DJ-MC: A Reinforcement Learning Agent for Music Playlist Recommendation

Elad Liebman  Maytal Saar-Tsechansky  Peter Stone

University of Texas at Austin

May 11, 2015
Many Internet radio services (Pandora, last.fm, Jango etc.)
Some knowledge of single song preferences
No knowledge of preferences over a sequence
...But music is usually in context of sequence
Key idea - learn transition model for song sequences
Use reinforcement learning
Overview

- Use real song data to obtain audio information
- Formulate the playlist recommendation problem as a Markov Decision Process
- Train an agent to adaptively learn song and transition preferences
- Plan ahead to choose the next song (like a human DJ)
- Our results show that sequence matters, and can be efficiently learned
The adaptive playlist generation problem – an episodic Markov Decision Process (MDP) \((S, A, P, R, T)\). For a finite set of \(n\) songs and playlists of length \(k\):

- **State space** \(S\) – the entire ordered sequence of songs played, \(S = \{(a_1, a_2, \ldots, a_i) | 1 \leq i \leq k; \forall j \leq i, a_j \in \mathcal{M}\}\).

- The set of actions \(A\) is the selection of the next song to play, \(a_k \in A\), i.e. \(A = \mathcal{M}\).

- \(S\) and \(A\) induce a deterministic transition function \(P\). Specifically, \(P((a_1, a_2, \ldots, a_i), a^*) = (a_1, a_2, \ldots, a_i, a^*)\) (shorthand notation).

- \(R(s, a)\) is the utility the current listener derives from hearing song \(a\) when in state \(s\).

- \(T = \{(a_1, a_2, \ldots a_k)\}: \) the set of playlists of length \(k\).
The adaptive playlist generation problem – an episodic Markov Decision Process (MDP) \((S, A, P, R, T)\). For a finite set of \(n\) songs and playlists of length \(k\):

- State space \(S\) – the entire ordered sequence of songs played, \(S = \{(a_1, a_2, \ldots, a_i) | 1 \leq i \leq k; \forall j \leq i, a_j \in \mathcal{M}\}\).
- The set of actions \(A\) is the selection of the next song to play, \(a_k \in A\), i.e. \(A = \mathcal{M}\).
- \(S\) and \(A\) induce a deterministic transition function \(P\). Specifically, \(P((a_1, a_2, \ldots, a_i), a^*) = (a_1, a_2, \ldots, a_i, a^*)\) (shorthand notation).
- \(R(s, a)\) is the utility the current listener derives from hearing song \(a\) when in state \(s\).
- \(T = \{(a_1, a_2, \ldots, a_k)\}\): the set of playlists of length \(k\).
The adaptive playlist generation problem – an episodic Markov Decision Process (MDP) \((S, A, P, R, T)\). For a finite set of \(n\) songs and playlists of length \(k\):

- **State space** \(S\) – the entire ordered sequence of songs played, \(S = \{(a_1, a_2, \ldots, a_i)|1 \leq i \leq k; \forall j \leq i, a_j \in \mathcal{M}\}\).
- The set of actions \(A\) is the selection of the next song to play, \(a_k \in A\), i.e. \(A = \mathcal{M}\).
- **\(S\) and \(A\) induce a deterministic transition function** \(P\). Specifically, \(P((a_1, a_2, \ldots, a_i), a^*) = (a_1, a_2, \ldots, a_i, a^*)\) (shorthand notation).
- \(R(s, a)\) is the utility the current listener derives from hearing song \(a\) when in state \(s\).
- \(T = \{(a_1, a_2, \ldots a_k)\}\): the set of playlists of length \(k\).
The adaptive playlist generation problem – an episodic Markov Decision Process (MDP) \((S, A, P, R, T)\). For a finite set of \(n\) songs and playlists of length \(k\):

- **State space** \(S\) – the entire ordered sequence of songs played, \(S = \{(a_1, a_2, \ldots, a_i)|1 \leq i \leq k; \forall j \leq i, a_j \in \mathcal{M}\}\).

- **The set of actions** \(A\) is the selection of the next song to play, \(a_k \in A\), i.e. \(A = \mathcal{M}\).

- **\(S\) and \(A\) induce a deterministic transition function** \(P\). Specifically, \(P((a_1, a_2, \ldots, a_i), a^*) = (a_1, a_2, \ldots, a_i, a^*)\) (shorthand notation).

- **\(R(s, a)\)** is **the utility the current listener derives from hearing song** \(a\) **when in state** \(s\).

