrary feature vector that describes item i with p_I real-valued variables and $\mathbf{x}_c \in \mathbb{R}^{p_C}$ be a feature vector that describes context c with p_C variables. This flexible representation allows to describe many different data including attributes, temporal or sequential context as well as their combinations [13]. A second order factorization machine (FM) over a feature vector $\mathbf{x} \in \mathbb{R}^p$ – here $\mathbf{x} = (\mathbf{x}_c, \mathbf{x}_i)$ and $p = p_C + p_I$ – is defined as

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{l=1}^{p} w_l\, x_l + \sum_{l=1}^{p} \sum_{l'>l}^{p} x_l\, x_{l'}\, \langle \mathbf{v}_l, \mathbf{v}_{l'} \rangle \tag{24}$$

where $w_0, w_1, \ldots, w_p, v_{1,1}, \ldots, v_{p,k}$ are the model parameters Θ. The complexity of computing the scoring function (eq. 24) is $T_{\text{pred}} = \mathcal{O}(k\, (N_Z(\mathbf{x}))$, where $N_Z(\mathbf{x})$ is the number of

[9]The proof of invariance of $\hat{y}(l|c, f)$ and $\hat{y}^*(l|c, f)$ follows from the fact that $\hat{y}(l|c, f)$ is a linear transformation of $\hat{y}^*(l|c, f)$.

[10]See [13] for details about how FM can mimic other factorization models.

non-zero values in $\mathbf{x}$. Also a SGD step has this complexity [13]. In the following, it is shown how to sample a negative item in amortized T_{pred} time.

Rank-Invariant Translation.

To derive an efficient sampling algorithm, first define for each context and item a $k+1$ dimensional factor vector $\mathbf{v}'$

$$\begin{aligned} v'_{c,f} &:= \sum_{l=1}^{p_C} v_{l,f}\, x_{c,l}, \\ v'_{c,0} &:= 1, \\ v'_{i,f} &:= \sum_{l=1}^{p_I} v_{l+p_C,f}\, x_{i,l}, \\ v'_{i,0} &:= \sum_{l=1}^{p_I} w_{l+p_C}\, x_{i,l} \\ &\quad + \sum_{l=1}^{p_I} \sum_{l'>l}^{p_I} x_{i,l+p_C}\, x_{i,l'+p_C}\, \langle \mathbf{v}_{l+p_C}, \mathbf{v}_{l'+p_C} \rangle. \end{aligned} \tag{25}$$

The transformed factors $\mathbf{v}'$ can be used to define a matrix factorization model

$$\hat{y}(i|c) := \sum_{f=0}^{k} v'_{c,f}\, v'_{i,f}. \tag{26}$$

This model is rank-invariant to the FM of eq. (24) and thus all the derivations from the MF section can be applied here (now on $\mathbf{v}'$ instead of $\mathbf{v}$). The proof of rank invariance follows directly from inserting the definition of $\mathbf{v}'$ into eq. (26) which is equal to eq. (24) except for a constant (rank-independent) term.

Efficient Algorithm.

For sampling negative items from a FM scoring model, the algorithm for matrix factorization (see fig. 4) has to be adapted slightly: The factor dimension $f \in \{0, \ldots, k\}$ is sampled from $p(f|c) \propto |v'_{c,f}|\, \sigma_f$, where V' are the translated vectors according to eqs. (25). Second, the ordering of items r^{-1} is generated not on raw factors V but on translated ones V' as well.

Note that the transformed representation V' is only used for sampling negative items. The SGD parameter learning should still be done on the original parameters $\mathbf{w}, V$.

5. EVALUATION

The properties of the proposed oversampling algorithm are studied on two real world recommender tasks. The convergence behavior of prediction quality and gradient magnitudes are investigated.

5.1 Experimental Setup

Dataset & Model.

