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

Elad Liebman   Maytal Saar-Tsechansky   Peter Stone

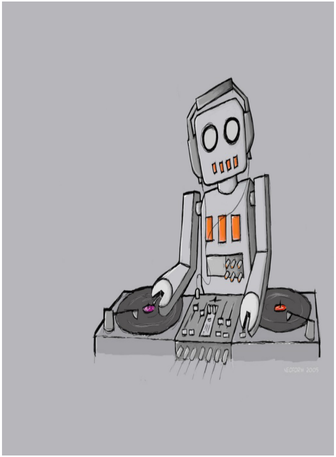
University of Texas at Austin

May 11, 2015

Background & Motivation

- 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

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

Reinforcement Learning Framework

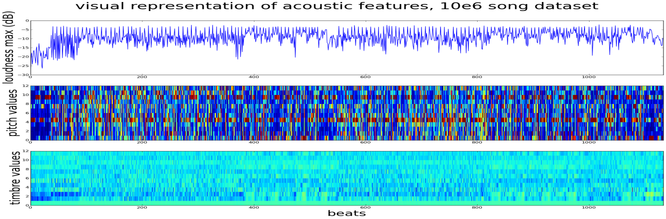
The adaptive playlist generation problem – an episodic Markov Decision Process (MDP) (S, A, P, R, T). For a finite set of n songs and playlists of length k:

- 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**.**
- $T = \{(a_1, a_2, \ldots a_k)\}$: 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**.**

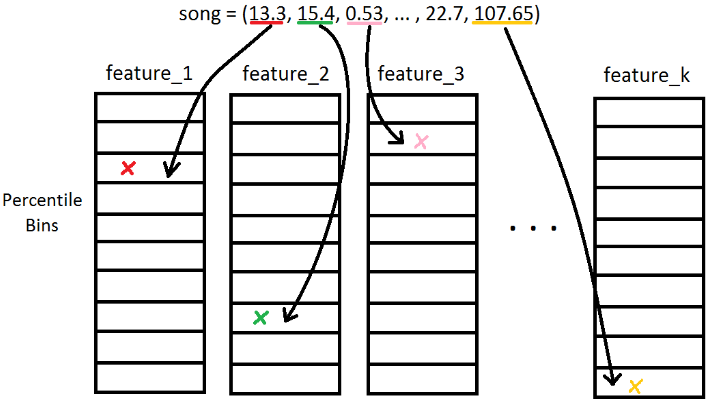
Song Descriptors



- 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

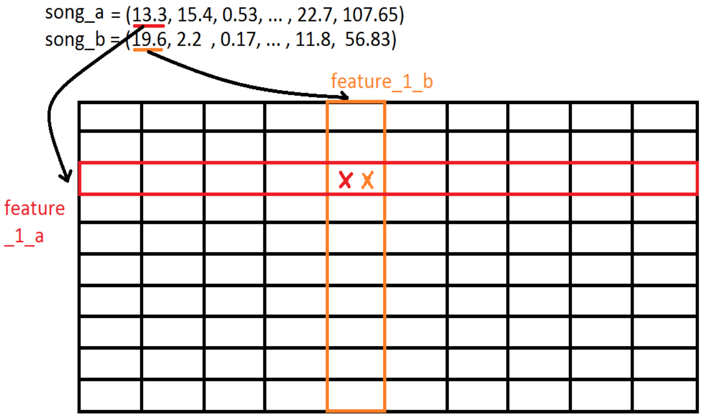
Song Representation

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:



Transition Representation

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:



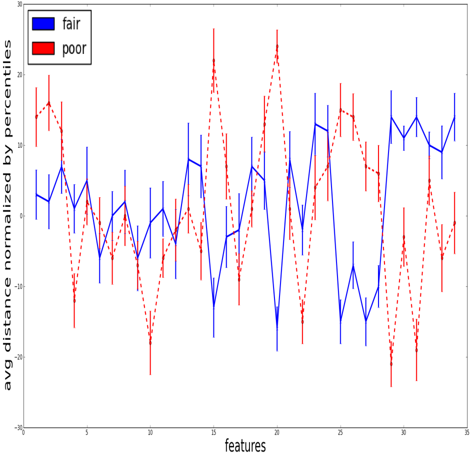
Modeling The Reward Function

We make several simplifying assumptions:

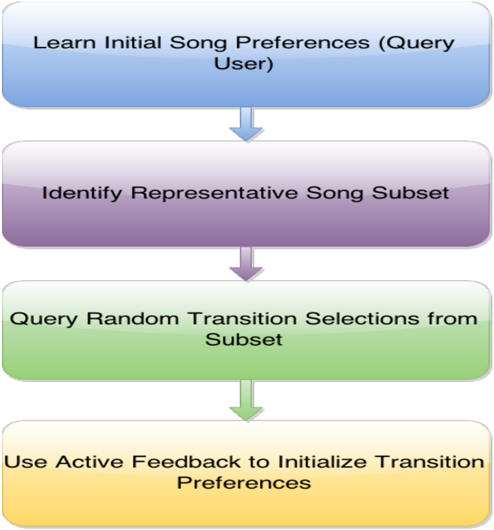
- The reward function R corresponding to a listener can be factored as $R(s, a) = R_s(a) + R_t(s, a)$.
- For each feature, for each each 10-percentile, the listener assigns reward
- for each feature, for each percentile-to-percentile transition, the listener assigns transition reward
- In other words, each listener internally assigns 3740 weights which characterize a unique preference.
- Transitions considered throughout history, stochastically (last song - non-Markovian state signal)
- $totalReward_t = R_s(a_t) + R_t((a_1, \ldots, a_{t-1}), a_t)$ 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

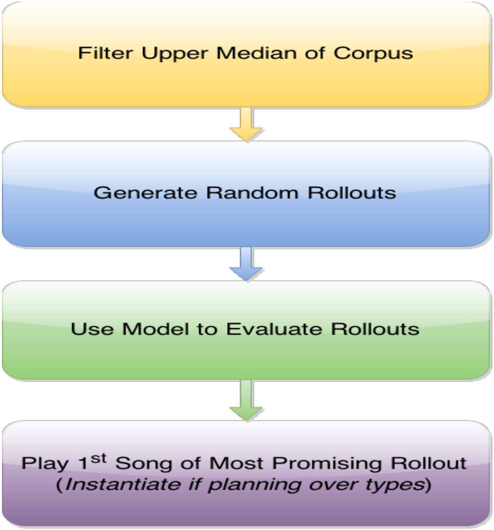
- Does the model capture differences between separate types of transition profiles? Yes
- Take same pool of songs
- Consider songs appearing in sequence originally vs. songs in random order
- Song transition profiles clearly different (19 of 34 features separable)



