Indexable Probabilistic Matrix Factorization for Maximum Inner Product Search

Marco Fraccaro[†] **Ulrich Paquet**[‡] **Ole Winther**[†]
[†]Technical University of Denmark, Lyngby, Denmark
[‡]Microsoft Research, Cambridge, United Kingdom

Abstract

The Maximum Inner Product Search (MIPS) problem, prevalent in matrix factorization-based recommender systems, scales linearly with the number of objects to score. Recent work has shown that clever post-processing steps can turn the MIPS problem into a nearest neighbour one, allowing sublinear retrieval time either through Locality Sensitive Hashing or various tree structures that partition the Euclidian space.

This work shows that instead of employing post-processing steps, substantially faster retrieval times can be achieved for the same accuracy when inference is not decoupled from the indexing process. By framing matrix factorization to be natively indexable, so that any solution is immediately sublinearly searchable, we use the machinery of Machine Learning to best learn such a solution. We introduce *Indexable Probabilistic Matrix Factorization* (IPMF) to shift the traditional post-processing complexity into the training phase of the model. Its inference procedure is based on Geodesic Monte Carlo, and adds minimal additional computational cost to standard Monte Carlo methods for matrix factorization. By coupling inference and indexing in this way, we achieve more than a 50% improvement in retrieval time against two state of the art methods, for a given level of accuracy in the recommendations of two large-scale recommender systems.

1 Introduction

The Maximum Inner Product Search (MIPS) problem is common when matrix factorization-based recommender systems are deployed, or when inner product-based comparisons are done between the embedded vector of a query object and many potential target objects' vectors, like that of words in a neural language model. Naively, it scales linearly with the number of objects to score, and various inroads have recently been made towards sublinear retrieval time (Bachrach et al. 2014; Neyshabur and Srebro 2015; Shrivastava and Li 2014; 2015). In the parlance of a basic recommender system, user i and item j are embedded into vectors $\mathbf{u}_i$ and $\mathbf{v}_j \in \mathbb{R}^D$. To present items to a user, all $j = 1, \ldots, M$ items are scored by $\mathbf{u}_i^T \mathbf{v}_j$, and the head of their sorted list is returned. As an operation, the MIPS problem retrieves the maximizer

$$\mathsf{MIPS}(\mathbf{V}, \mathbf{u}_i) = \mathrm{argmax}_j \, \mathbf{u}_i^T \mathbf{v}_j \ , \tag{1}$$

where $\mathbf{V} \in \mathbb{R}^{D \times M}$ contains $\mathbf{v}_j$ in its columns. To score all the items requires an $\mathcal{O}(M)$ linear scan. Real systems, however, are constrained to complete such requests in a few milliseconds, and scoring and sorting all items is practically infeasible (Bachrach et al. 2014).

Instead of solving (1) directly, a number of approaches apply a separate post-processing step to $\mathbf{V}$ and $\mathbf{u}_i$, so that the MIPS is transformed to an Euclidean search on the resulting vectors. The vectors might be expanded to $D+1$ dimensions, where the triangle inequality holds, and Approximate Nearest Neighbour Search (ANNS) data structures like kd-trees (Muja and Lowe 2014) or Locality Sensitive Hashing (LSH) can be used (Neyshabur and Srebro 2015). The component of the required extra dimension is typically much larger than any in $\mathbf{v}_j$, and this artifact reduces the speedups potentially obtainable through ANNS (Bachrach et al. 2014). Alternatively, a transformation to $D + m$ dimensions so that (1) becomes an Euclidian distance search in the transformation's limit $m \to \infty$ also allows LSH to be used (Shrivastava and Li 2014; 2015), although with weaker theoretical guarantees than those found in (Neyshabur and Srebro 2015). Both these post-processing steps give sublinear retrieval time by transforming the problem to one that is *geometrically indexable*, by which we mean one that allows $\mathcal{O}(\log M)$ or sublinear retrieval time.

As $\mathbf{V}$ and $\mathbf{u}_i$ originate from a statistical model, an alternative view is not to rely on retro-actively adjusting a model for faster retrieval, but instead to frame the model to operate on *natively* geometrically indexable vectors, and utilise the tools provided by an inference framework to find such a geometrically indexable embedding. Through the simple constraint of requiring all item vectors $\mathbf{v}_j$ to have a fixed norm, say $\|\mathbf{v}_j\| = 1$, $\mathsf{MIPS}(\mathbf{V}, \mathbf{u}_i)$ is equivalent to a minimization over Euclidean distances, i.e. a Nearest Neighbour Search (NNS):

$$\mathsf{NNS}(\mathbf{V}, \mathbf{u}_i) = \underset{j}{\mathrm{argmin}} \, \|\mathbf{u}_i - \mathbf{v}_j\|^2 \overset{(*)}{=} \underset{j}{\mathrm{argmax}} \, \mathbf{u}_i^T \mathbf{v}_j \, . \tag{2}$$

