Kernelizing Probabilistic Matrix Factorization to Enhance Music Recommendation

Albert Jan

Abstract

We apply kernelized probabilistic matrix factorization (KPMF) to model the rating a user assigns a music artist in the hetrec2011-lastfm-2k dataset. KPMF introduces complexity to probabilistic matrix factorization (PMF) by capturing covariances between the latent vectors learned, rather than assuming they are i.i.d. We construct two covariance matrices – one for users and one for artists – a priori using side information provided by the dataset, and incorporate them into the generative process. To construct the user covariance matrix, we leverage the social network graph provided and apply the Commute Time (CT) graph kernel, which embeds nodes (users) in an inner product space. To construct the artist covariance matrix, we embed each artist with a vector that stores tags assigned to that artist by users, and then apply the Radial Basis Function (RBF) kernel on two artists' embeddings to obtain their covariance. We learn the latent vectors via MAP inference. The CT kernel and RBF kernel both yield slightly lowest RMSEs than PMF, but combining the two is minimally helpful.

I. Introduction

In building recommendation systems, one often seeks to model what a user rates an item on some numeric scale. As a simple application, these items can be sorted by rating for each user, with the highest rated ones being recommended. We often formulate this problem as follows: Given N users and M items, we form a ratings matrix $R \in \mathbb{R}^{N\times M}$, where $R_{ij} \in \mathbb{R}$ denotes the rating user i gives item j. We seek to fill in missing entries, using existing ones as well as possibly domain-specific extraneous information.

Several popular data imputation techniques revolve around matrix factorization. Such algorithms learn a latent vector for every user and every item, and then take a rating to be the inner product of its corresponding user and item latent vectors. By constructing one matrix whose rows store the users' latent vectors, and another matrix whose columns store the items' latent vectors, we obtain a factorization of the partially observed R. However, such techniques often suffer from data sparsity issues in real-world scenarios. In fact, for most large commercial recommendation systems, fewer than 1% of entries are actually filled [1]. Thus, we often look beyond observed ratings and incorporate external data sources, referred to as "side information", into the algorithm. While many algorithms for factorizing the ratings matrix exist, we discuss probabilistic ones — models which assume some underlying generative process for the latent vectors and ratings.

A. *Probabilistic Matrix Factorization and Variants*

Given N users and M items, we form the ratings matrix $R \in \mathbb{R}^{N\times M}$. Let k be the dimension of the latent vectors learned. Let $U \in \mathbb{R}^{N\times k}$ store the N user latent vectors in its rows. Let $V \in \mathbb{R}^{M\times k}$ store the M item latent vectors in its rows. Let $U_{i,:}^\top$ and $V_{i,:}^\top$ denote row i of U and V, respectively.

Probabilistic Matrix Factorization (PMF) assumes the following generative process for U, V and R:

1) For each row i, draw latent vector $U_{i,:} \sim \mathcal{N}(\mathbf{0}, \sigma_u^2 I)$
2) For each row i, draw latent vector $V_{i,:} \sim \mathcal{N}(\mathbf{0}, \sigma_v^2 I)$
3) For each rating data point r_{ij}, draw $r_{ij} \sim \mathcal{N}(U_{i,:}^\top V_{i,:}, \sigma_r^2)$

where I denotes the identity matrix. This gives a factorization $R = UV^\top$, and all latent vectors are assumed to be independently drawn. By assuming such independence, PMF inherently restricts itself from leveraging information about other users or other items when modelling one r_{ij}. For sparse datasets in which user i and/or item j may have little to no ratings, PMF thus collapses. To reconcile this problem, Kernelized Probabilistic Matrix Factorization (KPMF) introduces complexity to the PMF generative process. KPMF captures covariances between items through the following generative process, which introduces covariance matrices $K_u \in \mathbb{R}^{N\times N}, K_v \in \mathbb{R}^{M\times M}$ [2]:

1) For each column i, draw latent vector $U_{:,i} \sim \mathcal{N}(\mathbf{0}, K_u)$
2) For each column i, draw latent vector $V_{i,:} \sim \mathcal{N}(\mathbf{0}, K_v)$
3) For each data point r_{ij}, draw $r_{ij} \sim \mathcal{N}(U_{i,:}^\top V_{i,:}, \sigma_r^2)$

Note that K_u and K_v are not random. We typically construct these covariance matrices a priori using side information. In addition, KPMF draws one column, not one row, at a time from a multivariate normal distribution, to capture covariances between entries in different rows.

B. Last.fm Dataset

The hetrec2011-lastfm-2k dataset contains social network, tagging, and music artist listening data for a set of 2,000 users and 1,000 artists on Last.fm [3]. Since there is no explicit rating data provided, we instead model the log(listening count), where the listening count is the number of times user i listened to some artist j. The social network data consists of 25,424 friendship pairs between the users. Users also labelled artists with 87,366 tags, of which 9,800 are unique. For experiments, we only keep a subset of the most popular tags. Note that since users can submit any string as a tag (i.e. they do not choose from a list of options), a large percentage of tags are unique. As described later, we take this side information to construct covariance matrices governing the generative process for each latent vector.