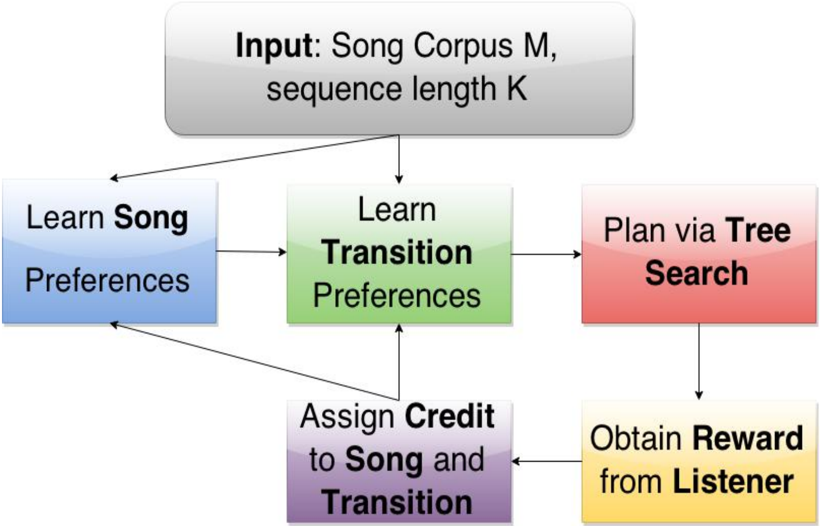
Learning Initial Models



Planning via Tree Search



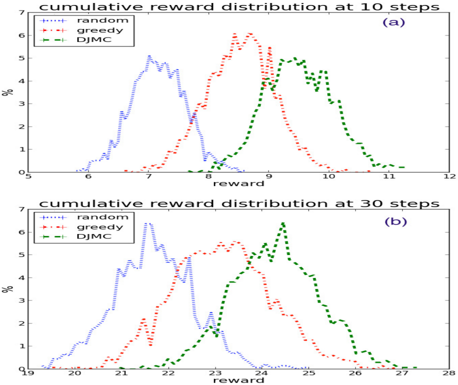
Full DJ-MC Architecture



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



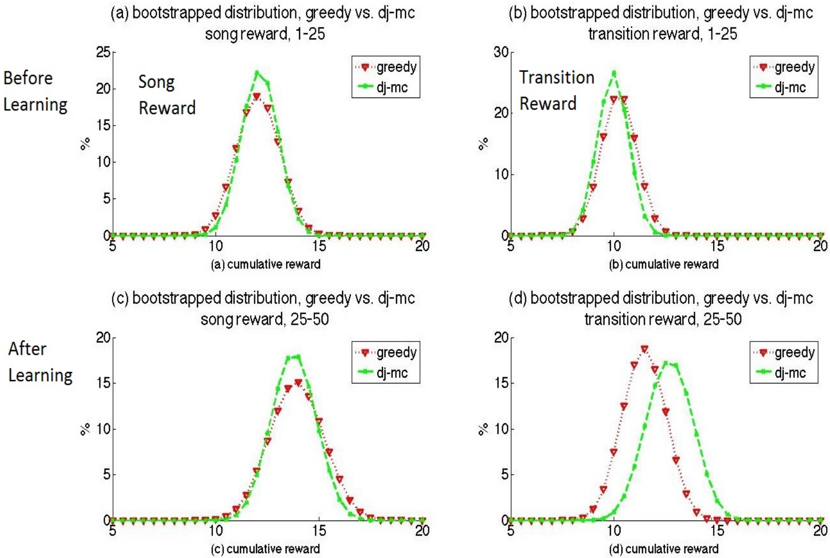
Experimental Evaluation on Human Listeners

- 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

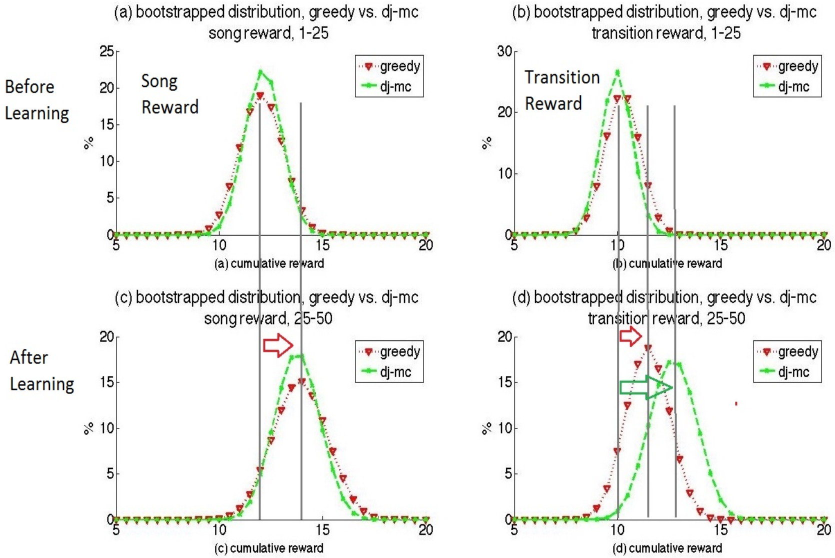
Experimental Evaluation on Human Listeners

- 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



Experimental Evaluation on Human Listeners



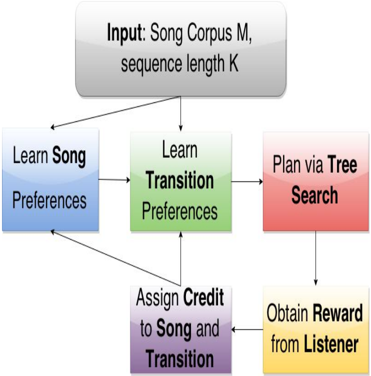
Related Work

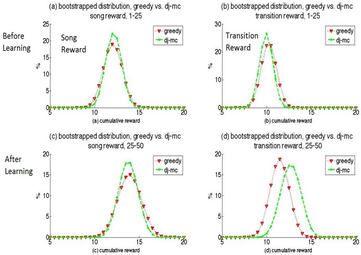
- Chen et al., Playlist prediction via metric embedding, KDD 2012
- Aizenberg et al., Build your own music recommender by modeling internet radio streams, WWW 2011
- Zheleva et al., Statistical models of music-listening sessions in social media, WWW 2010
- Mcfee and Lanckriet, The Natural Language of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

Summary

- 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.

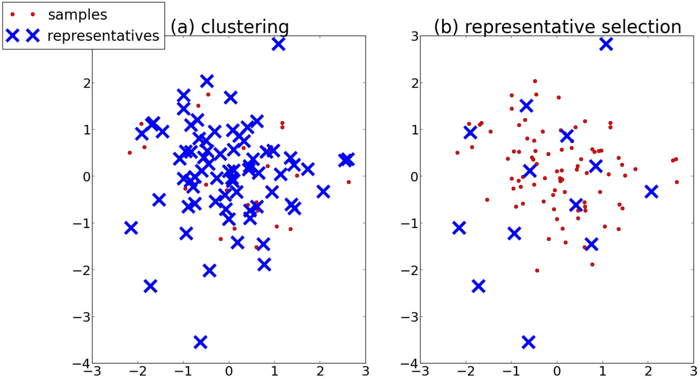
Questions?