The step denoted with $(*)$ follows from all $\|\mathbf{v}_j\|$ being equal. To exploit this, we introduce *Indexable Probabilistic Matrix Factorization* (IPMF) in Section 2. It is an extension of Bayesian Probabilistic Matrix Factorization (BPMF) (Salakhutdinov and Mnih 2008a) that is made inherently geometrically indexable by letting $\{\mathbf{v}_j\}$ be *a priori* governed by a Von Mises-Fisher distribution which puts zero probability mass on $\mathbf{v}_j$ with $\|\mathbf{v}_j\| \neq 1$. The prior ensures that (1) can be directly addressed through ANNS, for any data set, point estimate or posterior parameter sample. The inference procedure for the item vectors will be based on *Geodesic Monte Carlo* (Byrne and Girolami 2013), that allows efficient sampling from distributions that are constrained to a *manifold*, represented here by a hypersphere. Related techniques to transform a MIPS into a NNS are discussed in Sec. 3 and compared to our approach in Sec. 4 using two large scale data sets of movies and music recommendations. We will show that the restriction of the model to only embed items onto a manifold empirically gives equivalent recommendation quality in terms of predicting users' ratings of items. The unconventional prior adds a small computational overhead in the offline (inference) part when compared to BPMF, but allows a faster and more accurate online (retrieval) component with respect to state of the art methods.

2 Indexable Probabilistic Matrix Factorization

It is possible to transform a MIPS problem into a NNS one by fixing the norm of the item vectors in a Probabilistic Matrix Factorization (PMF) model (Salakhutdinov and Mnih 2008b), as shown in (2). In standard Monte Carlo methods for matrix factorization this constraint can be imposed with a small modification to a PMF model and its inference procedure, and will leave the quality of the recommendations intact. In this work we extend BPMF, but the main ideas can be applied to any matrix factorization model.

Although a modern recommender system operates on implicit and varied user feedback, its kernel is commonly built around a sparse $N \times M$ rating matrix $\mathbf{R}$, whose element r_{ij} contains user i's rating of item j. A low-rank matrix factorization $\mathbf{R} \approx \mathbf{U}^T\mathbf{V}$ embeds user i into a column vector $\mathbf{u}_i$ in $\mathbf{U} \in \mathbb{R}^{D\times N}$, and item j into a latent vector $\mathbf{v}_j$. BPMF models $r_{ij} = \mathbf{u}_i^T\mathbf{v}_j + \varepsilon_{ij}$ as a real-valued random variable, where ε_{ij} is additive white Gaussian noise with a variance of $1/\alpha$. As a generative model, user and item vectors are a priori dictated by Gaussian priors $p(\mathbf{u}_i|\mathbf{\Theta_u}) = \mathcal{N}(\mathbf{u}_i; \boldsymbol{\mu}_u, \mathbf{\Lambda}_u^{-1})$ and $p(\mathbf{v}_j|\mathbf{\Theta_v}) = \mathcal{N}(\mathbf{v}_j; \boldsymbol{\mu}_v, \mathbf{\Lambda}_v^{-1})$, where $\mathbf{\Theta_u} = \{\boldsymbol{\mu}_u, \mathbf{\Lambda}_u^{-1}\}$ and $\mathbf{\Theta_v} = \{\boldsymbol{\mu}_v, \mathbf{\Lambda}_v^{-1}\}$. Conjugate Gaussian-Wishart hyperpriors are used for the unknown shared means and precision matrices. The expected rating is then approximated with Markov Chain Monte Carlo (MCMC) techniques, using a Gibbs sampler to obtain samples from the posterior distribution over user and item vectors; see (Salakhutdinov and Mnih 2008a) for details.

2.1 Manifold restrictions

An indexable version of the BPMF, that we call *Indexable Probabilistic Matrix Factorization* (IPMF), can be obtained by replacing the Gaussian prior for the item vectors with a Von Mises-Fisher prior. The Von Mises-Fisher distribution is defined over points on a hypersphere, and therefore enforces the fixed-norm constraint that transforms the MIPS problem to a NNS. Due to its simplicity, this probability density is widely used in directional statistics (Dhillon and Sra 2003). Given a mean direction vector $\boldsymbol{\mu}_v$ with unit length and a concentration parameter $\kappa \geq 0$, the Von Mises-Fisher distribution $vMF(\mathbf{v}_j; \boldsymbol{\mu}_v, \kappa)$ is defined as

$$p(\mathbf{v}_j|\mathbf{\Theta_v}) = Z_D(\kappa)\, e^{\kappa \boldsymbol{\mu}_v^T \mathbf{v}_j}\, \mathcal{I}_{\mathcal{S}_D}(\mathbf{v}_j)\ , \tag{3}$$

where $\mathbf{\Theta_v} = \{\boldsymbol{\mu}_v, \kappa\}$. $\mathcal{S}_D = \{\mathbf{v} \in \mathbb{R}^D \mid \|\mathbf{v}\| = 1\}$ represents the D dimensional unit hypersphere, $\mathcal{I}_A$ is the indicator function of the set A, and the normalizing constant $Z_D(\kappa)$ is given by $Z_D(\kappa) = \kappa^{D/2-1}/(2\pi)^{D/2} I_{D/2-1}(\kappa)$, where I_p is the modified Bessel function of the first kind and order p. In the Von Mises-Fisher prior, higher values of κ increase the concentration of the distribution around the mean direction $\boldsymbol{\mu}_v$, while $\kappa = 0$ gives an uniform distribution over the D dimensional unit hypersphere[1]. In our experiments we will use $\kappa = 0$, assuming therefore that all the directions have the same prior probability. Like BPMF, the ratings are assumed to be conditionally independent given the user and item vectors, so that the likelihood decomposes over all observations through $p(\mathbf{R}|\mathbf{U}, \mathbf{V}, \alpha) = \prod_{(i,j)} \mathcal{N}(r_{ij}; \mathbf{u}_i^T\mathbf{v}_j, \alpha^{-1})$.[2]

