Cold-start Playlist Recommendation with Multitask Learning

Dawei Chen^{1,2}, Cheng Soon Ong^{2,1}, and Aditya Krishna Menon^{*1}

^{1}**Research School of Computer Science, Australian National University, Canberra, ACT, Australia**
^{2}**Machine Learning Research Group, Data61, CSIRO, Canberra, ACT, Australia**

Corresponding author:
Dawei Chen

Email address: dawei.chen@anu.edu.au

ABSTRACT

Playlist recommendation involves producing a set of songs that a user might enjoy. We investigate this problem in three cold-start scenarios: (i) *cold playlists*, where we recommend songs to form new personalised playlists for an existing user; (ii) *cold users*, where we recommend songs to form new playlists for a new user; and (iii) *cold songs*, where we recommend newly released songs to extend users' existing playlists. We propose a flexible multitask learning method to deal with all three settings. The method learns from user-curated playlists, and encourages songs in a playlist to be ranked higher than those that are not by minimising a bipartite ranking loss. Inspired by an equivalence between bipartite ranking and binary classification, we show how one can efficiently approximate an optimal solution of the multitask learning objective by minimising a classification loss. Empirical results on two real playlist datasets show the proposed approach has good performance for cold-start playlist recommendation.

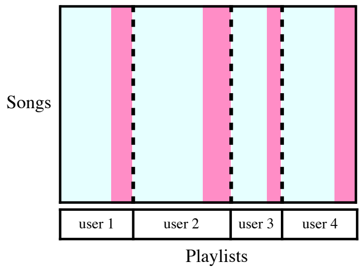
1 INTRODUCTION

Online music streaming services (e.g., Spotify, Pandora, Apple Music) are playing an increasingly important role in the digital music industry. A key ingredient of these services is the ability to automatically recommend songs to help users explore large collections of music. Such recommendation is often in the form of a *playlist*, which involves a (small) set of songs.

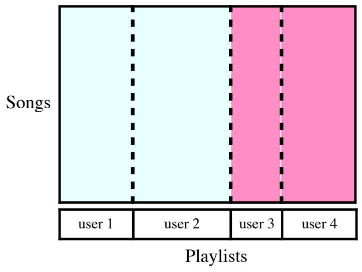
We investigate the problem of recommending a set of songs to form personalised playlists in *cold-start* scenarios, where there is no historical data for either users or songs. Conventional recommender systems for books or movies (Sarwar et al., 2001; Netflix, 2006) typically learn a score function via matrix factorisation (Koren et al., 2009), and recommend the item that achieves the highest score. This approach is not directly suited to cold-start settings due to the lack of interaction data. Further, in playlist recommendation, one has to recommend a subset of a large collection of songs instead of only one top ranked song. Enumerating all possible such subsets is intractable; additionally, it is likely that more than one playlist is satisfactory, since users generally maintain more than one playlist when using a music streaming service, which leads to challenges in standard supervised learning.

We first study the setting of recommending personalised playlists for a user by exploiting the (implicit) preference from her existing playlists. We call this setting *cold playlists*, since we do not have any contextual information about the new playlist. We find that learning from a user's existing playlists improves the accuracy of recommendation compared to suggesting popular songs from familiar artists (i.e., artists in the user's listening history). We further consider the setting of *cold users* (or new users, i.e., users with no listening history), where we recommend playlists for new users given playlists from existing users. We find it challenging to improve recommendations beyond simply ranking songs according to their popularity if we know nothing except the identifiers of new users, which is consistent with previous discoveries (McFee et al., 2012; Bonnin and Jannach, 2013, 2015). However, improvement can

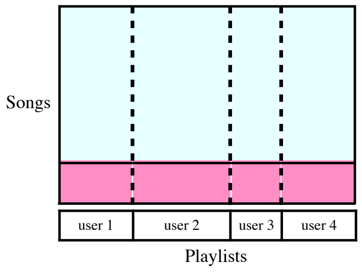
*This work has been done when the author was at the Australian National University, the author now works at Google Research.



(a) Cold Playlists



(b) Cold Users



(c) Cold Songs

Figure 1. Illustration of the three settings of cold-start playlist recommendation. In each setting, a row represents a song, and a column represents a playlist, which is a binary vector where an element denotes whether the particular song is in the corresponding playlist. Playlists are grouped by user. **Light Cyan** represents playlists or songs in the training set, and **dark Magenta** represents playlists or songs in the test set. **(a) Cold Playlists**: recommending personalised playlists for each user given the users' existing playlists; **(b) Cold Users**: recommending playlists for new users given the playlists from existing users; **(c) Cold Songs**: recommending newly released songs to extend users' existing playlists.

be achieved if we know a few simple attributes (e.g., age, gender, country) of a new user. Lastly, we investigate the setting of recommending newly released songs (i.e., *cold songs*) to extend users' existing playlists. We find that the set of songs in a playlist are particularly helpful in guiding the selection of new songs to be added to the given playlist.

Figure 1 illustrates the three cold-start settings for playlist recommendation that we study in this work. In the *cold playlists* setting, a target user (i.e., the one for whom we recommend playlists) maintains a number of playlists that can be exploited by the learning algorithm. In the *cold users* setting, however, we may only know a few simple attributes of a new user or nothing except her user identifier. The learning algorithm can only make use of playlists from existing users. Finally, in the *cold songs* setting, the learning algorithm have access to content features (e.g., artist, genre, audio data) of newly released songs as well as all playlists from existing users.

We propose a novel multitask learning method that can deal with playlist recommendation in all three cold-start settings. It optimises a bipartite ranking loss (Freund et al., 2003; Agarwal and Niyogi, 2005) that ranks the set of songs in a playlist above songs that are not in it. This results in a convex optimisation problem with an enormous number of constraints. Inspired by an equivalence between bipartite ranking and binary classification (Ertekin and Rudin, 2011), we efficiently approximate an optimal solution of the constrained objective by minimising an unconstrained classification loss. We present experiments on two real playlist datasets, and demonstrate that our multitask learning approach improves over existing strong baselines for playlist recommendation in cold-start scenarios.

2 MULTITASK LEARNING FOR PLAYLIST RECOMMENDATION

We first introduce a multitask learning objective and then show how the problem of playlist recommendation can be handled in each of the three cold-start settings. We discuss the challenge in optimising the multitask learning objective via convex constrained optimisation and show how one can efficiently approximate an optimal solution by minimising an unconstrained objective.

2.1 Multitask learning objective

Suppose we have a dataset $\mathscr{D}$ with N playlists from U users, where songs in every playlist are from a music collection with M songs. Content features (e.g., artist, genre, audio data) of all songs are available, and we use $d \in \mathbb{Z}^+$ to denote the number of features for each song. Depending on the playlist dataset, a few simple attributes of users (e.g., age, gender, country) could be provided, or nothing except the identifiers of users are known. We assume each user has at least one playlist, and each song in the collection appears in at least one playlist.

Let P_u denote the set of (indices of) playlists from user $u \in \{1,\dots,U\}$. We aim to learn a function $f(m,u,i)$ that measures the affinity between song $m \in \{1,\dots,M\}$ and playlist $i \in P_u$ from user u. Given the feature vector $\mathbf{x}_m \in \mathbb{R}^d$ of song m, suppose the affinity function $f : \mathbb{R}^d \to \mathbb{R}$ takes the form of a linear function, i.e., $f(m,u,i) = \mathbf{w}_{u,i}^\top \mathbf{x}_m$ where $\mathbf{w}_{u,i} \in \mathbb{R}^d$ are the weights of playlist i from user u.

Inspired by Ben-Elazar et al. (2017) where the weights of a playlist are decomposed into user weights and artist weights, we decompose $\mathbf{w}_{u,i}$ into three components

$$\mathbf{w}_{u,i} = \boldsymbol{\alpha}_u + \boldsymbol{\beta}_i + \boldsymbol{\mu}, \tag{1}$$

where $\boldsymbol{\alpha}_u \in \mathbb{R}^d$ are weights for user u, $\boldsymbol{\beta}_i \in \mathbb{R}^d$ are weights specific for playlist i, and $\boldsymbol{\mu} \in \mathbb{R}^d$ are the weights shared by all users (and playlists). This decomposition allows us to learn the user weights $\boldsymbol{\alpha}_u$ using all her playlists, and exploit all training playlists when learning the shared weights $\boldsymbol{\mu}$.

Let θ denote all parameters in $\left\{\{\boldsymbol{\alpha}_u\}_{u=1}^U, \{\boldsymbol{\beta}_i\}_{i=1}^N, \boldsymbol{\mu}\right\}$. The learning task is to minimise the empirical risk of affinity function f on dataset $\mathscr{D}$ over parameters θ, i.e.,

$$\min_\theta \; \Omega(\theta) + R_\theta(f, \mathscr{D}), \tag{2}$$

where $\Omega(\theta)$ is a regularisation term and $R_\theta(f, \mathscr{D})$ denotes the empirical risk of f on $\mathscr{D}$. We call the objective in problem (2) the *multitask learning objective*, since we jointly learn (parameters θ) from multiple tasks where each one involves recommending a set of songs given a user or a playlist.

