Playlist Prediction via Metric Embedding

Donadini Eleonora

Stochastic Modelling and Simulation

20 febbraio 2020

Outline

What is a Playlist?

- `Playlist` is a sequence of songs which can be created manually by users or automatically by application
- Some online cloud-based services like `Spotify` allow users to access millions of songs
- Company like Apple have developed algorithms to help `generate playlist automatically`

Topic

We present `Latent Markov Embedding` algorithm for generating `coherent` playlist.

Problem Definition

- `Playlist Prediction`: give e seed or a part of playlist what should be the following sequence of songs in order to naturally reflect manually constructed playlist
- The goal of algorithms is to make the generated playlist coherent
- A `coherent` playlist is defined by a Markov Chain with transition probabilities reflecting similarity of songs

Work Scope

Provide an algorithm named Logistic Markov Embedding

- Models the sequential nature of playlists
- Models playlist as a Markov Chain but not rely on semantic information of songs(genre, emotion, instrument ...)
- Represents songs in Euclidean space

Markov Chain Model of Playlist - I

Goal

The goal is to estimate a generate model of coherent playlist, enable to efficiently sample new playlists

Procedure

Given a connection of songs S_i:

$$S = \{s_1, \dots, s_{|S|}\}$$

we would like to estimate the distribution $Pr(p)$ of coherent playlist $p = (p^{[1]}, \dots, p^{[K_p]})$

- k_p is the length of playlist
- $p^{[i]}$ is one song from S

Markov Chain Model of Playlist - II

Markov Chain

- Stochastic process
- Satisfies the Markov property:

$$P(X_n = x_n \mid X_{n-1} = x_{n-1}, \dots, X_0 = x_0) =$$

$$P(X_n = x_n \mid X_{n-1} = x_{n-1})$$

Markov Chain of Playlist - III

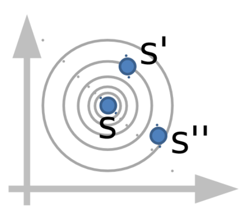
`LME` models playlist as paths through a latent space

- Songs are embedded as point or multiple points in this space
- Euclidean distance between songs reflects the transition probabilities
- The `key learning problem` is to determinate the location of each song using existing playlist as training data

Single Point Model - I

Space representation

Represent each song as a single vector $X(s)$ in d-dimensional Euclidean space



Transition probability

$$Pr(p^{[i]} \mid p^{[i-1]}) = \frac{e^{\|X(p^{[i]}) - X(p^{[i-1]})\|_2^2}}{\sum_{j=1}^{|S|} e^{\|X(s_j) - X(p^{[i-1]})\|_2^2}}$$

Single Point Model - II

If we write:

$$Z(p^{[i-1]}) = \sum_{j=1}^{|S|} e^{\|X(s_j)-X(p^{[i-1]})\|_2^2}$$

$$\Delta(s, s') = \|X(s) - X(s')\|_2$$

Probability of a playlist p:

$$Pr(p) = \prod_{i=1}^{k_p} Pr(p^{[i]} \mid p^{[i-1]}) = \prod_{i=1}^{k_p} \frac{e^{-\Delta(p^{[i]}, p^{[i-1]})^2}}{Z(p^{[i-1]})}$$

Model Regularization - I

- Dimensionality d can be provide some control of overfitting
- Introduce the norm-based regularizers by penalizing the Frobenius norm of $X \in R^{|S|xd}$

$$X = argmax_{X \in R^{|S|xd}} L(D \mid X) - \lambda \|X\|_F^2$$

- For increasing values of λ, this regularizer encourages vectors to stay close to the origin

Solve Location Parameters

- Calculate a matrix T where $T_{a,b}$ are number of transitions from song s_a to s_b in thw training set
- Get a equivalent model

$$L(D \mid X) = \sum_{a=1}^{|S|} \sum_{a=1}^{|S|} Ta, bl(s_a, s_b) - \Omega(X)$$

- Use stochastic gradient training to get the best location of all songs in the training data

Accelerating Solving Process

Computational cost

$O(|S|^2)$ running time is too slow in practical applications

- Do not need to consider all the transitions from current s_i to all the song in S
- Most songs are not likely targets to transit

Landmark heuristic method

only consider a subset C_i^r as possible successors for s_i where r is the percent of the total songs

$$L(D \mid X) = \sum_{a=1}^{|S|} \sum_{s_b \in C_a^r} T_{a,b} l(s_a, s_b) - \Omega(X)$$

What is embedding look like?