Given the structural similarity between BPMF and IPMF, the Gibbs samplers for BPMF and IPMF only differ in their conditional distributions of the item vectors. We refer the reader to (Salakhutdinov and Mnih 2008a) for other Gibbs steps, and only focus on conditional densities pertaining to the manifold constraint. For both models the log-likelihood is a quadratic function in $\mathbf{v}_j$:

$$\log p(\mathbf{r}_j|\mathbf{v}_j, \mathbf{U}_j, \alpha) \propto -\frac{\alpha}{2}(\mathbf{v}_j^T\mathbf{U}_j\mathbf{U}_j^T\mathbf{v}_j - 2\mathbf{v}_j^T\mathbf{U}_j\mathbf{r}_j)\ ,$$

where $\mathbf{U}_j$ is the restriction of $\mathbf{U}$ to the columns that are indexed by users that rated item j, and $\mathbf{r}_j$ contains the corresponding ratings. The prior in (3) introduces fixed-norm constraints to IPMF's conditional density $p(\mathbf{v}_j|\mathbf{r}_j, \mathbf{U}_j, \alpha, \mathbf{\Theta_v})$. Our choice of $\kappa = 0$ implies that the log-posterior is

$$\log p(\mathbf{v}_j|\mathbf{r}_j, \mathbf{U}_j, \alpha, \mathbf{\Theta_v}) \propto -\frac{\alpha}{2}(\mathbf{v}_j^T\mathbf{U}_j\mathbf{U}_j^T\mathbf{v}_j - 2\mathbf{v}_j^T\mathbf{U}_j\mathbf{r}_j) \quad \text{such that} \quad \|\mathbf{v}_j\| = 1\ . \tag{4}$$

The gradient of this constrained function with respect to the random vector $\mathbf{v}_j$ is given by:

$$\nabla_{\mathbf{v}_j} \log p(\mathbf{v}_j|\mathbf{r}_j, \mathbf{U}_j, \alpha, \mathbf{\Theta_v}) = -\alpha(\mathbf{U}_j\mathbf{U}_j^T\mathbf{v}_j - \mathbf{U}_j\mathbf{r}_j)\ .$$

As the gradient of the (unnormalized) log-posterior can be efficiently computed it is possible to sample posterior item vectors using the recently introduced *Geodesic Monte Carlo* (GMC) (Byrne and Girolami 2013).

[1] κ therefore plays a role similar to that of the precision parameter (inverse variance) in a Gaussian distribution.

[2] When ratings are modelled with additive user and item biases c_i and d_j through $r_{ij} = \mathbf{u}_i^T\mathbf{v}_j + c_i + d_j + \varepsilon_{ij}$, the definitions $\widetilde{\mathbf{u}}_i \triangleq [\mathbf{u}_i; 1]$ and $\widetilde{\mathbf{v}}_j \triangleq [\mathbf{v}_j; d_j]$ yield $\operatorname{argmax}_j(r_{ij}) = \operatorname{argmax}_j(\mathbf{u}_i^T\mathbf{v}_j + d_j) = \operatorname{argmax}_j(\widetilde{\mathbf{u}}_i^T\widetilde{\mathbf{v}}_j)$. IPMF requires a fixed-norm constraint on $\widetilde{\mathbf{v}}_j$ to enforce sublinear NNS. Biases are not considered in our analysis for simplicity in the exposition.

2.2 Geodesic Monte Carlo

Geodesic Monte Carlo (GMC) is an extension of Hamiltonian Monte Carlo (HMC) (Neal 2010) to distributions that are themselves defined over a hypersphere (or, more generally, over a manifold embedded in the Euclidean space for which an explicit form of the geodesics are known). It builds upon the same ideas that underpin Riemann manifold Hamiltonian Monte Carlo (RMHMC) (Girolami and Calderhead 2011). For uncluttered notation we drop the dependence of $\mathbf{v}_j$ on j below, and denote the target distribution by $p(\mathbf{v})$.

The GMC algorithm considers the Hamiltonian defined on the manifold (a hypersphere in our case) whose local geometric properties are described by the metric tensor $\mathbf{G}(\mathbf{v})$. Samples from $p(\mathbf{v})$ are obtained by interpreting $\mathbf{v}$ as the position of a particle that moves along the manifold according to the corresponding Hamiltonian dynamics, by introducing a position-dependent auxiliary momentum variable $\mathbf{p}$ drawn from $\mathcal{N}(\mathbf{p}; \mathbf{0}, \mathbf{G}(\mathbf{v}))$. The Hamiltonian is then given by $H(\mathbf{v},\mathbf{p}) = -\log p(\mathbf{v}) + \frac{1}{2}\mathbf{p}^T\mathbf{G}(\mathbf{v})^{-1}\mathbf{p}$. When the target distribution is defined on a manifold for which the form of the geodesics is known, Hamilton's equations are conveniently solved by "splitting the Hamiltonian" (Neal 2010). This is done by writing