We further assume that playlists from the *same* user have *similar* weights and the shared weights $\boldsymbol{\mu}$ are sparse (i.e., users only share a small portion of their weights). To impose these assumptions, we apply ℓ_1 regularisation to encourage sparse parameters in $\boldsymbol{\beta}_i$ and $\boldsymbol{\mu}$. The regularisation term in our multitask learning objective is

$$\Omega(\theta) = \lambda_1 \sum_{u=1}^U \|\boldsymbol{\alpha}_u\|_2^2 + \lambda_2 \sum_{i=1}^N \|\boldsymbol{\beta}_i\|_1 + \lambda_3 \|\boldsymbol{\mu}\|_1, \tag{3}$$

where constants $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}^+$, and the ℓ_2 regularisation term is to penalise large values in user weights. We specify the empirical risk $R_\theta(f, \mathscr{D})$ in Section 3.

2.2 Cold-start playlist recommendation

Once parameters θ have been learned, we make a recommendation by first scoring each song according to available information (e.g., an existing user or playlist), then form or extend a playlist by either taking the set of top-K scored songs or sampling a set of songs with probabilities proportional to their scores. Specifically, in the *cold playlists* setting where the target user u is known, we score song m as

$$r_m^{(a)} = (\boldsymbol{\alpha}_u + \boldsymbol{\mu})^\top \mathbf{x}_m. \tag{4}$$

Further, in the *cold users* setting where simple attributes of the new user are available, we first approximate the weights of the new user using the average weights of similar existing users (i.e., users whose attribute vectors are similar to that of the new user in terms of e.g., the cosine similarity), then we can score song m as

$$r_m^{(b)} = \left(\frac{1}{|\mathscr{U}|} \sum_{u \in \mathscr{U}} \boldsymbol{\alpha}_u + \boldsymbol{\mu} \right)^\top \mathbf{x}_m, \tag{5}$$

where $\mathscr{U}$ is the set of (e.g., 10) existing users that are most similar to the new user. On the other hand, if we know nothing about the new user except her identifier, we can simply score song m as

$$r_m^{(b)} = \boldsymbol{\mu}^\top \mathbf{x}_m, \tag{6}$$

where $\boldsymbol{\mu}$ is the shared weights that can be interpreted as a prior from a Bayesian point of view.

Lastly, in the *cold songs* setting where we are given a specific playlist i from user u, we therefore can score song m using both user weights and playlist weights, i.e.,

$$r_m^{(c)} = (\boldsymbol{\alpha}_u + \boldsymbol{\beta}_i + \boldsymbol{\mu})^\top \mathbf{x}_m. \tag{7}$$

3 EFFICIENT OPTIMISATION

We now specify the empirical risk $R_\theta(f, \mathscr{D})$ in problem (2) and develop methods to efficiently optimise the multitask learning objective.

3.1 Constrained optimisation with ranking loss

We aim to rank songs that are likely to be in a playlist above those that are unlikely to be chosen when making a recommendation. To achieve this, we optimise the multitask learning objective by minimising a bipartite ranking loss. In particular, we minimise the number of songs that are not in a training playlist but are ranked above the lowest ranked song in it. This is known as the Bottom-Push (Rudin, 2009) and the penalty of the affinity function f for playlist i from user u is

$$\Delta_f(u,i) = \frac{1}{M_-^i} \sum_{m': y_{m'}^i = 0} \left[\!\left[\min_{m: y_m^i = 1} f(m,u,i) \le f(m',u,i) \right]\!\right], \tag{8}$$

where M_-^i is the number of songs not in playlist i, binary variable y_m^i denotes whether song m appears in playlist i, and indicator function $[\![\cdot]\!]$ represents the 0/1 loss.

The empirical risk when employing the bipartite ranking loss $\Delta_f(u,i)$ is

$$R_\theta^{\text{RANK}}(f, \mathscr{D}) = \frac{1}{N} \sum_{u=1}^{U} \sum_{i \in P_u} \Delta_f(u,i). \tag{9}$$

There are two challenges to optimise the multitask learning objective with the empirical risk R_θ^{RANK}, namely, the non-differentiable 0/1 loss and the *min* function in $\Delta_f(u,i)$. To address these challenges, we first upper-bound the 0/1 loss with one of its convex surrogates, e.g., the exponential loss $[\![z \le 0]\!] \le e^{-z}$,

$$\Delta_f(u,i) \le \frac{1}{M_-^i} \sum_{m': y_{m'}^i = 0} \exp\left(f(m',u,i) - \min_{m: y_m^i = 1} f(m,u,i) \right). \tag{10}$$

One approach to deal with the *min* function in $\Delta_f(u,i)$ is introducing slack variables ξ_i to lower-bound the scores of songs in playlist i and transform problem (2) with empirical risk R_θ^{RANK} into a convex constrained optimisation problem

$$\begin{aligned} &\min_\theta \; \Omega(\theta) + \frac{1}{N} \sum_{u=1}^{U} \sum_{i \in P_u} \frac{1}{M_-^i} \sum_{m': y_{m'}^i = 0} \exp\left(f(m',u,i) - \xi_i \right) \\ &s.t.\; \xi_i \le f(m,u,i), \\ &\quad u \in \{1,\dots,U\},\; i \in P_u,\; m \in \{1,\dots,M\} \cap \{m : y_m^i = 1\}. \end{aligned} \tag{11}$$

Note that the number of constraints in problem (11) is $\sum_{u=1}^{U} \sum_{i \in P_u} \sum_{m=1}^{M} [\![y_m^i = 1]\!]$, i.e., the accumulated playcount of all songs, which is of order $O(\bar{L}N)$ asymptotically, where $\bar{L}$ is the average number of songs in playlists (typically less than 100). However, the total number of playlists N can be enormous in production systems (e.g., Spotify hosts more than 2 billion playlists (Spotify, 2018)) which imposes a significant challenge in optimisation. This issue could be alleviated by applying the cutting-plane method (Avriel, 2003) or the sub-gradient method. Unfortunately, we find both methods converge extremely slowly for this problem in practice. In particular, the cutting plane method is required to solve a constrained optimisation problem with at least N constraints in each iteration, which remains challenging.

3.2 Unconstrained optimisation with classification loss

Alternatively, we can approximate the *min* function in $\Delta_f(u,i)$ using the well known Log-sum-exp function (Boyd and Vandenberghe, 2004), i.e., $\min_j z_j = -\max_j(-z_j) = -\lim_{p \to +\infty} \frac{1}{p} \log \sum_j \exp(-pz_j)$, which allows us to approximate the empirical risk R_θ^{RANK} (with the exponential surrogate) by $\widetilde{R}_\theta^{\text{RANK}}$ defined as

$$\widetilde{R}_\theta^{\text{RANK}}(f, \mathscr{D}) = \frac{1}{N} \sum_{u=1}^{U} \sum_{i \in P_u} \frac{1}{M_-^i} \left[\delta_f(u,i) \right]^{\frac{1}{p}}, \tag{12}$$

where hyper-parameter $p \in \mathbb{R}^+$ and

$$\delta_f(u,i) = \sum_{m:y_m^i=1} \left[\sum_{m':y_{m'}^i=0} \exp(-(f(m,u,i) - f(m',u,i))) \right]^p . \tag{13}$$

We further observe that $\delta_f(u,i)$ can be transformed into the objective of the standard P-Norm Push (Rudin, 2009) by simply swapping the positives $\{m : y_m^i = 1\}$ and negatives $\{m' : y_{m'}^i = 0\}$. Inspired by the connections between bipartite ranking and binary classification (Menon and Williamson, 2016), we swap the positives and negatives in the objective of the P-Classification (Ertekin and Rudin, 2011) while taking care of signs. This results in an empirical risk with a classification loss:

$$R_\theta^{\text{MTC}}(f,\mathscr{D}) = \frac{1}{N} \sum_{u=1}^{U} \sum_{i \in P_u} \left(\frac{1}{pM_+^i} \sum_{m:y_m^i=1} \exp(-pf(m,u,i)) + \frac{1}{M_-^i} \sum_{m':y_{m'}^i=0} \exp(f(m',u,i)) \right), \tag{14}$$

where M_+^i is the number of songs in playlist i.

Lemma 1. *Let* $\theta^* \in \operatorname{argmin}_\theta R_\theta^{\text{MTC}}$ *(assuming minimisers exist), then* $\theta^* \in \operatorname{argmin}_\theta \widetilde{R}_\theta^{\text{RANK}}$.