- **\(T = \{(a_1, a_2, \ldots a_k)\}\)**: the set of playlists of length \(k\).
Reinforcement Learning Framework

The adaptive playlist generation problem – an episodic Markov Decision Process (MDP) \((S, A, P, R, T)\). For a finite set of \(n\) songs and playlists of length \(k\):

- **State space** \(S\) – the entire ordered sequence of songs played, \(S = \{(a_1, a_2, \ldots, a_i) | 1 \leq i \leq k; \forall j \leq i, a_j \in \mathcal{M}\}\).
- The set of actions \(A\) is the selection of the next song to play, \(a_k \in A\), i.e. \(A = \mathcal{M}\).
- \(S\) and \(A\) induce a deterministic transition function \(P\). Specifically, \(P((a_1, a_2, \ldots, a_i), a^*) = (a_1, a_2, \ldots, a_i, a^*)\) (shorthand notation).
- \(R(s, a)\) is the utility the current listener derives from hearing song \(a\) when in state \(s\).
- \(T = \{(a_1, a_2, \ldots a_k)\}: \text{the set of playlists of length } k\).
Used a large archive - The Million Song Dataset (Bertin-Mahieux et al.)

Feature analysis and metadata provided by The Echo Nest

44745 different artists, $10^6$ songs

Used features describing timbre (spectrum), rhythmic characteristics, pitch and loudness

12 meta-features in total, out of which 2 are 12-dimensional, resulting in a 34-dimensional feature vector
To obtain more compact state and action spaces, we represent each song as a vector of indicators marking the percentile bin for each individual descriptor:

$$\text{song} = (13.3, 15.4, 0.53, ..., 22.7, 107.65)$$
To obtain more compact state and action spaces, we represent each transition as a vector of pairwise indicators marking the percentile bin transition for each individual descriptor:

\[
\text{song}_a = (13.3, 15.4, 0.53, \ldots, 22.7, 107.65) \\
\text{song}_b = (19.6, 2.2, 0.17, \ldots, 11.8, 56.83)
\]
Modeling The Reward Function

We make several simplifying assumptions:

- The reward function $R$ corresponding to a listener can be factored as $R(s, a) = R_s(a) + R_t(s, a)$.
- For each feature, for each each 10-percentile, the listener assigns reward.
- For each feature, for each percentile-to-percentile transition, the listener assigns transition reward.
- In other words, each listener internally assigns 3740 weights which characterize a unique preference.
- Transitions considered throughout history, stochastically (last song - non-Markovian state signal).

$$\text{totalReward}_t = R_s(a_t) + R_t((a_1, \ldots, a_{t-1}), a_t)$$

where

$$E[R_t((a_1, \ldots, a_{t-1}), a_t)] = \sum_{i=1}^{t-1} \frac{1}{i^2} r_t(a_{t-i}, a_t)$$
Expressiveness of the Model

- Does the model capture differences between separate types of transition profiles? Yes
- Take same pool of songs
- Consider songs appearing in sequence originally vs. songs in random order
- Song transition profiles clearly different (19 of 34 features separable)
Learning Initial Models

1. Learn Initial Song Preferences (Query User)
2. Identify Representative Song Subset
3. Query Random Transition Selections from Subset
4. Use Active Feedback to Initialize Transition Preferences
Planning via Tree Search

1. Filter Upper Median of Corpus
2. Generate Random Rollouts
3. Use Model to Evaluate Rollouts
4. Play 1st Song of Most Promising Rollout
   (Instantiate if planning over types)
Input: Song Corpus M, sequence length K

Learn Song Preferences

Learn Transition Preferences

Plan via Tree Search

Assign Credit to Song and Transition

Obtain Reward from Listener
Experimental Evaluation in Simulation

- Use real user-made playlists to model listeners
- Generate collections of random listeners based on models
- Test algorithm in simulation
- Compare to baselines: random, and greedy
- Greedy only tries to learn song rewards
DJ-MC agent *gets more reward* than an agent which greedily chooses the “best” next song

*Clear advantage* in “cold start” scenarios
Experimental Evaluation on Human Listeners

- Simulation useful, but human listeners are (far) more indicative
- Implemented a lab experiment version, with two variants: DJ-MC and Greedy
- 24 subjects interacted with Greedy (learns song preferences)
- 23 subjects interacted with DJ-MC (also learns transitions)
- Spend 25 songs exploring randomly, 25 songs exploiting (still learning)
- queried participants on whether they liked/disliked songs and transitions
Experimental Evaluation on Human Listeners

- To analyze results and estimate distributions, used bootstrap resampling
- DJ-MC gains substantially more reward (likes) for transitions
- Comparable for song transitions
- Interestingly, transition reward for Greedy somewhat better than random
Experimental Evaluation on Human Listeners