$$H(\mathbf{v},\mathbf{p}) = H^{[1]}(\mathbf{v},\mathbf{p}) + H^{[2]}(\mathbf{v},\mathbf{p}) \tag{5}$$

where $H^{[1]}(\mathbf{v},\mathbf{p}) = -\log p(\mathbf{v})$ and $H^{[2]}(\mathbf{v},\mathbf{p}) = \frac{1}{2}\mathbf{p}^T\mathbf{G}(\mathbf{v})^{-1}\mathbf{p}$. Considering each term in (5) as a distinct Hamiltonian we alternately simulate their dynamics for some time step ϵ to obtain new proposals (as for HMC, ϵ should be small enough to have a low discretization error). To allow larger moves and suppress random walk behaviour Hamilton's equations are solved for l steps at each iteration[3] (the number of "leapfrog" steps in HMC). When solving Hamilton's equations for $H^{[1]}(\mathbf{v},\mathbf{p})$, the absence of a kinetic term leads to a linear update of the momentum $\mathbf{p}$: $\mathbf{p} \leftarrow \mathbf{p} + \frac{\epsilon}{2}\nabla_{\mathbf{v}} \log p(\mathbf{v})$. On the other hand, when considering $H^{[2]}(\mathbf{v},\mathbf{p})$ the trajectory followed will be a geodesic. No potential term is acting on the particle; it will therefore move in the manifold following the path determined by the bending of the surface. For a hypersphere, the geodesics are given by the *great circles*, the intersection of the hypersphere and any hyperplane passing through the center of the sphere.

In Fig. 1 we show a comparison on a toy example with $D = 3$ between GMC and the standard Metropolis-Hastings (MH) algorithm that uses a Von Mises-Fisher proposal distribution whose samples are drawn with the method presented in (Wood 1994). In Fig. 1 GMC converges after only one sample, and mixes much faster than MH, for which a long burn-in phase is needed. If we consider the sampling of the item vectors in the experiments presented in Sec. 4, GMC achieves a few orders of magnitude speedup on MH.

3 Related approaches

In recent years the need for fast recommendations in large scale systems has stimulated research on how to solve a

[3]In our simulations $l = 10$ and $\epsilon = 0.002$ gave good performances

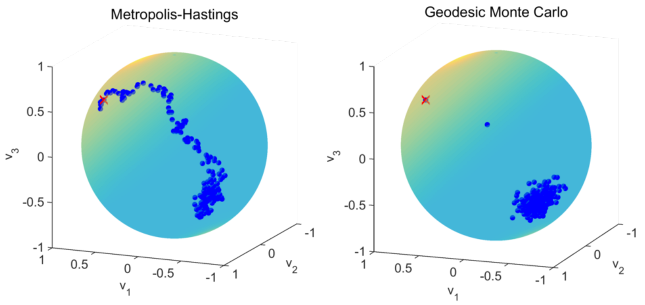

Figure 1: Convergence from a random starting point (the red cross) of MH with proposals on the sphere and GMC for $D = 3$. GMC reaches the high density region much faster than MH, and here converges to the stationary distribution after one burn-in sample.

MIPS in sublinear time. Due to the complexity of the problem, the early attempt of Khoshneshin and Street (2010) proposes a latent factor model in which predictions are no longer based on a maximization over inner products, but on a distance minimization problem, therefore enabling the use of data structures for ANNS. As their model is not based on matrix factorization techniques, their ideas are not applicable to many of the models presented in the literature and used in real world recommendation engines (Koren 2009; Koren, Bell, and Volinsky 2009; Lawrence and Urtasun 2009; Mackey, Weiss, and Jordan 2010; Paquet, Thomson, and Winther 2012; Porteous, Asuncion, and Welling 2010; Paquet and Koenigstein 2013). More recent works have instead used Euclidian transformations that increase the dimensionality of the output of an existing matrix factorization model (such as BPMF in our experiments) to pass from a maximization over inner products to a distance minimization problem.

Bachrach et al. (2014) extend user and item vectors with one extra dimension as

$$\mathbf{u}_i^\star = [\,\mathbf{u}_i\,;\,0\,] \qquad \mathbf{v}_j^\star = \left[\,\mathbf{v}_j\,;\,\sqrt{\phi^2 - \|\mathbf{v}_j\|^2}\,\right], \tag{6}$$

and show that $\mathsf{NNS}(\mathbf{V}^\star, \mathbf{u}_i^\star) = \mathsf{MIPS}(\mathbf{V}, \mathbf{u}_i)$, where $\phi \triangleq \max_j \|\mathbf{v}_j\|$ and $\mathbf{V}^\star \in \mathbb{R}^{D+1\times M}$ is the matrix that contains $\mathbf{v}_j^\star$ in its columns. Experiments on large-scale recommender systems indicate considerable performance improvements of this method compared to the space-partitioning trees of Koenigstein, Ram, and Shavitt (2012) and Ram and Gray (2012). An alternative transformation is that of Shrivastava and Li (2014),