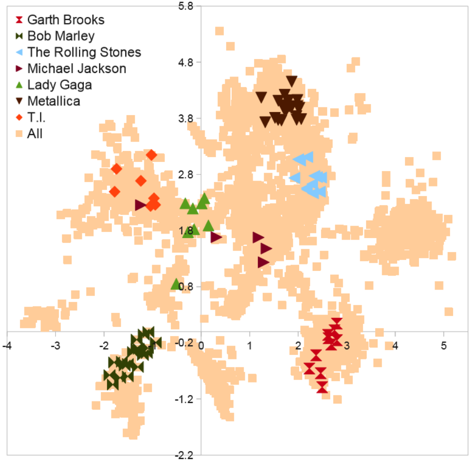


Figure 1: 2D features space

Dataset and Experiment

Dataset

Crawled playlist from Yes.com from Dec 2010 to May 2011 by using its API

set	Quantity
Number of song	163
Number of Train Transitions	134.431
Number of Test Transition	1.191.279

Evaluation

Compare the performance of `LME` versus `bigram` using as metric the `average log-likelihood as metric`:

$$log(Pr(D_{test}))/N_{test}$$

LME vs Bigram

Bigram Model

- Also a first-order Markov Chain
- Transition probabilities $p(s_j \mid s_i)$ are estimated for every pair of songs by calculating the number of appearances

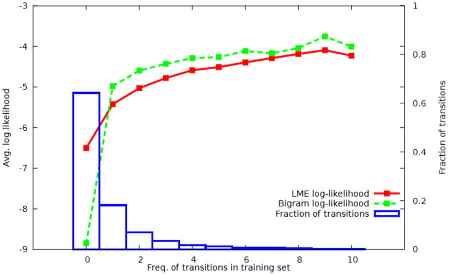


Figure 2: Log likelihood on testing transitions with respect to their frequencies in the training set

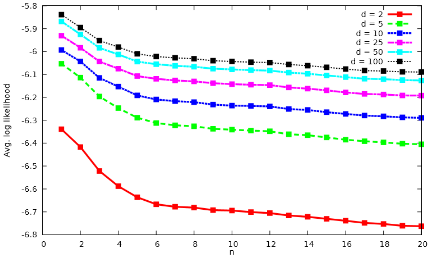
Effect of Landamark Heuristic Method

The landmarck heuristic significantly reduce the training iteration time

r	CPU time/s	Test log-likelihood
0.1	3,08	-6,421977
0.2	3,81	-6,117642
0.3	4,49	-6,058949
0.4	5,14	-6,043897
0.5	5,79	-6,048493
No heuristic	11,37	-6,054263

Capture Coherency of Playlist?

- Build the model on the 1-hop transition in the training dataset
- The test is done on the n-hop transitions in the test dataset



- Song that are sequentially to each other in the playlist are more likely to form a transition pair

How to Generate a Playlist?

Steps

- Given a seed location (a song) in the Euclidean Space
- Repeatedly sample songs from the transition distribution

Problem

The average model represents an average model of playlist but each user may has different preferences

Extending the model

Basic `LME` models can be extended in a variety of ways

- Have only limited means of expressing the popularity of a song
- Add a separate "popularity boost" b_i to song s_i

$$Pr(p^{[i]} \mid p^{[i-1]}) = \frac{e^{-\Delta(p^{[i]},p^{[i-1]})^2+b_i}}{\sum_j e^{-\Delta(s_j,p^{[i-1]})^2+b_j}}$$

- Other extending: user preference, semantic information

References

- S. Chen, J. Moore, D. Turnbull, and T. Joachims. Playlist Prediction via Metric Embedding
- B. McFee and G. R. G. Lanckriet. The natural language of playlists. In International Conference on Music Information Retrieval (ISMIR), 2011
- D. Jurafsky and J. Martin. Speech and language processing, 2008.