Proof. See Appendix A for a complete proof. Alternatively, we can use the proof of the equivalence between the P-Norm Push and the P-Classification (Ertekin and Rudin, 2011) if we swap the positives and negatives in R_θ^{MTC} and $\widetilde{R}_\theta^{\text{RANK}}$. □

By Lemma 1, we can optimise the parameters of the multitask learning objective by solving a (convex) unconstrained optimisation problem:[1]

$$\min_\theta \; \Omega(\theta) + R_\theta^{\text{MTC}}(f,\mathscr{D}). \tag{15}$$

Problem (15) can be efficiently optimised using the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) algorithm (Andrew and Gao, 2007), an L-BFGS variant that can address ℓ_1 regularisation effectively. We refer to the approach that solves problem (15) as *Multitask Classification* (MTC). As a remark, optimal solutions of problem (15) are not necessarily the optimal solutions of $\min_\theta \; \Omega(\theta) + \widetilde{R}_\theta^{\text{RANK}}$ due to regularisation. However, when parameters θ are small (which is generally the case when using regularisation), optimal solutions of the two objectives can nonetheless approximate each other well.

4 RELATED WORK

We summarise recent work most related to playlist recommendation and music recommendation in cold-start scenarios, as well as work on the connections between bipartite ranking and binary classification.

4.1 Playlist recommendation

There is a rich set of recent literature on playlist recommendation, which can be summarised into two typical settings: playlist generation and next song recommendation. Playlist generation is to produce a complete playlist given some seed. For example, the AutoDJ system (Platt et al., 2002) generates playlists given one or more seed songs; Groove Radio can produce a personalised playlist for the specified user given a seed artist (Ben-Elazar et al., 2017); or a seed location in hidden space (where all songs are embedded) can be specified in order to generate a complete playlist (Chen et al., 2012). There are also works that focus on evaluating the learned playlist model, without concretely generating playlists (McFee and Lanckriet, 2011, 2012). See this survey (Bonnin and Jannach, 2015) for more details.

Next song recommendation predicts the next song a user might play after observing some context. For example, the most recent sequence of songs with which a user has interacted was used to infer the contextual information, which was then employed to rank the next possible song via a topic-based sequential model learned from users' existing playlists (Hariri et al., 2012). Context can also be the artists in a user's listening history, which has been employed to score the next song together with the frequency of artist collocations as well as song popularity (McFee et al., 2012; Bonnin and Jannach, 2013). It is straightforward to produce a complete playlist using next song recommendation techniques, i.e., by picking the next song sequentially (Bonnin and Jannach, 2013; Ben-Elazar et al., 2017).

[1] We choose not to directly optimise the empirical risk $\widetilde{R}_\theta^{\text{RANK}}$, which involves the objective of P-Norm Push, since classification loss can be optimised more efficiently in this scenario (Ertekin and Rudin, 2011).

4.2 Cold-start recommendation

In the collaborative filtering literature, the cold-start setting has primarily been addressed through suitable regularisation of matrix factorisation parameters based on exogenous user- or item-features (Ma et al., 2008; Agarwal and Chen, 2009; Cao et al., 2010). Regularisation techniques for deep neural networks (e.g., dropout) have also been shown to help cold-start recommendation (Volkovs et al., 2017). The novel idea of jointly factorising a document-term matrix and a document-user matrix (Saveski and Mantrach, 2014) achieved promising cold-start document recommendations, especially with regularisation that encourages local smoothness in the learned embeddings.

Content-based approaches can handle the recommendation of new songs, typically by making use of content features of songs extracted either automatically (Seyerlehner et al., 2010; Eghbal-Zadeh et al., 2015) or manually by musical experts (John, 2006). Further, content features can also be combined with other approaches, such as those based on collaborative filtering (Yoshii et al., 2006; Donaldson, 2007; Shao et al., 2009), which is known as the hybrid recommendation approach (Burke, 2002; Aggarwal, 2016). Another popular approach for cold-start recommendation involves explicitly mapping user- or item-content features to latent embeddings (Gantner et al., 2010). This approach can be adopted to recommend new songs, e.g., by learning a convolutional neural network to map audio features of new songs to the corresponding latent embeddings (Oord et al., 2013), which were then used to score songs together with the latent embeddings of playlists. The problem of recommending music for new users can also be tackled using a similar approach, e.g., by learning a mapping from user attributes to user embeddings.

A slightly different approach to deal with music recommendation for new users is learning hierarchical representations for genre, sub-genre and artist. By adopting an additive form with user and artist weights, it can fall back to using only artist weights when recommending music to new users; if the artist weights are not available (e.g., a new artist), this approach further falls back to using the weights of sub-genre or genre (Ben-Elazar et al., 2017). However, the requirement of seed information (e.g., artist, genre or a seed song) restricts its direct applicability to the *cold playlists* and *cold users* settings. Further, encoding song usage information as features makes it unsuitable for recommending new songs directly.

4.3 Connections between bipartite ranking and binary classification

It has been well known that bipartite ranking and binary classification are closely related (Ertekin and Rudin, 2011; Menon and Williamson, 2016). In particular, Ertekin and Rudin (2011) showed that the objective of the P-Norm Push and that of the P-Classification share the same minimiser(s). Further, the P-Norm Push is an approximation of the Infinite-Push (Agarwal, 2011), or equivalently, the Top-Push (Li et al., 2014), which focuses on the highest ranked negative example instead of the lowest ranked positive example in the Bottom-Push adopted in this work.

Compare to the Bayesian Personalised Ranking (BPR) approach (Rendle et al., 2009; McFee et al., 2012) that requires all positive items to be ranked higher than those unobserved ones, the adopted approach only penalises unobserved items that are ranked higher than the lowest ranked positive item, which can be optimised more efficiently when only the top ranked items are of interest (Rudin, 2009; Li et al., 2014).

5 EXPERIMENTS

We present empirical evaluations for cold-start playlist recommendation on two real playlist datasets, and compare the proposed multitask learning method with a number of well known baseline approaches.

5.1 Dataset

We evaluate on two publicly available playlist datasets: the 30Music (Turrin et al., 2015) and the AotM-2011 (McFee and Lanckriet, 2012) dataset. The Million Song Dataset (MSD) (Bertin-Mahieux et al., 2011) serves as an underlying dataset where songs in all playlists are intersected (i.e., filtering out songs not in the MSD); additionally, song and artist information in the MSD are used to compute song features.

30Music Dataset is a collection of listening events and user-generated playlists retrieved from Last.fm.[2] We first intersect the playlists data with songs in the MSD, then filter out playlists with less than 5 songs. This results in about 17K playlists over 45K songs from 8K users.

AotM-2011 Dataset is a collection of playlists shared by Art of the Mix[3] users during the period from 1998 to 2011. Songs in playlists have been matched to those in the MSD. It contains roughly 84K

[2]https://www.last.fm
[3]http://www.artofthemix.org

Table 1. Statistics of music playlist datasets

	30Music	AotM-2011
Playlists	17,457	84,710
Users	8,070	14,182
Avg. Playlists per User	2.2	6.0
Songs	45,468	114,428
Avg. Songs per Playlist	16.3	10.1
Artists	9,981	15,698
Avg. Artists per Playlist	11.5	9.0
Avg. Songs per Artist	4.6	7.1

playlists over 114K songs from 14K users after filtering out playlists with less than 5 songs. Table 1 summarises the two playlist datasets used in this work.

The histograms of the number of playlists per user as well as song popularity (i.e., the accumulated playcount of the song in the training set) of the two datasets are shown in Figure 2 and Figure 3, respectively. We can see that both the number of playlists per user (Figure 2) and song popularity (Figure 3) follow a long-tailed distribution, which imposes further challenge to the learning task as the amount of data is very limited for users (or songs) at the tail.