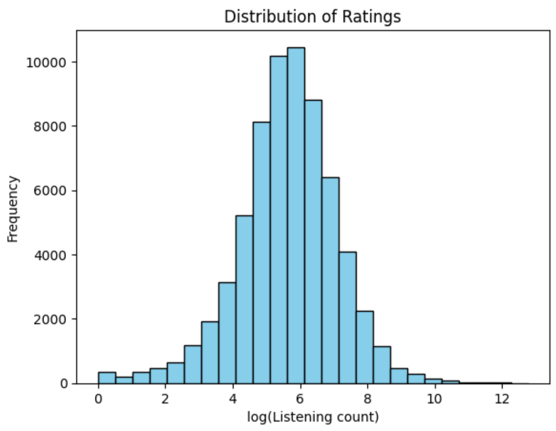

Fig. 1: Listening count is roughly log-normally distributed and highly imbalanced, so we model its logarithm and only consider the top 500 artists.

$$\mathbf{T} = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{cccc|l} \text{classical} & \text{pop} & \text{old school} & \text{2000's} & \\ 0 & 1 & 1 & 0 & \text{ABBA} \\ 1 & 0 & 0 & 0 & \text{Mozart} \\ 0 & 1 & 0 & 1 & \text{Britney Spears} \end{array}$$

Fig. 2: A tag matrix for a small subset of the dataset.

II. Methodology

A. Constructing Covariance Matrices

Due to computational limitations, we only include data for the top $M = 500$ artists, for which there are roughly 76,000 ratings. We also only retain the top 100 artist tags, though this number can be treated as a hyperparameter in future studies.

To construct the user covariance matrix $K_u \in \mathbb{R}^{N\times N}$, we leverage the social network friendship data, which we represent as an undirected graph G with adjacency matrix A. The Laplacian matrix of G is defined as $L = D - A$, where D is a diagonal matrix with entries $d_i = \sum_{j=1}^{N} A_{i,j}$.

To estimate the covariance between latent vectors for user i and user j, we apply popular graph kernel methods, which provide a way to capture the "similarity" between nodes (which correspond to users) in a graph. A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ computes the inner product of two objects in some feature space. That is, $k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)$, where ϕ is some feature map. Given a dataset of n items $\{x_n\}_n$, the kernel matrix $K \in \mathbb{R}^{n \times n}$ is defined as $K_{i,j} = k(x_i, x_j)$. It can be shown that k is a valid kernel if K is symmetric positive-semidefinite for any $\{x_n\}_n$. We can choose k such that it provides a good notion of similarity between the two objects, which are graph nodes in this case. In particular, we investigate the Commute Time (CT) graph kernel, which defines kernel matrix $K_{CT} = L^\dagger$, the pseudoinverse of the Laplacian. The CT kernel is closely related to the number of steps a random walker takes to commute between two nodes, and embeds a graph node in $N-$dimensional space [4]. We directly take $K_u = K_{CT}$.

To construct the artist covariance matrix $K_v \in \mathbb{R}^{M \times M}$, we leverage the artist tagging data. After retaining the top 100 artist tags, we embed each artist i as a vector $x^i \in \mathbb{R}^{100}$, where $x^i_j = \mathbf{1}[\text{Artist received tag j from any user}]$. We apply the Radial Basis Function (RBF) kernel, given by $k(x_i, x_j) = \exp\left(\frac{||x_i - x_j||^2}{2}\right)$, to generate the entries of K_v. It can be shown that the RBF kernel embeds vectors in an infinite-dimensional space (and returns their inner product).

B. *Gradient Descent for KPMF*

We learn the latent matrices U and V by performing a MAP estimate, which maximizes the log-posterior. Let $U_{i,:}^\top$ and $V_{i,:}^\top$ denote rows i of U and V respectively. Let $U_{:,i}$ and $V_{:,i}$ denote columns i of U and V respectively. We equivalently minimize the negative log-posterior $L = -\log p(U, V \mid \mathbf{r})$, where $\mathbf{r}$ denotes the set of all user-arist rating pairs in the training set:

$$L = -\frac{1}{2\sigma_{r^2}} \sum_{i,j \in \mathbf{r}} (r_{ij} - U_{i,:}^\top V_{j,:})^2 - \frac{1}{2} \sum_{k=1}^{K} U_{:,k}^\top K_u^{-1} U_{:,k} - \frac{1}{2} \sum_{k=1}^{K} V_{:,k}^\top K_u^{-1} V_{:,k} \tag{1}$$

