Movie Recommendation Based on HMM and Low-Rank Matrix Completion

Xu Bowen, Jiang Haoran, Gong Ke, Hu Jintian, Peng Yucen

January 7, 2023

Abstract

In this project, we focus on movie recommendation systems based on artificial intelligence techniques. We explore two common methods, via Hidden Markov Models (HMM) and via low-rank matrix completion, implement them and compare their performances.

1 Introduction

Recommendation systems based on artificial intelligence techniques have gain increasing popularity over the years, since they make people easier to find what they need. Recommendation systems have their applications in many fields, ranging from music, movies to online shopping and search engines. Researches [1, 2] have shown that with enough information of users, it is possible for an automatic movie recommendation system to provide satisfactory recommendations.

In modern recommendation systems, various algorithms in the field of machine learning and deep learning can be of help in an effort to achieve good performance. For example, [3] proposed the movie recommendation system MOVREC that uses collaborative filtering [4]. Another system based on inductive learning, a iterative and inductive machine learning algorithm used for generating classification rules, is discussed in [5]. Moreover, unsupervised methods like clustering [6] play a role in large-scale recommendation systems. Although clustering is somewhat inefficient, it is possible to utilize clustering as the initial step for contracting the selection of significant neighbors [7] in collaborative filtering.

In this project, we explore two different ways of implementing the movie recommendation system: via a Hidden Markov Model, and via low-rank matrix completion. The Hidden Markov Model (HMM) [8] is a statistical Markov model, in which the system is modeled according to a Markov process with unobservable (hidden) states. Low-rank matrix completion is a well-known optimization problem with plenty of researches such as [9], [10] and [11]. We implement these algorithms and have them tested on some data set to see their advantages and drawbacks.

2 Via HMM

2.1 HMM Modeling

The theory of hidden Markov model is the further development of Markov model, and Markov model is the modeling of Markov process.

As shown in Figure **??** , Markov model regards a general stochastic process as a series of transitions between states. Where, the state at time t is expressed by q_t, and its value is taken to the state set $S = [s_1, s_2..., s_n]$. The probability of the latter state in the model is determined by its previous state, and the transformation relationship between the states is mainly represented by ”transition probability”.

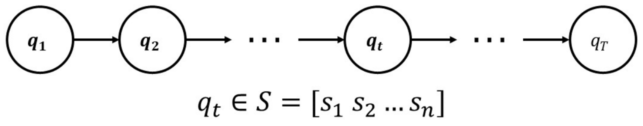

Figure 1: Markov model

As shown in Figure ?? , the hidden Markov model actually introduces "observation value" for each state based on the Markov model. o_t is the observation value output by q_t for the state at time t. In this process, the state is usually hidden, that is, it cannot be directly observed through the experimental output. Therefore, hidden Markov model describes a double random process, that is, hidden Markov chain with a certain number of states and a set of explicit random functions.

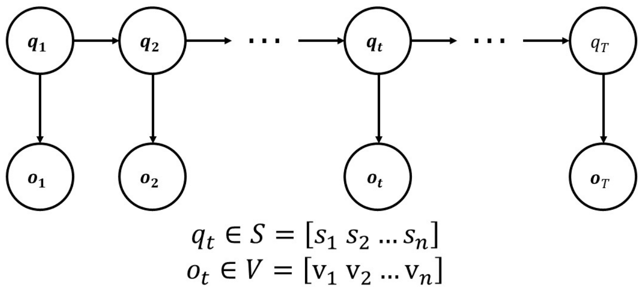
q_1
q_2
q_t
q_T
o_1
o_2
o_t
o_T
$q_t \in S = [s_1\ s_2 \dots s_n]$
$o_t \in V = [v_1\ v_2 \dots v_n]$

Figure 2: Hidden Markov model

For each user, assuming that the user can be classified into some hidden clusters, the user's preference can be used to predict the cluster to which they belong, and recommendations for users can be obtained from their possible clusters. In reality, users' preferences may change, which can be interpreted as a change in their clusters. A widely used approach is to treat these clusters as hidden states and use Hidden Markov Model to model the transition between hidden states and the relationship between hidden states and movies. Let the number of hidden states be k, and the HMM is defined by $\lambda = (A, B, \pi)$ where $A \in \mathbb{R}^{k\times k}$ is the transition probability matrix represents the transition probability over hidden states, $B \in \mathbb{R}^{k\times n}$ is the emission probability matrix represents the relationship between hidden states and movies, $\pi \in \mathbb{R}^{k}$ is the initial probability vector.

2.2 HMM training

To determine the parameters of λ, the Baum-Welch algorithm, which does not rely on hidden state annotation nor prior probability, is necessarily introduced. Given an user's preference sequence $O = (O_1 = o_1, \dots, O_T = o_t)$, the algorithm can be described as:

step 1: Initialize

Set $\theta = (A, B, \pi)$ with random conditions. Let X_t be a discrete hidden random variable with k possible values.

step 2: Forward procedure

Let $\alpha_i(t) = P(O_1 = o_1, \dots, O_t = o_t, O_t = i \mid \theta)$.We can calculate $\alpha_i(t)$ as,

1. $\alpha_i(1) = \pi_i b_1(o_1)$
2. $\alpha_i(t+1) = b_i(o_{t+1}) \sum_{j=1}^{N} \alpha_j(t) a_{ji}$

step 3: Backward procedure

Let $\beta_i(t) = P(O_{t+1} = o_{t+1}, \ldots, O_T = o_T \mid X_t = i, \theta)$. We can calculate $\beta_i(t)$ as,