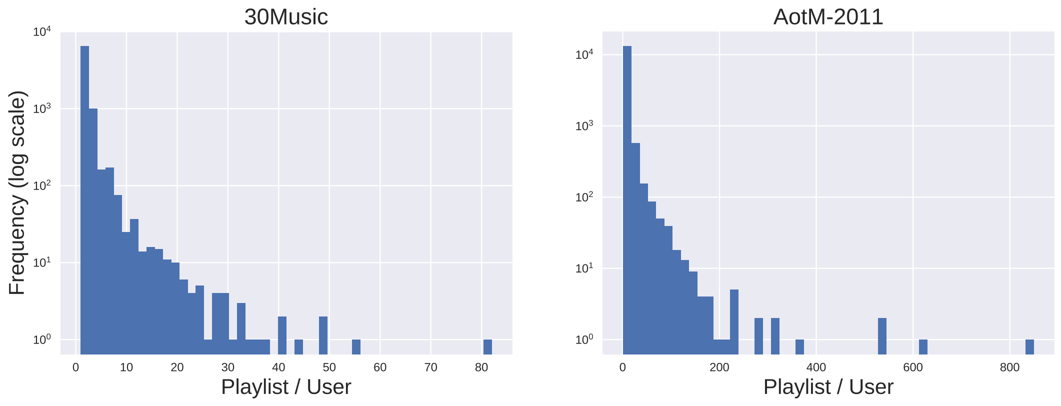


Figure 2. Histogram of the number of playlists per user

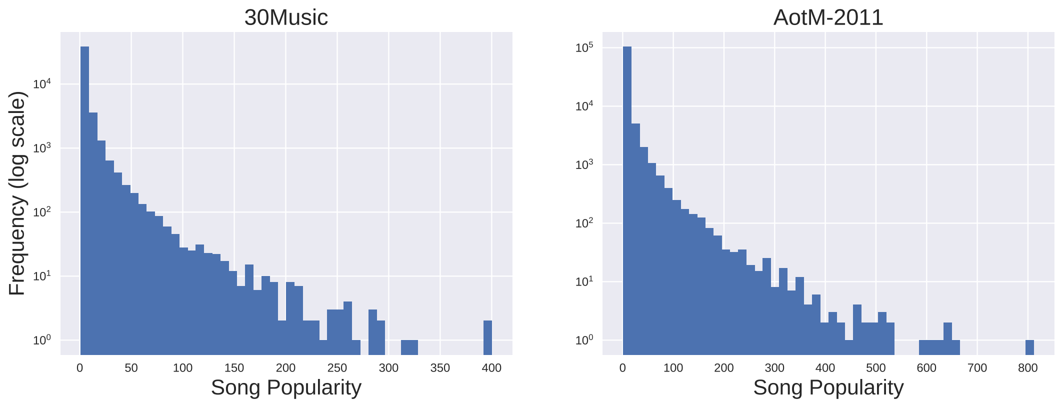


Figure 3. Histogram of song popularity

Table 2. Statistics of datasets in three cold-start settings.

		Training Set			Test Set		
		Users	Playlists	Songs	Users	Playlists	Songs
Cold Playlists	30Music	8,070	15,262	45,468	1,644	2,195	45,468
	AotM-2011	14,182	75,477	114,428	2,722	9,233	114,428
Cold Users	30Music	5,649	14,067	45,468	2,420	3,390	45,468
	AotM-2011	9,928	76,450	114,428	4,254	8,260	114,428
Cold Songs	30Music	8,034	17,342	40,468	8,034	8,215	5,000
	AotM-2011	14,177	84,646	104,428	14,177	19,504	10,000

5.2 Features

Song metadata, audio data, genre and artist information, as well as song popularity and artist popularity (i.e., the accumulated playcount of all songs from the artist in the training set) are encoded as features. The metadata of songs (e.g., duration, year of release) and pre-computed audio features (e.g., loudness, mode, tempo) are from the MSD. We use genre data from the Top-MAGD genre dataset (Schindler et al., 2012) and tagtraum genre annotations for the MSD (Schreiber, 2015) via one-hot encoding. If the genre of a song is unknown, we apply mean imputation using genre counts of songs in the training set.

To encode artist information as features, we create a sequence of artist identifiers for each playlist in the training set, and train a word2vec[4] model that learns embeddings of artists. We assume no popularity information is available for newly released songs, and therefore song popularity is not a feature in the *cold songs* setting. Finally, we add a constant feature (with value 1.0) for each song to account for bias.

5.3 Experimental setup

We first split the two playlist datasets into training and test sets, then evaluate the performance of the proposed method (on the test set), and compare it against several baseline approaches in each of the three cold-start settings.

Dataset split In the *cold playlists* setting, we hold a portion of the playlists from about 20% of users in both datasets for testing, and all other playlists are used for training. The test set is formed by sampling playlists where each song has been included in at least four other playlists among the whole dataset. We also make sure each song in the test set appears in the training set, and all users in the test set have a few playlists in the training set.

In the *cold users* setting, we sample 30% of users and hold all of their playlists for testing in both datasets. Similarly, we require songs in the test set to appear in the training set, and a user will thus not be used for testing if holding all of her playlists breaks this requirement.

Lastly, in the *cold songs* setting, we hold 5K of the latest released songs in the 30Music dataset, and 10K of the latest released songs in the AotM-2011 dataset where more songs are available. We remove playlists where all songs have been held for testing.

Table 2 summaries the dataset splits in three cold-start settings.

Baselines We compare the performance of our proposed method (i.e., MTC) with the following baseline approaches in each of the three cold-start settings:

- The *Popularity Ranking* (PopRank) method scores a song using only its popularity in the training set. In the *cold songs* setting where song popularity is not available, a song is scored by the popularity of the corresponding artist.

- The *Same Artists - Greatest Hits* (SAGH) (McFee et al., 2012) method scores a song by its popularity if the artist of the song appears in the given user's playlists (in the training set); otherwise the song is scored zero. In the *cold songs* setting, this method only considers songs from artists that appear in the given playlist, and scores a song using the popularity of the corresponding artist.

[4]https://github.com/dav/word2vec

- The *Collocated Artists - Greatest Hits* (CAGH) (Bonnin and Jannach, 2013) method is a variant of SAGH. It scores a song using its popularity, but weighted by the frequency of the collocation between the artist of the song and artists that appear in the given user's playlists (in the training set). In the *cold users* setting, we use the 10 most popular artists instead of artists in the user's listening history (which is not available), and the *cold songs* setting is addressed in the same way as in SAGH (i.e., considering only those artists that appear in the given playlist).

We also compare with a variant of Matrix Factorisation (MF) in each setting, which first learns the latent factors of songs, playlists or users, then scores each song by the dot product of the corresponding latent factors. Recommendations are made as per the proposed method. To be specific,

- In the *cold playlists* setting, we factorise the song-user playcount matrix using the weighted matrix factorisation (WMF) algorithm (Hu et al., 2008), which learns the latent factors of songs and users. We call this method WMF.

- In the *cold users* setting, we first learn the latent factors of songs and users by factorising the song-user playcount matrix using WMF, then approximate the latent factors of a new user by the average latent factors of the k (e.g., 100) nearest neighbours (in terms of the cosine similarity between user attribute vectors) in the training set. We call this method WMF+kNN.[5]

- In the *cold songs* setting, we factorise the song-playlist matrix to learn the latent factors of songs and playlists, which are further employed to train a fully-connected neural network that maps the content features of a song to its corresponding latent factors (Gantner et al., 2010; Oord et al., 2013). We can then obtain the latent factors of a new song as long as its content features are available. We call this method MF+MLP.

Evaluation We first evaluate all approaches using two accuracy metrics that have been adopted in playlist recommendation tasks: HitRate@K and Area under the ROC curve (AUC).

HitRate@K (Hariri et al., 2012), which is also known as Recall@K, is the number of correctly recommended songs amongst the top-K recommendations over the number of songs in the observed playlist. It has been widely employed to evaluate playlist generation and next song recommendation methods (Hariri et al., 2012; Bonnin and Jannach, 2013, 2015; Jannach et al., 2015).

Area under the ROC curve (AUC) (Manning et al., 2008), which is the probability that a positive instance is ranked higher than a negative instance (on average). AUC has been primarily used to measure the performance of classifiers. It has been applied to evaluate playlist generation methods when the task has been cast as a sequence of classification problems (Ben-Elazar et al., 2017).

It is believed that useful recommendations need to include previously unknown items, and this ability can be measured by *Novelty* (Herlocker et al., 2004; Zhang et al., 2012; Schedl et al., 2017),

$$\text{Novelty@K} = \frac{1}{U} \sum_{u=1}^{U} \frac{1}{|P_u^{\text{test}}|} \sum_{i \in P_u^{\text{test}}} \sum_{m \in S_K^i} \frac{-\log_2 pop_m}{K}, \tag{16}$$

where P_u^{test} is the (indices of) test playlists from user u, S_K^i is the set of top-K recommendations for test playlist i and pop_m is the popularity of song m. Intuitively, the more popular a song is, the more likely a user is to be familiar with it, and therefore the less likely to be novel.

We also adopt another beyond-accuracy metric called *Spread* (Kluver and Konstan, 2014), which measures the ability of a recommender system to spread its attention across all possible items. It is defined as the entropy of the distribution of all songs,

$$\text{Spread} = -\sum_{m=1}^{M} \mathbb{P}(m) \log \mathbb{P}(m), \tag{17}$$

where $\mathbb{P}(m)$ denotes the probability of song m being recommended, which is computed from the scores of all possible songs using the *softmax* function in this work.

[5]The WMF+kNN method does not apply to the AotM-2011 dataset in the cold users setting, since such user attributes (e.g., age, gender and country) are not available in the dataset.