$$\begin{aligned} \mathbf{u}_i' &= [\,\mathbf{u}_i\,;\,1/2\,;\,1/2\,;\ldots;\,1/2\,] \\ \mathbf{v}_j' &= \left[\,\mathbf{v}_j\,;\,\|\mathbf{v}_j\|^2\,;\,\|\mathbf{v}_j\|^4\,;\ldots;\,\|\mathbf{v}_j\|^{2^m}\,\right], \end{aligned} \tag{7}$$

who show that for $m \to \infty$ (typically $m \geq 3$ is enough) we have $\mathrm{argmax}_j\, \mathbf{u}_i^T\mathbf{v}_j \simeq \mathrm{argmin}_j\, \|\mathbf{u}_i' - \mathbf{v}_j'\|$. Their nearest neighbour search is performed with Locality Sensitive Hashing (Andoni and Indyk 2008), which uses hashing functions with the property that vectors that are close to each other will have a high probability of collision. This method is improved in (Shrivastava and Li 2015), which exploits a new transformation similar to (7) such that when $m \to \infty$ the MIPS problem can be expressed as a cosine similarity search[4] that can be efficiently solved using LSH with signed random projections (Charikar 2002). Finally, Neyshabur and Srebro (2015) propose the *simple-LSH* method, that combines the independently rederived transformation in (6) with a LSH that uses signed random projections hashing functions. Theoretical comparisons show that simple-LSH outperforms the methods in (Shrivastava and Li 2014; 2015) in terms of quality of hashing, in addition to giving better empirical performances and stronger guarantees. In our experiments we will therefore only compare IPMF with the state of the art methods of (Bachrach et al. 2014; Neyshabur and Srebro 2015), by extending the latent vectors of BPMF using (6) and testing different data structures for ANNS.

We note in particular that the new vectors $\mathbf{v}_j^\star$ in (6) all have the *same* norm ϕ, and like those of IPMF, also lie on a *hypersphere*. The presence of ϕ in only the extra dimension is such that the extra component typically has a much bigger value than the rest of the latent variables (see discussion in (Bachrach et al. 2014)), meaning that the item vectors will be concentrated in one region of the hypersphere. While this is not an issue when looking for exact nearest neighbours, this data imbalance will affect the retrieval accuracy in the down-stream approximate NNS[5]; see Sec. 4.2. As IPMF natively restricts item vectors to a hypersphere, all vector components are used equally in determining the items' latent embeddings, with no component consistently being substantially larger. This holds practical benefits for the accuracy of ANNS data structures.

4 Results

We evaluate our model on the two biggest publicly available data sets for collaborative filtering. The **Netflix** data set (Bennett and Lanning 2007) contains more than 100 million ratings from 1 to 5 stars (integers) given to 17770 movies by 480189 users. The **Yahoo! Music** data set (Dror et al. 2011) contains around 283 million ratings given by slightly more that 1 million users to 624961 music items, with integer ratings between 0 and 100. To gauge the quality of the recommendations given by IPMF, we show that it is practically equivalent to that of BPMF in terms of *Root Mean Squared Error (RMSE)* when predicting ratings in a test set. Finally, we compare our method against state of the art techniques in terms of *speedup*, defined as the ratio between the time taken to make recommendations solving the MIPS problem and the time taken with ANN. To measure the trade-off between the quality of the top K recommendations and the speedup that one can obtain with the ANNS, we will use the same quantities defined in (Bachrach et al. 2014):

$$\text{Precision@K} \triangleq \frac{|L_{opt} \cap L_{approx}|}{K} \, ,$$

$$\text{RMSE@K} \triangleq \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(L_{opt}(k) - L_{approx}(k)\right)^2} \, ,$$

where L_{approx} and L_{opt} represent the list of the top K approximated (retrieved) and optimal recommendations, sorted in decreasing order of predicted rating. The optimal recommendations are those obtained by a full $\mathcal{O}(M)$ search. Precision@K measures how many of the top K approximate recommendations are among the optimal K ones, whereas the RMSE@K measures the difference in the quality of the recommendations (in terms of predicted rating) of the items in the lists L_{approx} and L_{opt}.

	Netflix			
	BPMF		*IPMF*	
D	RMSE	Time (h)	RMSE	Time (h)
20	0.9059	0.91	0.9067	1.04
50	0.8985	1.54	0.8997	1.79
100	0.8951	3.25	0.8976	3.60
	Yahoo! Music			
20	24.902	3.17	25.063	8.95
50	24.695	5.33	24.877	14.16
100	24.677	10.42	24.856	31.45

Table 1: Test RMSE and training times (in hours) for different values of the number D of latent features. For both models 150 posterior samples were drawn.

4.1 Root Mean Squared Error

The main purpose of this section is to understand how the choice of the Von Mises-Fisher prior affects recommendations, as it would be useless to speed up retrieval if the quality of the final recommendations degraded. The Von Mises-Fisher prior of IPMF imposes much stronger constraints on the item vectors than the Gaussian prior of BPMF, as it assigns zero probability to all points but those on a unit hypersphere (hence we have only $D-1$ degrees of freedom).