A video dataset from the BBC[11] with play events (user-movie pairs) and the social tagging dataset from the ECML PKDD 2009 Discovery challenge[12] with tagging events (user-article-tag triples) are used. For BBC, the context is the user; for ECML'09, the context is a user-article pair. For the BBC dataset, all activities from randomly selected 100,000 users are picked. From the resulting set, one activity is randomly selected per user (only for users with at least 10 activities) into a test set S_{test} and the remaining activities form the training set S. The hyperparameters (learning rate, regularization and sampling rate) are tuned on a second random subset of 100,000 users that has been generated by the same protocol as described above. For the ECML'09 dataset, we use the official split from the challenge.

For the dyadic BBC dataset, a matrix factorization model is used. For the ternary ECML'09 dataset, the winning pairwise interaction tensor factorization (PITF) model [15] is applied. In total, both the simple MF model as well as a complex model (PITF) are investigated. Pairwise learning (see sec. 2.2) is used to learn the model parameters.

Compared Sampling Methods.

The negative item sampler is varied:

1. *Uniform sampling*: equivalent to the common BPR algorithm [14], (fig. 2).
2. *Static oversampling*: the algorithm described in section 3.1, which samples negative items from the global popularity distribution.
3. *Adaptive oversampling*: the proposed sampling distribution of section 3.2 with the algorithm of figure 4, which samples negative items with respect to the estimated ranking for this context.

Measures & Protocol.

The recommendation quality is evaluated on the test set S_{test}. For each context in the test dataset, a ranking is generated and it is measured on which ranks the items in the test set appear. The average measure over all test context is reported where the selected measures are: average precision (MAP) with a cutoff of 1000 and the half-life utility (HLU) [2]. The influence of the sampled pairs is measured by the average gradient magnitude. The gradient magnitude (eq. 9) is measured for each sampled pair and the average magnitude over a training epoch is reported. A training epoch is defined as $10 \cdot |S|$ single SGD update steps.

Experimental Reproducibility.

All reported results use a factorization dimension of $k = 64$. Results for $k = 16, 32, 128$ show similar behavior but are omitted for space reasons. The hyperparameters for BBC are: learning rate $\eta = 0.05$, random Gaussian initialization $\mathcal{N}(0, 0.01)$, regularization of 0.01 for oversampling and 0.001 for uniform sampling, $\lambda = 500$ for the Geometric distribution and 256 training epochs. The hyperparameters for ECML'09 differ in the regularization, which is 0.005 for oversampling and 0.00005 for uniform sampling, and the number of epochs is 2000. The source code of the learning algorithms can be obtained from our website[13].

5.2 Prediction Quality

Figure 5 shows the prediction quality as a function of training epochs. For comparison a non-personalized, most-popular baseline model is shown which ranks items by global popularity in the observed data S. All three personalized