Table 3. AUC for playlist recommendation in three cold-start settings. *Higher* values indicate better performance.

Cold Playlists			Cold Users			Cold Songs		
Method	30Music	AotM-2011	Method	30Music	AotM-2011	Method	30Music	AotM-2011
PopRank	94.0	93.8	PopRank	88.3	**91.8**	PopRank	70.9	76.5
CAGH	94.8	94.2	CAGH	86.3	88.1	CAGH	68.0	77.4
SAGH	64.5	79.8	SAGH	54.5	53.7	SAGH	51.5	53.6
WMF	79.5	85.4	WMF+kNN	84.9	N/A	MF+MLP	81.4	80.8
MTC	**95.9**	**95.4**	MTC	**88.8**	**91.8**	MTC	**86.6**	**84.3**

Novelty and Spread are two of the beyond-accuracy metrics that are specifically tailored to recommender systems. Unlike the AUC and Hit Rate, where higher values indicate better performance, here *moderate* values are usually preferable for these two beyond-accuracy metrics (Kluver and Konstan, 2014; Schedl et al., 2017).

5.4 Results

We analyse the empirical results of the proposed method (i.e., MTC) as well as many baselines in terms of both accuracy metrics (i.e., HitRate and AUC) and beyond accuracy metrics (i.e., Novelty and Spread).

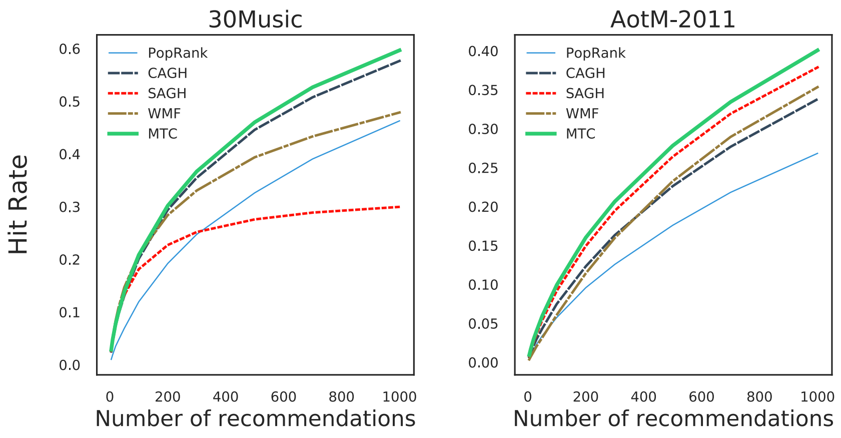
Accuracy Table 3 shows the performance of all methods in terms of AUC. We can see that PopRank achieves good performance in all three cold-start settings. This is in line with results reported in (Bonnin and Jannach, 2013, 2015). Artist information, particularly the frequency of artist collocations that is exploited in CAGH, improves recommendation in the cold playlists and cold songs settings. Further, PopRank is one of the best performing methods in the cold users setting, which is consistent with previous discoveries (McFee et al., 2012; Bonnin and Jannach, 2013, 2015). The reason is believed to be the long-tailed distribution of songs in playlists (Cremonesi et al., 2010; Bonnin and Jannach, 2013). The MF variant does not perform well in the cold playlists setting, but it performs reasonably well in the cold users setting when attributes of new users are available (i.e., in the 30Music dataset), and it works particularly well in the cold songs setting where both song metadata and audio features of new songs are provided.

Lastly, MTC is the best performing method in all three cold-start settings on both datasets. Interestingly, it is the tied best on the AotM-2011 dataset in the cold users setting (recall that this dataset does not provide user attributes such as age, gender and country), and it achieves the same performance as PopRank in the cold users setting on the AotM-2011 dataset, which suggests that MTC might degenerate to simply ranking songs according to the popularity when making recommendations for new users; however, when simple attributes of new users are available, it can improve the recommendations by exploiting information learned from existing users.

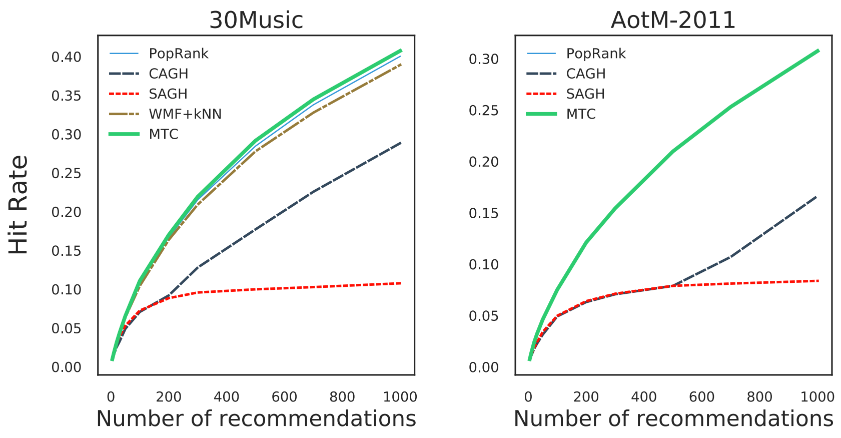
Figure 4 shows the Hit Rate of all methods in three cold-start settings when the number of recommended songs K varies from 5 to 1000. As expected, the performance of all methods improves when the number of recommendations increases. We can see from Figure 4a that SAGH and CAGH perform better than PopRank (except for SAGH on the 30Music dataset when K is larger than 300) in the cold playlists setting, which confirms that artist information is helpful in retrieving songs in ground truth playlists (i.e., improving recall). It is interesting to observe that the performance of WMF is always between SAGH and CAGH on both datasets, although the performance of both SAGH and CAGH vary significantly across datasets. This might suggest that this variant of matrix factorisation is more robust than approaches based on ranking according to song popularity and artist information.

It is challenging to improve upon simply ranking by song popularity in the cold users setting, as shown in Figure 4b, which is in line with previous discoveries (McFee et al., 2012; Bonnin and Jannach, 2013, 2015). In contrast, learning-based approaches (i.e., MTC and MF+MLP) always perform better than other baselines that use only artist information in the cold songs setting (Figure 4c). PopRank works surprisingly well; it even outperforms CAGH which exploits artist collocations on the 30Music dataset. The fact that CAGH always performs better than SAGH confirms that artist collocation is helpful for music recommendation.

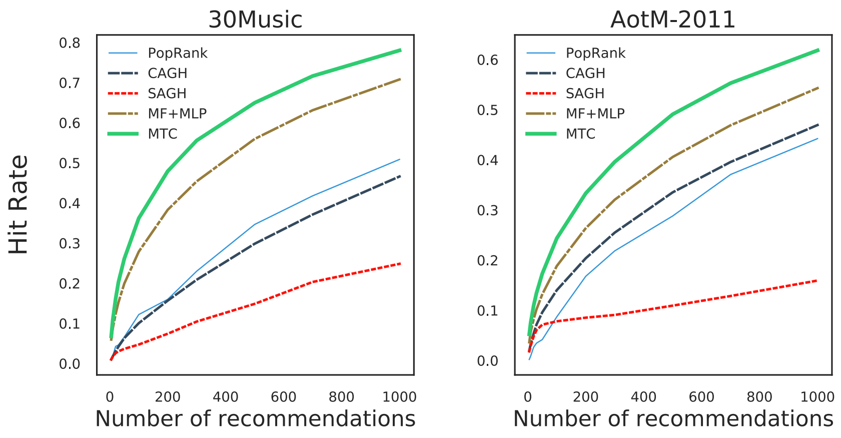
In summary, MTC outperforms all other methods by a big margin on both datasets in the cold songs setting (Figure 4c). It performs as well as PopRank in the cold users setting; however, MTC can improve



(a) Cold Playlists



(b) Cold Users



(c) Cold Songs

Figure 4. Hit Rate of playlist recommendation in three cold-start settings. *Higher* values indicate better performance.

the recommendations when attributes of new users are available (Figure 4b). We also observe that MTC outperforms other baselines in the cold playlists setting (Figure 4a), although the margin is not as big as that in the cold songs setting. This demonstrates the effectiveness of the proposed approach for cold-start playlist recommendation.

Table 4. Spread for playlist recommendation in three cold-start settings. *Moderate* values are preferable.