Quality. The Netflix data set is accompanied by a test set of around 1.4 million predictions, whereas the Yahoo! Music test set has slightly more than 6 million elements. Table 1 draws a comparison between the RMSE and the training times (in hours) achievable with BPMF and IPMF on both data sets for different D. Despite restricting $\mathbf{v}_j$ to a hypersphere, the model's predictive accuracy is not significantly affected. To gain more insight in the difference between the results of BPMF and IPMF, we show in Fig. 2 the RMSEs computed for groups of items with a similar number of viewers. This is useful in discerning whether a specific group of items is particularly penalized by IPMF. Only the analysis for $D = 20$ is shown, as for $D = 50$ and $D = 100$ we obtained similar trends. From the figure we notice that IPMF tends to penalize more items with a small number of view-

[4]Due to the fixed norm constraints this also holds for IPMF.

[5]To mitigate this problem Bachrach et al. (2014) use Principal Component Analysis (PCA) to rotate the expanded $\{\mathbf{v}_j^\star\}$.

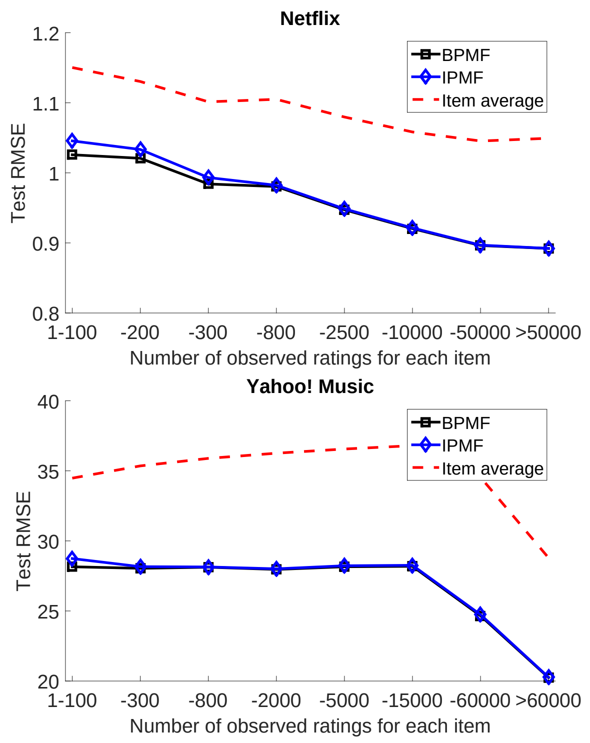

Figure 2: Test RMSE for the different groups of items ($D = 20$). The RMSE of a naive algorithm that assigns to each item its average rating (item average), is also plotted.

ers.[6] The difference between IPMF and BPMF's test RMSE is negligible, but noticeable for items that received very little user feedback. In practice this is ignorable, as in a live system items with only a few views or ratings are often not considered as candidates for recommendation, in order to avoid potentially poor user experiences due their little training data and large uncertainty associated with their predictions. An analysis similar to the one of Fig. 2, but considering different groups of users (separated according to the number of items viewed), showed almost overlapping lines for BPMF and IPMF. This result is crucial, as it implies that no specific type of user is penalized by the modelling choice.

Training times. For all the simulations in Table 1 we used a machine with an 8-core processor and 64GB of RAM. The differences in training times for a given value of D are ascribed to IPMF needing GMC to sample the items vectors, whereas in BPMF the items are Gibbs-sampled from a Gaussian distribution. The Netflix data set catalogue is not too large, and there is a very small difference in computational cost between BPMF and IPMF. Because of the huge number of items in the Yahoo! Music catalog (more than 600 thousand), IPMF is almost 3 times slower than BPMF. As the architectures of real systems contain an offline (inference) and online (retrieval) component, the increase in offline computation time may be allowable if this guarantees optimal online performance.

4.2 Retrieval speedup

State of the art methods apply a post-processing step that *increases* the dimensionality of the computed latent vectors (BPMF in our experiments) to transform a MIPS to a NNS; see Sec. 3. Our proposed approach, on the other hand, constructs models that are natively indexable (such as IPMF). In our experiments we only compare IPMF to BPMF *with* an additional dimension, introduced independently by Bachrach et al. (2014) and Neyshabur and Srebro (2015). We will refer to it as BPMF+ (BPMF plus post-processing); see (6). As Neyshabur and Srebro (2015) show that their simple-LSH algorithm outperforms the two methods of (Shrivastava and Li 2014; 2015), this will be the only one considered in our simulations. As suggested by Bachrach et al. (2014), a Principal Component Analysis transform is applied to the BPMF+ user and item vectors.[7]

Once the maximization over inner products (the recommendation step) is expressed as a distance minimization problem, it is then possible to speed up the retrieval time by doing Approximate Nearest Neighbor Search (ANNS), using any of the data structures developed in the literature. We experimentally compare against *randomized kd-trees*, *priority search (hierarchical) k-means trees* and *locality sensitive hashing*, as they were shown to be very effective when dealing with high dimensional data (Muja and Lowe 2009; 2014). For the first two data structures we used the implementation in the well known Fast Library for Approximate Nearest Neighbor (FLANN) (Muja and Lowe 2014), and both data structures explore the tree in a *best-bin-first* manner. LSH uses the *signed random projection* hashing scheme (Charikar 2002), so that the combination of BPMF+ and LSH is equivalent to the *simple-LSH* method of (Neyshabur and Srebro 2015). The LSH algorithm implementation is from (Aly, Munich, and Perona 2011).