[11] http://www.bbc.co.uk/

[12] http://www.kde.cs.uni-kassel.de/ws/dc09/, task 2

[13] http://www.libfm.org/

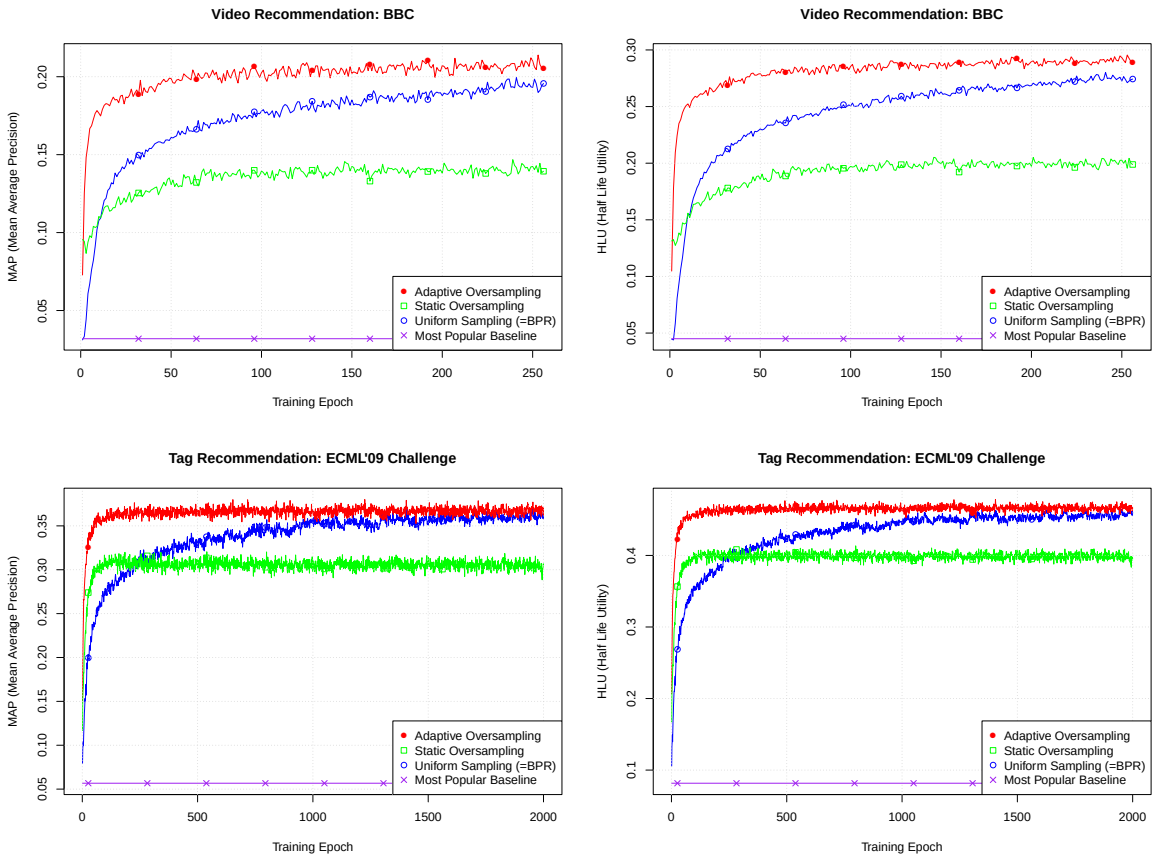

Figure 5: Quality as a function of training epochs for standard BPR with uniform sampling of negative items, static oversampling and the proposed adaptive oversampling algorithm (see fig. 4). Adaptive oversampling converges much faster than the common BPR algorithm.

models (independently of the sampling algorithm) outperform this baseline.

Adaptive Oversampling vs. Uniform Sampling (BPR).
On both datasets and both measures, the proposed adaptive oversampling has a much faster convergence and better prediction quality than the BPR algorithm. The much steeper learning curve confirms that oversampling improves learning. On both datasets, adaptive oversampling achieves after very few iterations the same accuracy as BPR achieves after hundreds of iterations. E.g. for ECML'09 the MAP of adaptive oversampling after about 50 iterations is comparable to the quality of BPR after 1000 iterations. Also for BBC, the quality of adaptive oversampling after 10 iterations is already better than BPR with 100 iterations. On the long run (esp. on the 2000 iterations for ECML'09), BPR slowly catches up with adaptive oversampling which emphasizes that uniform sampling (BPR) generates many useless training pairs.

Static Oversampling.
A second observation is that simple static oversampling using the global item distribution does not result in competitive quality. In the very first iterations, static oversampling seems to work well and outperforms standard BPR. However, convergence stops too early and on all datasets and measures, a much worse final quality than BPR or adaptive oversampling is achieved. This indicates that the oversampling distribution should adapt to the model parameters/ scoring function during learning. Adaptive oversampling shows that it is possible to sample meaningful negative items throughout training.

Runtime.
The empirical runtime on the BBC dataset for one training epoch increases from 12 seconds for BPR to 16 seconds for adaptive oversampling. This confirms that adaptive oversampling does not increase computational complexity and that the empirical overhead is only marginal.

5.3 Gradient Magnitude

Figure 6 shows the average gradient magnitudes. The first observation is that both oversampling methods are successful in increasing the gradient magnitudes, i.e. the sampled pairs result in SGD updates that actually change model parameters (eq. 8). E.g. on BBC, the average gradient of adaptive oversampling is 0.034 after 256 iterations whereas for BPR it is 0.004 – an increase of 8.5 times. For ECML'09 the numbers are 0.016 after 2000 iterations for adaptive oversampling and 0.0005 for BPR – an increase of 32 times.