Cold Playlists			Cold Users			Cold Songs		
Method	30Music	AotM-2011	Method	30Music	AotM-2011	Method	30Music	AotM-2011
PopRank	9.8	10.5	PopRank	9.8	10.5	PopRank	7.4	7.8
CAGH	5.8	2.3	CAGH	4.2	5.3	CAGH	4.3	4.6
SAGH	10.3	10.4	SAGH	10.0	10.7	SAGH	6.5	5.9
WMF	10.7	11.6	WMF+kNN	10.7	N/A	MF+MLP	8.5	9.2
MTC	9.4	10.4	MTC	9.9	11.4	MTC	7.9	8.3

Beyond accuracy Table 4 shows the performance of all recommendation approaches in terms of *Spread*. In the cold songs setting, CAGH and SAGH focus on songs from artists in users' listening history (and similar artists), which explains the relative low *Spread*. However, in the cold playlists and cold users settings, SAGH improves its attention spreading due to the set of songs it focuses on is significantly bigger (i.e., songs from all artists in users' previous playlists and songs from the 10 most popular artists, respectively). Surprisingly, CAGH remains focusing on a relatively small set of songs in both settings. Lastly, in all three cold-start settings, the MF variants have the highest *Spread*, while both PopRank and MTC have (similar) moderate *Spread*, which is considered better.

Figure 5 shows the *Novelty* of all methods in three cold-start settings. The values of *Novelty* of all methods raise as the number of recommendations increases. We can see from Figure 5a that PopRank has the lowest *Novelty* in the cold playlists setting, which is not surprising given the definition (Equation 16). Both SAGH and CAGH start with low *Novelty* and grow as the number of recommended songs increases, but the *Novelty* of CAGH saturates much earlier than that of SAGH. The reason could be that, when the number of recommendations is larger than the total number of songs from artists in a user's existing playlists, SAGH will simply recommend songs randomly (which are likely to be novel) while CAGH will recommend songs from artists that are similar to those in the user's existing playlists (which could be comparably less novel). Further, MTC achieves lower *Novelty* than WMF and CAGH, which indicates that MTC tends to recommend popular songs to form new playlists.

It is interesting to observe that MTC and PopRank perform identically in the cold users setting, as shown in Figure 5b. SAGH has the largest *Novelty* on both datasets, likely for similar reasons to those in the cold playlists setting. CAGH and WMF+kNN have moderate *Novelty*, which are considered to be better. The performance of different methods (in terms of *Novelty*) in the cold songs setting (Figure 5c) are similar to those in the cold playlists setting (Figure 5b); however, there are two differences: (i) The *Novelty* of SAGH saturates after the number of recommendations reaches a certain value (roughly 60), the reason could be that, on average, the total number of songs from the set of artists in a playlist is about 60 (Table 1); (ii) MTC achieves higher *Novelty* than WMF and CAGH, which might suggest that MTC tends to recommend new songs that will be (comparably) less popular.

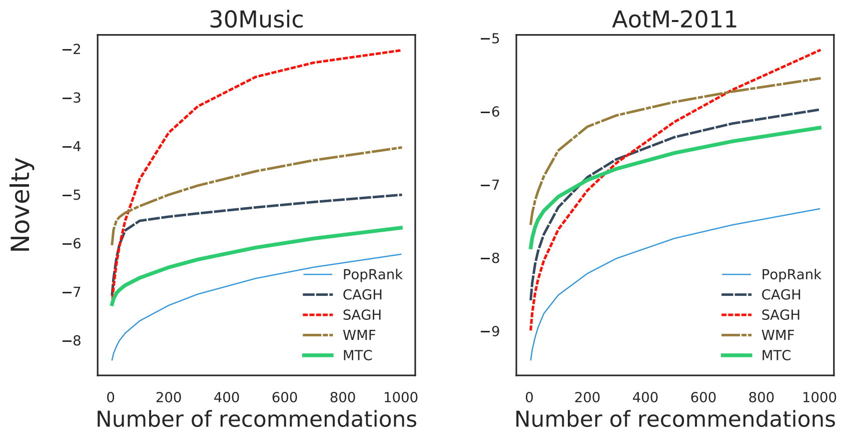
To conclude, MTC and CAGH have moderate *Novelty* in both the cold playlists and cold songs settings, and therefore perform better than other approaches. On the other hand, in the cold users setting, CAGH and the MF variant are preferred. Lastly, in both the cold playlists and the cold songs settings, the MF variants also achieve decent recommendations (in terms of *Novelty*).

6 DISCUSSION

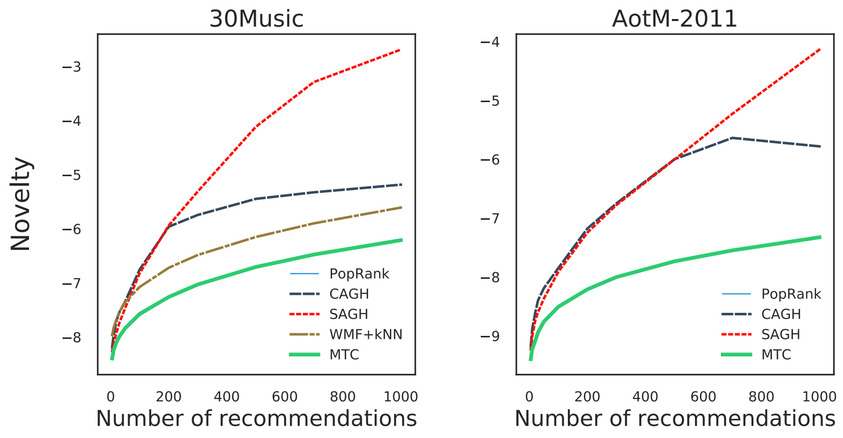
We discuss the relationship between multitask learning, bipartite ranking and binary classification (from the perspective of loss function). In addition, we also remark several design choices adopted in this work and compare our problem setup with a closely related setting considered in a recent RecSys challenge.

6.1 Multitask learning, bipartite ranking and binary classification

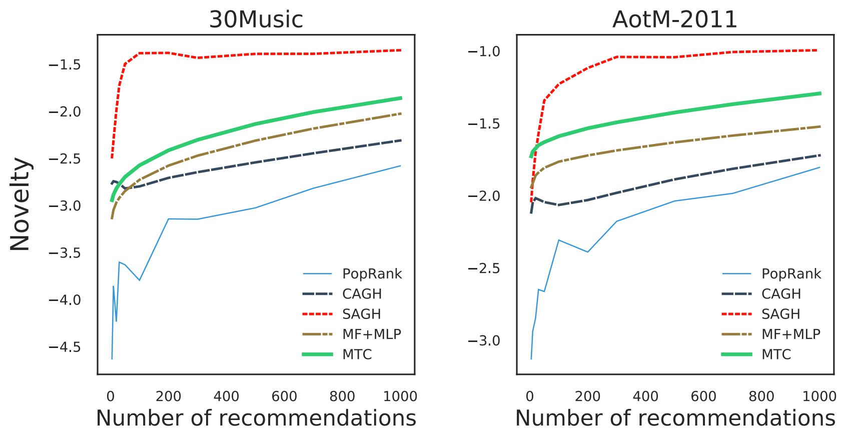
Multitask learning is a method that learns more than one tasks in parallel by using a shared representation to achieve inductive transfer between tasks, it could improve generalisation accuracy of a particular task by leveraging additional signals from related tasks (Caruana, 1993, 1997). Sharing representation among multiple tasks, which is the central idea of multitask learning, allows us to jointly learn the parameters of users and playlists, as well as the shared parameters from multiple recommendation tasks, which further enables us to deal with the problem of recommending a set of songs in three cold-start settings.



(a) Cold Playlists



(b) Cold Users



(c) Cold Songs

Figure 5. Novelty of playlist recommendation in three cold-start settings. *Moderate* values are prefereable.

The bipartite ranking loss adopted in this work (i.e., the Bottom-Push) guides the learning process such that the learned parameters will (generally) reflect our intention to rank the set of songs in a playlist higher than those that are not. Ideally, we can directly optimise this loss function using training data; unfortunately, this is infeasible due to the enormous number of constraints in the involved optimisation problem, we therefore resort to minimise an approximation of the constrained objective. It turns out that the approximation[6] transforms the Bottom-Push to the P-Norm Push. Although one can optimise the objective of the P-Norm Push using standard techniques (e.g., gradient descent), we find that more efficient optimisation can be achieved if we make use of the equivalence between bipartite ranking and binary classification, which results in an unconstrained objective with a classification loss.

6.2 Cold-start playlist recommendation versus playlist continuation

In the cold playlists setting, we recommend more than one playlists for a given user; however, all these recommendations are identical as the ranking of songs for a specific user is the same (Equation 4). This is due to the fact that no other contextual information for a recommendation is available except the user identity. Similarly, in the cold users setting, the same set of songs will always be suggested no matter how many times the recommendation have been made because of the lack of contextual information in each recommendation. A more plausible and perhaps more realistic setting is to provide one or more *seed* songs for each task,[7] and the recommended playlist should be cohesive with the given seed. This setup is known as *playlist continuation* (Schedl et al., 2017), which has been explored in a recent RecSys challenge[8]. One may notice that the setup of playlist continuation is similar to the cold songs setting, except that the set of songs to be added to a playlist are not necessarily newly released songs.