(a) bootstrapped distribution, greedy vs. dj-mc
song reward, 1-25

Song Reward

Before Learning

(b) bootstrapped distribution, greedy vs. dj-mc
transition reward, 1-25

Transition Reward

(c) bootstrapped distribution, greedy vs. dj-mc
song reward, 25-50

After Learning

(d) bootstrapped distribution, greedy vs. dj-mc
transition reward, 25-50
Experimental Evaluation on Human Listeners

(a) bootstrapped distribution, greedy vs. dj-mc
song reward, 1-25

Before Learning

Song Reward

% %

5 10 15 20

(a) cumulative reward

(b) bootstrapped distribution, greedy vs. dj-mc
transition reward, 1-25

Transition Reward

% %

5 10 15 20

(b) cumulative reward

(c) bootstrapped distribution, greedy vs. dj-mc
song reward, 25-50

After Learning

(c) cumulative reward

(d) bootstrapped distribution, greedy vs. dj-mc
transition reward, 25-50

(d) cumulative reward
Related Work

- Chen et al., Playlist prediction via metric embedding, KDD 2012
- Aizenberg et al., Build your own music recommender by modeling internet radio streams, WWW 2011
- Zheleva et al., Statistical models of music-listening sessions in social media, WWW 2010
- Mcfee and Lanckriet, The Natural Language of Playlists, ISMIR 2011
Sequence matters.

Learning meaningful sequence preferences for songs is possible.

A reinforcement-learning approach that models transition preferences does better (on actual human participants) compared to a method that focuses on single song preferences only.

Learning can be done with respect to a single listener and online, in reasonable time and without strong priors.
Questions?

- Input: Song Corpus M, sequence length K
- Learn Song Preferences
- Learn Transition Preferences
- Plan via Tree Search
- Assign Credit to Song and Transition
- Obtain Reward from Listener

Thank you for listening!
A few words on representative selection

(a) clustering

(b) representative selection
1: **Input:** data $x_0 \ldots x_m$, required distance $\delta$
2: Initialize $\textit{representatives} = \emptyset$.
3: Initialize $\textit{clusters} = \emptyset$
4: **representative assignment subroutine, RepAssign**, lines 5-22:
5: **for** $i = 0$ **to** $m$ **do**
6: Initialize $\textit{dist} = \infty$
7: Initialize $\textit{representative} = \textit{null}$
8: **for** $\textit{rep}$ in $\textit{representatives}$ **do**
9: **if** $d(x_i, \textit{rep}) \leq \textit{dist}$ **then**
10: $\textit{representative} = \textit{rep}$
11: $\textit{dist} = d(x_i, \textit{rep})$
12: **end if**
13: **end for**
14: **if** $\textit{dist} \leq \delta$ **then**
15: add $x_i$ to $\textit{cluster}_{\textit{representative}}$
16: **else**
17: $\textit{representative} = x_i$
18: Initialize $\textit{cluster}_{\textit{representative}} = \emptyset$
19: add $x_i$ to $\textit{cluster}_{\textit{representative}}$
20: add $\textit{cluster}_{\textit{representative}}$ to $\textit{clusters}$
21: **end if**
22: **end for**
A few words on representative selection

1: **Input:** data $x_0 \ldots x_m$, required distance $\delta$
2: $t = 0$
3: Initialize $representatives_{t=0} = \emptyset$.
4: Initialize $clusters = \emptyset$
5: repeat
6: $t = t + 1$
7: call $RepAssign$ subroutine, lines 5-22 of Algorithm 2
8: Initialize $representatives_t = \emptyset$
9: for cluster in $clusters$ do
10: \[ representative = \arg\min \sum_{x \in cluster} d(x, s) \text{ s.t. } \forall x \in cluster \cdot d(x, s) \leq \delta \]
11: add $representative$ to $representatives_t$
12: end for
13: until $representatives_t \equiv representatives_{t-1}$
Tree-Search Algorithm