Our comparisons use ANNS data structures on one posterior MCMC sample of $\mathbf{V}$ for BPMF+ and IPMF. Given one indexed $\mathbf{V}$-sample, effects like freshness of recommendations can be addressed by retrieving different sets of items using different $\mathbf{u}_i$ samples from the conditional $\mathbf{u}_i|\mathbf{V}$. Alternatively, multiple posterior $\mathbf{V}$-samples can be indexed separately or together. These combinations are beyond the scope of this work, and not addressed here.

Comparative Speedups. Fig. 3 shows the speedup obtainable with IPMF and BPMF+ using the data structures for ANNS previously introduced. We consider $D = 20$ latent features and focus on top 10 recommendations, hence as measures of quality we use the previously defined Precision@10 and RMSE@10. Like Table 1, we also tested for $D = 50$ and $D = 100$; the trends were similar, and are

[6]At first sight, these results may seem inconsistent with the RMSEs shown in Table 1, where for $D = 20$ the test RMSE obtained with BPMF is extremely close to the one obtained with IPMF. Items with a few viewers also have only a few ratings in the test set, hence the total test RMSE is only slightly influenced by them.

[7]A kd-tree built on data transformed with a PCA is also known as *PCA-tree*

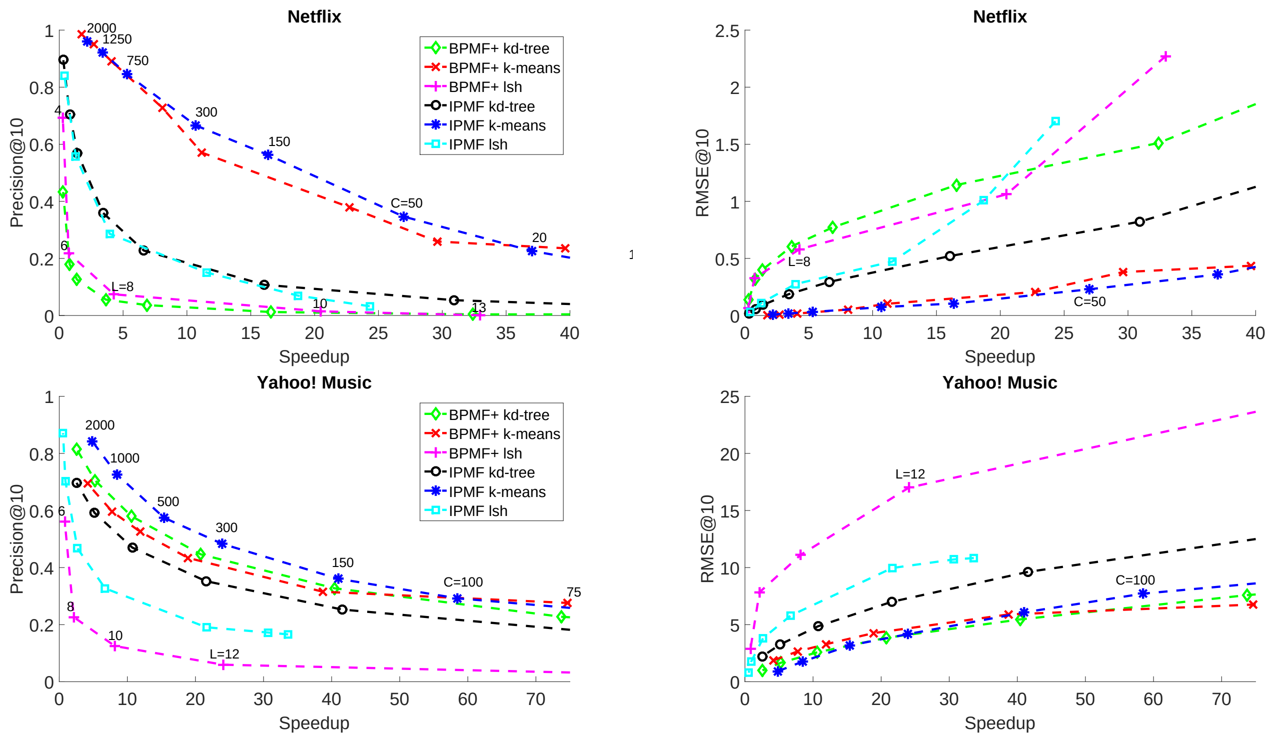

Figure 3: Performance analysis for the ANNS step of the recommender system ($D = 20$), using the mean values over all the users of Precision@10 (higher is better), RMSE@10 (lower is better) and speedup (higher is better). For the tree structures we vary the maximum number C of visited internal nodes allowed during the exploration of the tree (for the priority search k-means tree used for IPMF the figures are annotated with C). The priority search k-means tree has 32 and 64 cluster centers for the Netflix and Yahoo! Music data sets respectively. LSH uses $T = 10$ hash tables and different hash key lengths L; the figures are annotated with some settings of L. (We additionally tried T in $\{1, 3, 5, 10, 20, 30, 50, 75, 100\}$ on optimized code, and of these, $T = 10$ was the best setting in this instance.)

not included here for brevity. A value for Precision@10 in the range from 0.4 to 0.8 gives a good compromise between speedup obtainable and quality of the recommendations. In both tree structures this trade-off can be expressed by trying different values for the maximum number C of internal nodes visited during the NNS, as Muja and Lowe (2014) suggest. For LSH, we tested different hash key lengths L.