6.3 Information of songs, playlists and users

In this work, we assume that content features of songs (e.g., metadata, audio data) are provided, even for newly released songs. On the other hand, no user (or playlist) feature is available. We may have a few simple attributes of users (e.g., age, gender, country) or only user identifiers are known. In practice, users might reveal their preferences in profiles, and playlist metadata (e.g., title, description, created time) might also be available, which could be exploited by the learning algorithm.

Further, we treat a playlist as a set of songs by discarding the sequential order. It turns out that the sequential order of songs in a playlist has not been well understood (Schedl et al., 2017), some work suggest that the order of songs and song-to-song transitions are important for the quality of the recommended playlist (McFee and Lanckriet, 2012; Kamehkhosh et al., 2018), while other work discover that the order of songs seems to be negligible, but the ensemble (i.e., set) of songs in a playlist do matter (Tintarev et al., 2017; Vall et al., 2017).

As a remark, in the cold users setting, we approximate the weights (or latent factors) of a new user using the average weights (or latent factors) of similar users in the training set in MTC (or WMF+kNN). One could also use a weighted average (e.g., weighted by the normalised cosine similarity between user attribute vectors) of those similar users' weights (or latent factors), however, we did not find any significant difference in performance compared to the arithmetic mean in the experiments.

7 CONCLUSION AND FUTURE WORK

We study the problem of recommending a set of songs to form playlists in three cold-start settings: cold playlists, cold users and cold songs. We propose a multitask learning method that learns user- and playlist-specific weights as well as shared weights from user-curated playlists, which allows us to form new personalised playlists for an existing user, and to produce playlists for a new user, or to extend users' playlists with newly released songs. We optimise the parameters (i.e., weights) by minimising a bipartite ranking loss that ranks the set of songs in a playlist above songs that are not in it. An equivalence between bipartite ranking and binary classification further enables efficient approximation of optimal parameters. Empirical evaluations on two real playlist datasets demonstrate the effectiveness of the proposed method for cold-start playlist recommendation.

For future work, we would like to explore auxiliary data sources (e.g., music information shared on social media) and additional features of songs and users (e.g., lyrics, user profiles) to make better recommendations. Further, non-linear models such as deep neural networks have been shown to work extremely well in a wide range of tasks, and the proposed linear model with sparse parameters could be more compact if a non-linear model were adopted.

[6]The approximation also transforms the constrained optimisation problem into an unconstrained optimisation problem.
[7]The seed information can also be an artist or a genre, as considered in Ben-Elazar et al. (2017).
[8]http://www.recsyschallenge.com/2018/

REFERENCES

Agarwal, D. and Chen, B.-C. (2009). Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28.

Agarwal, S. (2011). The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 839–850.

Agarwal, S. and Niyogi, P. (2005). Stability and generalization of bipartite ranking algorithms. In *International Conference on Computational Learning Theory*, pages 32–47.

Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer.

Andrew, G. and Gao, J. (2007). Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40.

Avriel, M. (2003). *Nonlinear programming: analysis and methods*. Courier Corporation.

Ben-Elazar, S., Lavee, G., Koenigstein, N., Barkan, O., Berezin, H., Paquet, U., and Zaccai, T. (2017). Groove radio: A bayesian hierarchical model for personalized playlist generation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pages 445–453.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 591–596.

Bonnin, G. and Jannach, D. (2013). Evaluating the quality of playlists based on hand-crafted samples. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pages 263–268.

Bonnin, G. and Jannach, D. (2015). Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12:331–370.

Cao, B., Liu, N. N., and Yang, Q. (2010). Transfer learning for collective link prediction in multiple heterogenous domains. In *Proceedings of the 27th international conference on machine learning*, pages 159–166.

Caruana, R. (1993). Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the 10th International Conference on Machine Learning*, pages 41–48.

Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

Chen, S., Moore, J. L., Turnbull, D., and Joachims, T. (2012). Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 714–722.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM conference on Recommender systems*, pages 39–46.

Donaldson, J. (2007). A hybrid social-acoustic recommendation system for popular music. In *Proceedings of the ACM conference on Recommender systems*, pages 187–190.

Eghbal-Zadeh, H., Lehner, B., Schedl, M., and Widmer, G. (2015). I-vectors for timbre-based music similarity and music artist classification. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 554–560.

Ertekin, Ş. and Rudin, C. (2011). On equivalence relationships between classification and ranking algorithms. *Journal of Machine Learning Research*, 12:2905–2929.

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., and Schmidt-Thieme, L. (2010). Learning attribute-to-feature mappings for cold-start recommendations. In *IEEE International Conference on Data Mining*, pages 176–185.

Hariri, N., Mobasher, B., and Burke, R. (2012). Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the 6th ACM conference on Recommender systems*, pages 131–138.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53.

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

Jannach, D., Lerche, L., and Kamehkhosh, I. (2015). Beyond hitting the hits: Generating coherent music playlist continuations with the right tracks. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 187–194.

John, J. (2006). Pandora and the music genome project. *Scientific Computing*, 23:40–41.

Kamehkhosh, I., Jannach, D., and Bonnin, G. (2018). How automated recommendations affect the playlist creation behavior of users. In *ACM IUI workshop on Intelligent Music Interfaces for Listening and Creation (MILC)*.

Kluver, D. and Konstan, J. A. (2014). Evaluating recommender behavior for new users. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 121–128.

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

Li, N., Jin, R., and Zhou, Z.-H. (2014). Top rank optimization in linear time. In *Advances in neural information processing systems*, pages 1502–1510.

Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). SoRec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

McFee, B., Bertin-Mahieux, T., Ellis, D. P., and Lanckriet, G. R. (2012). The Million Song Dataset Challenge. In *Proceedings of the 21st International Conference on World Wide Web*, pages 909–916.

McFee, B. and Lanckriet, G. R. (2011). The natural language of playlists. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 537–542.

McFee, B. and Lanckriet, G. R. (2012). Hypergraph models of playlist dialects. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*, pages 343–348.

Menon, A. K. and Williamson, R. C. (2016). Bipartite ranking: a risk-theoretic perspective. *Journal of Machine Learning Research*, 17:1–102.

Netflix (2006). Netflix Prize. http://www.netflixprize.com/.

Oord, A. v. d., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651.

Platt, J. C., Burges, C. J., Swenson, S., Weare, C., and Zheng, A. (2002). Learning a gaussian process prior for automatically generating music playlists. In *Advances in neural information processing systems*, pages 1425–1432.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th conference on uncertainty in artificial intelligence*, pages 452–461.

Rudin, C. (2009). The P-Norm Push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research*, 10:2233–2271.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Saveski, M. and Mantrach, A. (2014). Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 89–96.

Schedl, M., Zamani, H., Chen, C.-W., Deldjoo, Y., and Elahi, M. (2017). Current challenges and visions in music recommender systems research. *ArXiv e-prints*.

Schindler, A., Mayer, R., and Rauber, A. (2012). Facilitating comprehensive benchmarking experiments on the million song dataset. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*, pages 469–474.

Schreiber, H. (2015). Improving genre annotations for the million song dataset. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pages 241–247.

Seyerlehner, K., Widmer, G., Schedl, M., and Knees, P. (2010). Automatic music tag classification based on blocklevel features. In *Proceedings of Sound and Music Computing*.

Shao, B., Wang, D., Li, T., and Ogihara, M. (2009). Music recommendation based on acoustic features and user access patterns. *IEEE Transactions on Audio, Speech, and Language Processing*, 17:1602–1611.

Spotify (2018). Spotify – Company Info. `https://newsroom.spotify.com/companyinfo`, retrieved September 2018.

Tintarev, N., Lofi, C., and Liem, C. (2017). Sequences of diverse song recommendations: An exploratory study in a commercial system. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 391–392.

Turrin, R., Quadrana, M., Condorelli, A., Pagano, R., and Cremonesi, P. (2015). 30music listening and playlists dataset. In *Proceedings of the Poster Track of the ACM Conference on Recommender Systems*.

Vall, A., Schedl, M., Widmer, G., Quadrana, M., and Cremonesi, P. (2017). The importance of song context in music playlists. In *Proceedings of the Poster Track of the ACM Conference on Recommender Systems*.

Volkovs, M., Yu, G., and Poutanen, T. (2017). Dropoutnet: Addressing cold start in recommender systems. In *Advances in Neural Information Processing Systems*, pages 4957–4966.

Yoshii, K., Goto, M., Komatani, K., Ogata, T., and Okuno, H. G. (2006). Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *Proceedings of the 7th International Society for Music Information Retrieval Conference*.