We split the ratings with an 80-20 train-test split, and implement gradient descent to minimize L. The gradients are given by:

$$\frac{\partial L}{\partial U_{i,j}} = -\frac{1}{\sigma_r^2} \sum_{m=1,(i,m)\in \mathbf{r}}^{M} (r_{i,m} - U_{i,:}^\top V_{m.:}) V_{j,m} + \mathbf{e}_{(n)}^\top K_u^{-1} U_{:,j} \tag{2}$$

$$\frac{\partial L}{\partial V_{i,j}} = -\frac{1}{\sigma_r^2} \sum_{n=1,(n,i)\in \mathbf{r}}^{N} (r_{n,i} - U_{n,:}^\top V_{i,:}) U_{j,n} + \mathbf{e}_{(m)}^\top K_v^{-1} V_{:,j} \tag{3}$$

where $\mathbf{e}_{(n)}$ denotes the n-dimensional vector with a 1 in the n'th entry and 0 in the others.

We employ grid search hyperparameter tuning on σ_r^2 and k to find the combination that yields the lowest root mean squared error (RMSE) for each variation of covariance matrices used.

We implement all data processing steps, the learning algorithm, and hyperparameter tuning from scratch with the exception of the actual gradient calculation itself, for which we resort to PyTorch's autodifferentiation engine.

III. Results and Discussion

We consider four different generative models: general PMF (which assumes independence across all latent vectors and thus a diagonal covariance matrix), KPMF using just the CT graph kernel on the users' side, KPMF using just the RBF kernel on the artists' side, and KPMF using both kernels. For each configuration, we grid search for optimal hyperparameters over $\sigma_r \in \{0.25, 0.5, 1, 1.5, 2, 2.5\}$ and $K \in \{5, 10, 15, 20, 25\}$, then take the lowest MSE. We then compare the four final RMSEs.

Grid search hyperparameter tuning over $\sigma_r \in \{0.25, 0.5, 1, 1.5, 2, 2.5\}$ and $K \in \{5, 10, 15, 20, 25, 30\}$ reveals that $\sigma_r = 1, K = 20$ generally yields the lowest RMSE:

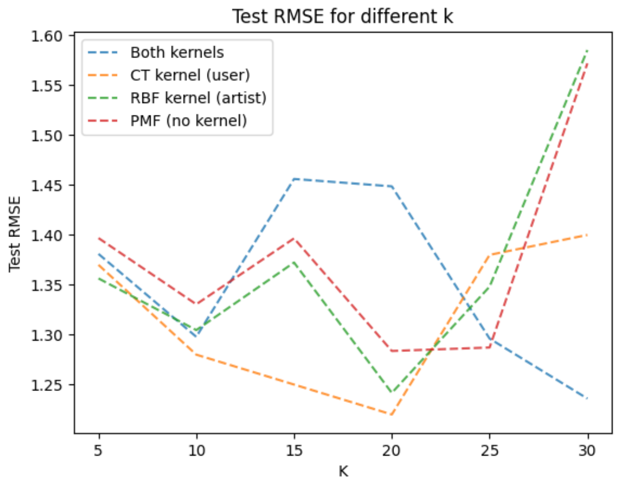

Fig. 3: Test RMSEs across different K for $\sigma_r = 1$.

Note that although the RMSE for the model leveraging both kernels appears to still be leveling out in this plot, additional experimentation reveals that it never dips below the best RMSE provided by the CT kernel model, which ultimately prevails. Even still, the improvement is small; one consideration is that Last.fm's friendship system is not widely used by its members. Interestingly, despite the two kernels being leveraged independently, combining them yields a higher RMSE than simply using the CT kernel. To a lesser extent, applying just the RBF kernel yields a slight performance improvement compared to the general PMF model.

IV. Future Research

While this study generates covariance matrices for the generative model with some level of effectiveness, much experimentation remains to be done. Previous studies have investigated a number of different graph kernels, including the diffusion graph kernel [5], which provides an interpretation on the diffusion process of a substance such as heat. We can experiment with such pre-existing kernels, or perhaps come up with our own, in hopes of finding one that better quantifies two users'/artists' similarity. Alternatively, we can embed nodes as vectors through popular methods such as node2Vec, and apply a kernel on such embeddings afterwards.

There is also room to incorporate even more complexity to the matrix factorization model. For example, constrained PMF (CPMF) introduces a latent "similarity constraint" matrix, which assigns users who give similar ratings with similar priors for their latent vectors [6].

Finally, rather than keeping the number of tags constant (100), we can instead treat this value as a hyperparameter. Intuitively, introducing too many infrequent tags may simply add noise to the embeddings.

V. References

[1] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 10, pages 285–295, 2001.

[2] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernalized Probabilistic Matrix Factorization: Exploiting Graphs and Side Information. 2012.

[3] Cantador, I., Brusilovsky, P., and Kuflik, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

[4] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[5] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *ICML*, 2001.

[6] Mnih, A. and Salakhutdinov, R. R. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pp. 1257–1264, 2008.