1: **Input:** Song corpus $\mathcal{M}$, planning horizon $q$
2: Select upper median of $\mathcal{M}$, $\mathcal{M}^*$, based on $R_s$
3: $\text{BestTrajectory} = \text{null}$
4: $\text{HighestExpectedPayoff} = -\infty$
5: while computational power not exhausted do
6:     $\text{trajectory} = []$
7:     for $1 \ldots q$ do
8:         $\text{song} \leftarrow$ selected randomly from $\mathcal{M}^*$
9:             (avoiding repetitions)
10:            optional:
11:                $\text{song}_\text{type} \leftarrow$ selected randomly from $\text{song}_\text{types}(\mathcal{M}^*)$
12:                   (avoiding repetitions, $\text{song}_\text{types}()$ reduces the set to clusters)
13:            end for
14:     $\text{expectedPayoffForTrajectory} =$
15:        $R_s(\text{song}_1) + \sum_{i=2}^{q} (R_t((\text{song}_1, \ldots, \text{song}_{i-1}), \text{song}_i) + R_s(\text{song}_i))$
16:     if $\text{expectedPayoffForTrajectory} > \text{HighestExpectedPayoff}$ then
17:         $\text{HighestExpectedPayoff} = \text{expectedPayoffForTrajectory}$
18:         $\text{BestTrajectory} = \text{trajectory}$
19:     end if
20: end while
21: optional: if planning over types, replace $\text{BestTrajectory}[0]$ with song.
22: return $\text{BestTrajectory}[0]$
Model Update

1: **Input:** Song corpus, \( \mathcal{M} \) 
   Planned playlist duration, \( K \)
2: **for** \( i \in \{1, \ldots, K\} \) **do**
3: Use Algorithm 4 to select song \( a_i \), obtaining reward \( r_i \)
4: let \( \bar{r} = \text{average}(\{r_1, \ldots, r_{i-1}\}) \)
5: \( r_{\text{incr}} = \log(r_i / \bar{r}) \)
6: weight update:
   \[ w_s = \frac{R_s(a_i)}{R_s(a_i) + R_t(a_{i-1}, a_i)} \]
7: \[ w_t = \frac{R_t(a_{i-1}, a_i)}{R_s(a_i) + R_t(a_{i-1}, a_i)} \]
8: \( \phi_s = \frac{i}{i+1} \cdot \phi_s + \frac{1}{i+1} \cdot \theta_s \cdot w_s \cdot r_{\text{incr}} \)
9: \( \phi_t = \frac{i}{i+1} \cdot \phi_t + \frac{1}{i+1} \cdot \theta_t \cdot w_t \cdot r_{\text{incr}} \)
10: Per \( d \in \text{descriptors} \), normalize \( \phi_s^d, \phi_t^d \)
   (where \( \phi_x^d \) denotes coordinates in \( \phi_x \) corresponding to 10-percentile bins of descriptor \( d \))
11: **end for**
1: **Input:** Song corpus, $\mathcal{M}$

   Number of preferred songs to be provided by listener, $k_s$

2: initialize all coordinates of $\phi_s$ to $1/(k_s + \#bins)$

3: $\textit{preferredSet} = \{a_1, \ldots, a_{k_s}\}$ (chosen by the listener)

4: **for** $i = 1$ **to** $k_s$ **do**

5: \[ \phi_s = \phi_s + \frac{1}{(k_s+1)} \cdot \theta_s(a_i) \]

6: **end for**
1: **Input:** Song corpus $\mathcal{M}$
   Number of transitions to poll the listener, $k_t$
2: initialize all coordinates of $\phi_t$ to $1/(k_t + \#bins)$
3: Select upper median of $\mathcal{M}$, $\mathcal{M}^*$, based on $R_s$
4: $\delta = 10$th percentile of all pairwise distances between songs in $\mathcal{M}$
5: representative set $\mathcal{C} = \delta$ -medoids ($\mathcal{M}^*$)
6: $song_0 = \text{choose a song randomly from } \mathcal{C}$
7: **for** $i = 1$ **to** $k_t$ **do**
8: $song_i \leftarrow \text{chosen by the listener from } \mathcal{C}$
9: $\phi_t = \phi_t + \frac{1}{(k_t+1)} \cdot \theta_t(song_{i-1}, song_i)$
10: **end for**
Full DJ-MC Architecture

1: **Input:** $\mathcal{M}$ - song corpus, $K$ - planned playlist duration, $k_s$ - number of steps for song preference initialization, $k_t$ - the number of steps for transition preference initialization

Initialization:

1: Initialize song preferences with corpus $\mathcal{M}$ and parameter $k_s$ to initialize song weights $\phi_s$.
2: Initialize transition preferences with corpus $\mathcal{M}$ and parameter $k_t$ to initialize transition weights $\phi_t$.

Planning and Model Update:

1: Simultaneously exploit and learn for $K$ steps with corpus $\mathcal{M}$ (this procedure iteratively selects the next song to play by calling the tree search procedure, and then updates $R_s$ and $R_t$. This is repeated for $K$ steps.)
Joint Feature Dependence

cumulative reward distribution at 10 steps dependent listeners

cumulative reward distribution at 30 steps dependent listeners
Joint Feature Dependence

cumulative reward distribution at 10 steps dependent listeners

cumulative reward distribution at 30 steps dependent listeners