Monte Carlo Video Recommendation

Jiangsheng Yu, Jihong Ma

* Condition: for each user, given the information of city, phone specification, and the videos that have been watched in history. Suppose that all the videos are tagged with the well-defined labels.
* Target: based on the given data, how to recommend the favorite videos to each particular user?

Let be the set of all labels (video second classes). The discrete distribution over the labels, say , describes the interest of any concerned user. We call it the **interest distribution** and denote it by .

From the historical data, the interest distribution is surveyed for each user, as well as for each city and phone type. Obviously, the interest distributions enable us to implement the recommendation from various angles. The main algorithm is divided into two steps as follows.

1. Sampling the labels of interest, say , where according to the interest distribution of . The detailed algorithm can be found in the Appendix.
2. Searching the videos with labels in the database, in which the videos are sorted by the correlation to the specific combinations of labels.



It is required that the interest distribution is extracted with a well-defined time window (for instance, two months), and updated by the feedback of user. In addition, the videos with specific labels are sorted and checked manually with high reliability.

**Model Required Input Data:**

We require 3 tables as input data to this model: **user\_profile** table, **user\_watched\_video** table, **video\_label (video\_second\_class)** table, below please find below detailed schema for each input table.

1. **user\_profile table**

The user\_profile table contains basic user profile information, e.g., user index, the city to which the user belongs, the type of user’s device.

We require that each user is assigned a unique String ID

1. Schema: (userIndex: String, residence\_city: String, phone\_type: String)

2. Example table for two users

|  |  |  |
| --- | --- | --- |
| userIndex | residence\_city | phone\_type |
| 001 | Beijing | mate8 |
| 002 | ShenZhen | mate9 |

**2. user\_watched\_video table**

The user\_watched\_video table records the videos each user has watched, where each record include user index, video index and the date when the user watched the video. We require that each user and each video is assigned a unique string index and the user index must be consistent with the user index in user\_profile table

1. Schema: (userIndex: string, videoIndex: string, play\_date: String)
2. Example table of 3 records

|  |  |  |
| --- | --- | --- |
| userIndex | videoIndex | play\_date |
| **001** | **2** | **2016-12-31** |
| **001** | **6** | **20xx-xx-xx** |
| **002** | **29** | **20xx-xx-xx** |

**3. video\_label (video\_second\_class) table**

The video\_label table contains the second class information for each video. We assumed that there are n second classes and each video can belong to multiple (at most n) second classes. The video index must be consistent with the video index in **user\_watched\_video** table

1. Schema: (videoIndex: string, second\_class: String)
2. Example table of 2 videos.

|  |  |
| --- | --- |
| videoIndex | second\_class |
| 1 | action, comedy |
| 2 | action, family |

**Functionalities:**

1. **The distribution of the click counts of all second classes**. Based on the three input tables, we will first output the click count distribution of all second classes for each city, phone type and user as a parquet file.

Let n\_i be the be the number of videos which belong to second class label\_i and are watched by a user or users in the same city or users using the phone of the same type. Let m be the total number of second classes. Then the we define prob\_i = n\_i / sum\_{j = 1 to m} n\_j

* 1. Schema: (residence\_city/phone\_type/userIndex: String, probabilities: Array[Double])
  2. Example output table for 2 city and 5 second classes

|  |  |
| --- | --- |
| residence\_city | prob |
| Beijing | 0.1, 0.3, 0.2, 02, 0.2 |
| Shenzhen | 0.5, 0.1, 0.2, 0.1, 0.1 |

1. **The popularity (click count) of all the videos.** We calculate the popularity of each video in a time window as follows: Let be the number of clicks of video i within the time window and be the total number of videos watched within the time window. Then the popularity of video i is defined as
2. **Recommendation list**. Based on the second class distribution table generated above and user\_watched\_video table, we will recommend to each user a list of videos she/he might be interested in but has not watched before.

We first make recommendations based on the watching history within the lasted time window, e.g., one month. If the recommended videos is less than the required number of videos to recommend, we then make recommendations based on the whole watching history and add the videos to the final recommendation result.

* 1. Input:
     1. k -- number of second classes based on which to make recommendations.
     2. n -- how many videos to recommend
  2. Output

The list of video to recommend for each user. The videos are sorted descending according to probability sum and popularity, where the probability sum is the sum of the label probabilities computed for the city/phone/user, and the popularity is calculated as above. The videos recommend based on the whole watching history will be ranked lower than those recommended based on the latest watching history.

|  |  |  |
| --- | --- | --- |
| userIndex | videoIndex | rank |
| **001** | **2** | **1** |
| **001** | **6** | **2** |
| **002** | **29** | **1** |
| **002** | **18** | **2** |

1. **Interest change over time**. To keep track of the evolution of the distribution of the second-class, and check if the distribution is stable over time, we compute the distance of the distributions between each two consecutive time windows. Specifically, let the distributions of two time windows be:

We calculate Kullback-Leibler divergence as the distance between the two distributions

For the case when , let m be the number of zero items in Q, then we have

We output the following statistics of the KL divergence of all users, cities or phone types: minimum, maximum, average, divergence.

# Appendix

Algorithm: generating a random number of discrete distribution , denoted by by the following method.

1. Draw a random number from the uniform distribution .
2. Let be the least integer satisfying that , then let .