We see in Fig. 3 that for both data sets the combination of IPMF and the priority search k-means tree is by far the best performing method in the interesting region, both in terms of Precision@10 and RMSE@10. For the Netflix data set, we see for example that to obtain a Precision@10 value of 0.6 our approach is around 50% faster than the second best performing method, where a priority search k-means tree is used for BPMF+. Randomized kd-trees and LSH also work better with IPMF than with BPMF+, but they are still both largely outperformed by priority search k-means trees. Similar trends can be noticed for the Yahoo! Music data set, where the priority search k-means tree with our IPMF method is for example at least 50% faster than all the other methods for a Precision@10 of 0.6. For the BPMF+ the randomized kd-trees now allow better performances than the priority search k-means tree.

5 Conclusion

In this work we complement the work of (Bachrach et al. 2014; Neyshabur and Srebro 2015; Shrivastava and Li 2014; 2015), by developing a novel approach to solve a MIPS in sublinear time for large-scale recommender systems. We have shown that it is possible to simply extend existing models to have natively geometrically indexable vectors, introducing the *Indexable Probabilistic Matrix Factorization* as an extension of the BPMF. The GMC algorithm allows efficient inference, and the results on two different large-scale data sets show that the required embedding on a manifold does not affect the quality of the recommendations and leads to significant improvements in retrieval time with respect to state of the art techniques. In this work, the training phase of our model and the construction of the data structure for ANNS are still done separately, but IPMF could potentially combine inference and indexing in a natural way.

Acknowledgements

Marco Fraccaro is supported by Microsoft Research through its PhD Scholarship Programme.

References

Aly, M.; Munich, M.; and Perona, P. 2011. Indexing in large scale image collections: scaling properties and benchmark. In *IEEE Workshop on Applications of Computer Vision (WACV)*.

Andoni, A., and Indyk, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51(1):117–122.

Bachrach, Y.; Finkelstein, Y.; Gilad-Bachrach, R.; Katzir, L.; Koenigstein, N.; Nice, N.; and Paquet, U. 2014. Speeding up the Xbox recommender system using a Euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14. ACM.

Bennett, J., and Lanning, S. 2007. The Netflix prize. In *In KDD Cup and Workshop*.

Byrne, S., and Girolami, M. 2013. Geodesic Monte Carlo on embedded manifolds. *Scandinavian Journal of Statistics* 40(4):825–845.

Charikar, M. S. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02. ACM.

Dhillon, I. S., and Sra, S. 2003. Modeling data using directional distributions. Technical report, University of Texas.

Dror, G.; Koenigstein, N.; Koren, Y.; and Weimer, M. 2011. The Yahoo! Music dataset and KDDcup'11. In *Proceedings of KDDCup 2011*.

Girolami, M., and Calderhead, B. 2011. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J. of the Royal Statistical Society, Series B (Methodological)*.

Khoshneshin, M., and Street, W. N. 2010. Collaborative filtering via Euclidean embedding. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10. New York, NY, USA: ACM.

Koenigstein, N.; Ram, P.; and Shavitt, Y. 2012. Efficient retrieval of recommendations in a matrix factorization framework. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, 535–544. ACM.

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

Koren, Y. 2009. The BellKor solution to the Netflix Grand Prize.

Lawrence, N. D., and Urtasun, R. 2009. Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09. ACM.

Mackey, L. W.; Weiss, D. J.; and Jordan, M. I. 2010. Mixed membership matrix factorization. In Fürnkranz, J., and Joachims, T., eds., *ICML*, 711–718. Omnipress.

Muja, M., and Lowe, D. G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*.

Muja, M., and Lowe, D. G. 2014. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36.

Neal, R. M. 2010. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo* 54.

Neyshabur, B., and Srebro, N. 2015. On symmetric and asymmetric LSHs for inner product search. In *Proceedings of the 32nd Annual International Conference on Machine Learning*, ICML '15. ACM.

Paquet, U., and Koenigstein, N. 2013. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd International Conference on World Wide Web*, 999–1008.

Paquet, U.; Thomson, B.; and Winther, O. 2012. A hierarchical model for ordinal matrix factorization. *Statistics and Computing* 22(4):945–957.

Porteous, I.; Asuncion, A. U.; and Welling, M. 2010. Bayesian matrix factorization with side information and dirichlet process mixtures. In Fox, M., and Poole, D., eds., *AAAI*. AAAI Press.

Ram, P., and Gray, A. G. 2012. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, 931–939. New York, NY, USA: ACM.

Salakhutdinov, R., and Mnih, A. 2008a. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the International Conference on Machine Learning*, volume 25.

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

Shrivastava, A., and Li, P. 2014. Asymmetric LSH (ALSH) for sublinear time Maximum Inner Product Search (MIPS). In *Advances in Neural Information Processing Systems 27*.

Shrivastava, A., and Li, P. 2015. Improved asymmetric locality sensitive hashing (ALSH) for Maximum Inner Product Search (MIPS). In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

Wood, A. T. 1994. Simulation of the von Mises Fisher distribution. *Communications in Statistics - Simulation and Computation* 23(1):157–164.