Thank you for listening!

A few words on representative selection



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

A few words on representative selection

Input: data $x_0 \dots x_m$, required distance δ
$t = 0$
Initialize $representatives_{t=0} = \emptyset$.
Initialize $clusters = \emptyset$
repeat
  $t = t + 1$
  call *RepAssign* subroutine, lines 5-22 of Algorithm 2
  Initialize $representatives_t = \emptyset$
  for *cluster* in *clusters* **do**
    $representative = \underset{s \in cluster}{argmin} \sum_{x \in cluster} d(x, s)$ s.t. $\forall x \in cluster . d(x, s) \leq \delta$
    add *representative* to $representatives_t$
  end for
until $representatives_t \equiv representatives_{t-1}$

Tree-Search Algorithm

Input: Song corpus $\mathcal{M}$, planning horizon q
Select upper median of $\mathcal{M}$, $\mathcal{M}^*$, based on R_s
BestTrajectory $=$ *null*
HighestExpectedPayoff $= -\infty$
while computational power not exhausted **do**
    trajectory $=$ []
    for $1 \ldots \ldots q$ **do**
        song $\leftarrow$ selected randomly from $\mathcal{M}^*$
        (avoiding repetitions)
        optional:
        song_type $\leftarrow$ selected randomly from *song_types*$(\mathcal{M}^*)$
        (avoiding repetitions, song_types$(\cdot)$ *reduces the set to clusters)*
        add *song* to *trajectory*
    end for
    expectedPayoffForTrajectory $=$

$$R_s(song_1) + \sum_{i=2}^{q} (R_t((song_1, \ldots, song_{i-1}), song_i) + R_s(song_i))$$

    if expectedPayoffForTrajectory $>$ HighestExpectedPayoff **then**
        HighestExpectedPayoff $=$ expectedPayoffForTrajectory
        BestTrajectory $=$ *trajectory*
    end if
end while
optional: if planning over types, replace *BestTrajectory*[0] with song.
return *BestTrajectory*[0]

Model Update

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

Initializing Song Preferences

Input: Song corpus, $\mathcal{M}$
Number of preferred songs to be provided by listener, k_s
initialize all coordinates of ϕ_s to $1/(k_s + \#bins)$
preferredSet = $\{a_1, \ldots, a_{k_s}\}$ *(chosen by the listener)*
for $i = 1$ **to** k_s **do**
    $\phi_s = \phi_s + \frac{1}{(k_s+1)} \cdot \theta_s(a_i)$
end for

Initializing Transition Preferences

```
Input: Song corpus 𝓜
       Number of transitions to poll the listener, k_t
initialize all coordinates of φ_t to 1/(k_t + #bins)
Select upper median of 𝓜, 𝓜*, based on R_s
δ = 10th percentile of all pairwise distances between songs in 𝓜
representative set 𝒞 = δ-medoids (𝓜*)
song_0 = choose a song randomly from 𝒞
for i = 1 to k_t do
    song_i ← chosen by the listener from 𝒞
    φ_t = φ_t + 1/(k_t+1) · θ_t(song_{i−1}, song_i)
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

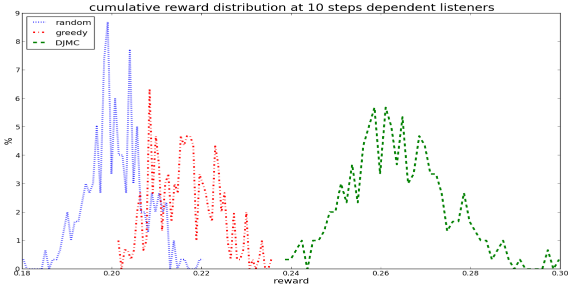
Initialization:

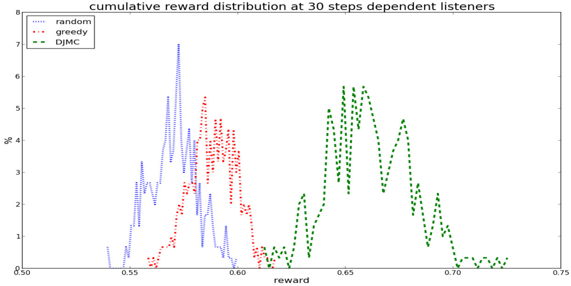
1: Initialize song preferences with corpus $\mathcal{M}$ and parameter k_s to initialize song weights ϕ_s.
2: Initialize transition preferences with corpus $\mathcal{M}$ and parameter k_t to initialize transition weights ϕ_t.

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





Joint Feature Dependence