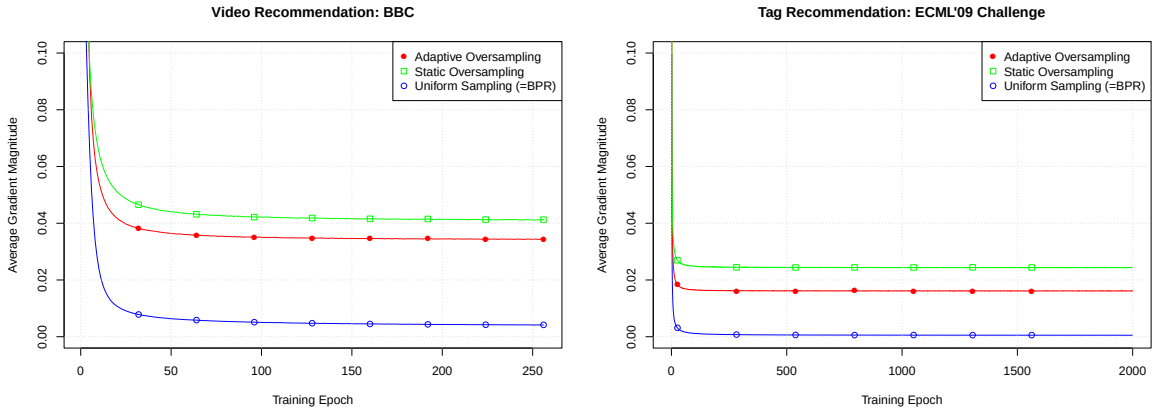

Figure 6: Average gradient magnitude as a function of training epochs. Oversampling increases the average gradient magnitude by a large factor and thus SGD updates result in faster change of model parameters. Static oversampling generates pairs with large gradient magnitudes but the pairs are not as informative as with adaptive oversampling (see figure 5).

Comparing static oversampling with adaptive oversampling, static oversampling has even a larger average gradient magnitude than adaptive oversampling throughout all epochs. However, the qualitative results in figure 5 show that static oversampling has worse prediction quality than adaptive oversampling and even than uniform sampling (on the long run). This indicates that a large gradient magnitude alone is not enough but the sampled pairs also should be informative. The qualitative results (fig. 5) indicate that adaptive oversampling fulfills both requirements.

6. RELATED WORK

Factorization models play a central role in modern recommender systems. The popular matrix factorization model (e.g. [16]), has been extended for many recommender scenarios, e.g. using implicit information [10, 18], time [11], neighborhood information [10] or attributes [20]. For context-aware settings, tensor factorization approaches have been applied (e.g. [9, 13]). Whereas these works solve the rating prediction task (i.e. regression), our work deals with item recommendation (i.e. ranking) from implicit (one-class) feedback. The item recommendation task is much harder because the optimization target is not directly observed. Hu et al. [6] and Pan et al. [12] investigate item recommendation from implicit feedback and propose to cast the one-class problem into a two class problem by imputing all non-observed values with 0 and to apply regression. This is similar to standard (algebraical) singular value decomposition (SVD) but includes confidence weights and L2 regularization. Weimer et al. [21] optimize a matrix factorization model for the ranking measure NDCG. This approach is mainly designed for data where ratings (or other ordering information on user feedback) is present. Recently, CLiMF [19] has been proposed for optimizing a matrix factorization model using implicit feedback datasets for the reciprocal rank measure. Bayesian personalized ranking (BPR) [14] is a generic optimization framework (not restricted to matrix factorization) for learning recommender systems from implicit feedback. BPR has been applied among others to matrix factorization and nearest-neighbor [14], tensor factorization for tag recommendation [15], factorized markov chains for sequential basket recommendation with taxonomy-awareness [8], social update streams [4] and relation extraction [17]. All these works on BPR use the uniform sampling assumption and thus are supposed to suffer from slow convergence. Our proposed adaptive sampler has the potential to improve all existing recommender systems that are based on BPR learning.