Zhang, Y. C., Séaghdha, D. Ó., Quercia, D., and Jambor, T. (2012). Auralist: introducing serendipity into music recommendation. In *Proceedings of the 5th ACM international conference on Web search and data mining*, pages 13–22.

A PROOF OF LEMMA 1

First, we approximate the empirical risk R_θ^{RANK} (with the exponential surrogate) as follows:

$$\begin{aligned}
R_\theta^{\text{RANK}}(f,\mathscr{D}) &= \frac{1}{N}\sum_{u=1}^{U}\sum_{i\in P_u}\frac{1}{M_-^i}\sum_{m':y_{m'}^i=0}\exp\left(-\min_{m:y_m^i=1} f(m,u,i)+f(m',u,i)\right)\\
&= \frac{1}{N}\sum_{u=1}^{U}\sum_{i\in P_u}\frac{1}{M_-^i}\exp\left(-\min_{m:y_m^i=1} f(m,u,i)\right)\sum_{m':y_{m'}^i=0}\exp\left(f(m',u,i)\right)\\
&\approx \frac{1}{N}\sum_{u=1}^{U}\sum_{i\in P_u}\frac{1}{M_-^i}\exp\left(\frac{1}{p}\log\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)\sum_{m':y_{m'}^i=0}\exp\left(f(m',u,i)\right)\\
&= \frac{1}{N}\sum_{u=1}^{U}\sum_{i\in P_u}\frac{1}{M_-^i}\left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)^{\frac{1}{p}}\sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\\
&= \frac{1}{N}\sum_{u=1}^{U}\sum_{i\in P_u}\frac{1}{M_-^i}\left(\left(\sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\right)^{p}\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)^{\frac{1}{p}}\\
&= \frac{1}{N}\sum_{u=1}^{U}\sum_{i\in P_u}\frac{1}{M_-^i}\left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\left(\sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\right)^{p}\right)^{\frac{1}{p}}\\
&= \frac{1}{N}\sum_{u=1}^{U}\sum_{i\in P_u}\frac{1}{M_-^i}\left(\sum_{m:y_m^i=1}\left(\sum_{m':y_{m'}^i=0} e^{-\left(f(m,u,i)-f(m',u,i)\right)}\right)^{p}\right)^{\frac{1}{p}}\\
&= \widetilde{R}_\theta^{\text{RANK}}(f,\mathscr{D}).
\end{aligned}$$

Recall that R_θ^{MTC} is defined as

$$R_\theta^{\text{MTC}}(f,\mathscr{D}) = \frac{1}{N}\sum_{u=1}^{U}\sum_{i\in P_u}\left(\frac{1}{pM_+^i}\sum_{m:y_m^i=1} e^{-pf(m,u,i)}+\frac{1}{M_-^i}\sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\right).$$

Let $\theta^* \in \operatorname{argmin}_\theta R_\theta^{\text{MTC}}$ (assuming minimisers exist), we want to prove that $\theta^* \in \operatorname{argmin}_\theta \widetilde{R}_\theta^{\text{RANK}}$.

Proof. We follow the proof technique in (Ertekin and Rudin, 2011) by first introducing a constant feature 1 for each song, without loss of generality, let the first feature of $\mathbf{x}_m$, $m\in\{1,\dots,M\}$ be the constant feature, i.e., $x_m^0 = 1$. We can show that $\frac{\partial R_\theta^{\text{MTC}}}{\partial\theta}=0$ implies $\frac{\partial \widetilde{R}_\theta^{\text{RANK}}}{\partial\theta}=0$, which means minimisers of R_θ^{MTC} also minimise $\widetilde{R}_\theta^{\text{RANK}}$.

Let $0=\dfrac{\partial R_\theta^{\text{MTC}}}{\partial \beta_i^0}=\dfrac{1}{N}\left(\dfrac{1}{pM_+^i}\sum_{m:y_m^i=1} e^{-pf(m,u,i)}(-p)+\dfrac{1}{M_-^i}\sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\right)$, $i\in P_u$, $u\in\{1,\dots,U\}$,

we have

$$\frac{1}{M_+^i}\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\Bigg|_{\theta=\theta^*} = \frac{1}{M_-^i}\sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\Bigg|_{\theta=\theta^*},\ i\in P_u,\ u\in\{1,\dots,U\}. \tag{18}$$

Further, let

$$\mathbf{0}=\frac{\partial R_\theta^{\text{MTC}}}{\partial \boldsymbol{\beta}_i}=\frac{1}{N}\left(\frac{1}{pM_+^i}\sum_{m:y_m^i=1} e^{-pf(m,u,i)}(-p\mathbf{x}_m)+\frac{1}{M_-^i}\sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\mathbf{x}_{m'}\right),\ i\in P_u,\ u\in\{1,\dots,U\},$$

we have

$$\frac{1}{M_{+}^{i}} \sum_{m:y_m^i=1} e^{-pf(m,u,i)} \mathbf{x}_m \Big|_{\theta=\theta^*} = \frac{1}{M_{-}^{i}} \sum_{m':y_{m'}^i=0} e^{f(m',u,i)} \mathbf{x}_{m'} \Big|_{\theta=\theta^*}, \ i \in P_u, u \in \{1,\dots,U\}. \tag{19}$$

By Eq. (18) and (19), for $i \in P_u$, $u \in \{1,\dots,U\}$, we have

$$\begin{aligned}
&\frac{\partial \widetilde{R}_\theta^{\text{RANK}}}{\partial \boldsymbol{\beta}_i}\Big|_{\theta=\theta^*} \\
&= \frac{1}{NM_-^i}\left[\frac{1}{p}\left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)^{\frac{1}{p}-1} \sum_{m:y_m^i=1} e^{-pf(m,u,i)}(-p\mathbf{x}_m) \sum_{m':y_{m'}^i=0} e^{f(m',u,i)} + \left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)^{\frac{1}{p}} \sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\mathbf{x}_{m'}\right] \\
&= \frac{-1}{NM_-^i}\left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)^{\frac{1}{p}-1}\left[\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\mathbf{x}_m \sum_{m':y_{m'}^i=0} e^{f(m',u,i)} - \sum_{m:y_m^i=1} e^{-pf(m,u,i)} \sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\mathbf{x}_{m'}\right] \\
&= \frac{-1}{NM_-^i}\left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)^{\frac{1}{p}-1}\left[\left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\mathbf{x}_m\right)\left(\frac{M_-^i}{M_+^i}\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right) - \sum_{m:y_m^i=1} e^{-pf(m,u,i)} \sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\mathbf{x}_{m'}\right] \\
&= \frac{-1}{NM_-^i}\left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)^{\frac{1}{p}}\left[\frac{M_-^i}{M_+^i}\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\mathbf{x}_m - \sum_{m':y_{m'}^i=0} e^{f(m',u,i)}\mathbf{x}_{m'}\right] \\
&= \mathbf{0}.
\end{aligned} \tag{20}$$

We further let

$$h(u,i) = \frac{1}{NM_-^i}\left(\sum_{m:y_m^i=1} e^{-pf(m,u,i)}\right)^{\frac{1}{p}} \sum_{m':y_{m'}^i=0} e^{f(m',u,i)}, \ i \in P_u, u \in \{1,\dots,U\},$$

and similar to Eq. (20), we have

$$\frac{\partial h(u,i)}{\partial \boldsymbol{\beta}_i}\Big|_{\theta=\theta^*} = \mathbf{0}, \ i \in P_u, u \in \{1,\dots,U\}. \tag{21}$$

By Eq. (21), for $u \in \{1,\dots,U\}$, we have

$$\frac{\partial \widetilde{R}_\theta^{\text{RANK}}}{\partial \boldsymbol{\alpha}_u}\Big|_{\theta=\theta^*} = \sum_{i\in P_u} \frac{\partial h(u,i)}{\partial \boldsymbol{\alpha}_u}\Big|_{\theta=\theta^*} = \sum_{i\in P_u} \frac{\partial h(u,i)}{\partial \boldsymbol{\beta}_i}\Big|_{\theta=\theta^*} = \mathbf{0}, \tag{22}$$

and

$$\frac{\partial \widetilde{R}_\theta^{\text{RANK}}}{\partial \boldsymbol{\mu}}\Big|_{\theta=\theta^*} = \sum_{u=1}^{U}\sum_{i\in P_u} \frac{\partial h(u,i)}{\partial \boldsymbol{\mu}}\Big|_{\theta=\theta^*} = \sum_{u=1}^{U}\sum_{i\in P_u} \frac{\partial h(u,i)}{\partial \boldsymbol{\beta}_i}\Big|_{\theta=\theta^*} = \mathbf{0}. \tag{23}$$

Finally, by Eq. (20), (22), and (23), $\theta^* \in \operatorname{argmin}_\theta \widetilde{R}_\theta^{\text{RANK}}$. □