Gantner et al. [3] extend BPR with non-uniform sampling where the sampling probabilities for negative items are a priori given weights resulting from the problem description. In our work, the sampling probabilities are not fixed, but are generated from the current model. In the WARP algorithm [22], negative items (='annotations') are drawn repeatedly until the score of a drawn item is large enough. This algorithm increases the runtime because up to $N \leq |I|$ samples are drawn and their score is computed each time (in $\mathcal{O}(N\,T_{\text{pred}})$) before an SGD update (costs $\mathcal{O}(T_{\text{pred}})$) is performed. Moreover, within the rejection sampler, the negative items are sampled uniformly in WARP, which might need a large number of draws N before finding an item that is not in the tail.

7. CONCLUSION

In this paper, we have shown how to increase convergence of BPR-style learning algorithms. The motivation is that uniform sampling of negative items results in mostly non-informative updates. An adaptive and context-dependent sampling distribution for negative items is proposed which oversamples top ranked items. An efficient approximative sampling algorithm is developed for MF and a broad class of factorization machine models, including PITF, attribute-aware MF or sequential MF. The proposed algorithm has an amortized constant runtime. Empirically, the computational overhead over common BPR was about 33%. On two real-world datasets, the proposed sampler improves convergence by a large factor – in our experiments about 10 and 20 times faster. We expect our improved algorithm also to be valuable for existing recommender systems that are based on the BPR algorithm, such as [17, 8, 7, 1, 5].

8. ACKNOWLEDGMENTS

We would like to thank Chris Newell from the BBC for providing us with the anonymized video recommendation dataset. This work was supported by the DFG Research Training Group GK-1042 'Explorative Analysis and Visualization of Large Information Spaces', University of Konstanz.

9. REFERENCES

[1] A. Ahmed, B. Kanagal, S. Pandey, V. Josifovski, L. G. Pueyo, and J. Yuan. Latent factor models with additive and hierarchically-smoothed user preferences. In *Proceedings of the sixth ACM international conference on Web search and data mining*, WSDM '13, pages 385–394, New York, NY, USA, 2013. ACM.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco, 1998. Morgan Kaufmann.

[3] Z. Gantner, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme. Personalized ranking for non-uniformly sampled items. *Journal of Machine Learning Research Workshop and Conference Proceedings*, 2012.

[4] L. Hong, R. Bekkerman, J. Adler, and B. D. Davison. Learning to rank social update streams. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 651–660, New York, NY, USA, 2012. ACM.

[5] L. Hong, A. S. Doumith, and B. D. Davison. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *Proceedings of the sixth ACM international conference on Web search and data mining*, WSDM '13, pages 557–566, New York, NY, USA, 2013. ACM.

[6] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM 2008)*, pages 263–272, 2008.

[7] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, L. Garcia-Pueyo, and J. Yuan. Focused matrix factorization for audience selection in display advertising. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 386–397, 2013.

[8] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, J. Yuan, and L. G. Pueyo. Supercharging recommender systems using taxonomies for learning user purchase behavior. *PVLDB*, 5(10):956–967, 2012.

[9] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86, New York, NY, USA, 2010. ACM.

[10] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[11] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, New York, NY, USA, 2009. ACM.

[12] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *IEEE International Conference on Data Mining (ICDM 2008)*, pages 502–511, 2008.

[13] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[14] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009)*, 2009.

[15] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90, New York, NY, USA, 2010. ACM.

[16] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[17] S. Riedel, L. Yao, B. M. Marlin, and A. McCallum. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL '13)*, June 2013.

[18] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[19] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 139–146, New York, NY, USA, 2012. ACM.

[20] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 111–120, New York, NY, USA, 2009. ACM.

[21] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1593–1600, Cambridge, MA, 2008. MIT Press.

[22] J. Weston, S. Bengio, and N. Usunier. Wsabie: scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2764–2770. AAAI Press, 2011.