1. $\beta_i(T) = 1$
2. $\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(o_{t+1})$

step 4: Calculate temporary variables

1. $\gamma_i(t) = P(X_t = i \mid O, \theta) = \frac{P(X_t=i, O|\theta)}{P(O|\theta)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$
2. $\xi_{ij}(t) = P(X_t = i, X_{t+1} = j \mid Y, \theta) = \frac{P(X_t=i, X_{t+1}=j, O|\theta)}{P(O|\theta)} = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(o_{t+1})}{\sum_{k=1}^{N} \sum_{w=1}^{N} \alpha_k(t) a_{kw} \beta_w(t+1) b_w(o_{t+1})}$

step 5: Update

1. $\pi_i^* = \gamma_i(1)$
2. $a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$
3. $b_i^*(v_k) = \frac{\sum_{t=1}^{T} 1_{o_t = v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$

step 6: Termination

If $\theta = (A, B, \pi)$ is converged or the number of iterations is larger than the max iteration set at the beginning, then stop and return θ. Otherwise, go to step 2.

2.3 HMM predicting

In this section, w e want to calculate the probability of the user giving high marks to a movie m. This can be interpreted as calculating probability of an observed sequence:

$$P(m) = \sum_{i=1}^{k} P(m|S_i) P(S_i|o_1 \ldots t)$$

where S is the last state.

3 Via Low-Rank Matrix Completion

3.1 Problem Formulation

Suppose there are m users and n movies. For a video platform or a movie theater, it is possible to collect users' feedback on the movies they have seen by sending questionnaires or having other forms of surveys. Suppose $M \in \mathbb{R}^{m \times n}$ is the rating matrix, for which the (i, j)-th entry M_{ij} represents the rating of the i-th user for the j-th movie. Note that there might be many entries missing, because the users only rate the movies they have watched.

Let Ω be the set of pairs (i, j) such that the entry M_{ij} is known. The movie recommendation problem could be seen as constructing a suitable $X \in \mathbb{R}^{m \times n}$, in which $X_{ij} = M_{ij}$ for all $(i, j) \in \Omega$, and that the matrix X best represents the true preference of users. In reality, the ratings for movies have locality with respect to both users and movies, i.e. similar movies may get similar ratings from a same group of people, and similar users may share a common taste on movies. Therefore, it is reasonable to construct X by filling the unknown entries in a way that minimizes its rank, which leads to the following problem formulation.

Problem 1 (Low-rank matrix completion).

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} \quad & \text{rank}(X) \\ \text{subject to} \quad & X_{ij} = M_{ij}, \quad \forall (i, j) \in \Omega. \end{aligned}$$

Unfortunately, Problem 1 is non-convex and NP-Hard. To solve it efficiently, we may turn to the following convex relaxation.

Problem 2.

$$\begin{aligned} \min_{X \in \mathbb{R}^{m\times n}} \quad & \|X\|_* \\ \text{subject to} \quad & X_{ij} = M_{ij}, \quad \forall (i,j) \in \Omega. \end{aligned}$$

Remark 1 (Nuclear norm)**.** *The* nuclear norm *of a matrix* X*, denoted* $\|X\|_*$*, is defined to be the sum of singular values of* X*, i.e.*

$$\|X\|_* = \sum_i \sigma_i(X).$$

The nuclear norm minimization problem (Problem 2) is a convex problem. We can solve it by minimizing its quadratic penalty function

$$\mu\|X\|_* + \frac{1}{2}\sum_{(i,j)\in\Omega} (X_{ij} - M_{ij})^2, \tag{1}$$

where μ is the *penalty factor* or *regularization factor*.

3.2 Proximal Gradient Method

To describe the Proximal Gradient Method, we first introduce the *proximal operator*.

Definition 1 (Proximal operator)**.** *For a convex function* h*, the* proximal operator *of* h *is defined as*

$$\mathrm{prox}_h(x) = \arg\min_{u \in \mathbf{dom}\, h} \left(h(u) + \frac{1}{2}\|u - x\|^2 \right).$$

The Proximal Gradient Method aims to solve optimization problems of the following form

$$\min \quad \psi(x) = f(x) + h(x),$$

where f is differentiable with $\mathbf{dom} f = \mathbb{R}^n$, and h is convex (maybe non-smooth) whose proximal is easy to compute. The Proximal Gradient Method is an iterative algorithm that performs gradient descent on the smooth part (f), and uses the proximal operator on the non-smooth part (h). The algorithm framework is as follows.

Algorithm 1 Proximal Gradient Method

```
Input: Initial point $x^0$.
  $k \leftarrow 0$
  while not converge do
      $x^{k+1} \leftarrow \mathrm{prox}_{t_k h}\left(x^k - t_k \nabla f\left(x^k\right)\right)$
      $k \leftarrow k + 1$
  end while
  return $x^k$
```

At each step, t_k is the *step size* or the *learning rate* that could be either set to a constant or obtained from a line-search algorithm. From another perspective, Algorithm 1 could be seen as a combination of the gradient and sub-gradient descent methods if we note that

$$\mathrm{prox}_{t_k h}\left(x^k - t_k \nabla f\left(x^k\right)\right) = x^k - t_k \nabla f\left(x^k\right) - t_k g^k, \quad g^k \in \partial h\left(x^{k+1}\right).$$

Let $P \in \mathbb{R}^{m\times n}$ be a matrix with the (i,j)-th entry defined as

$$P_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases}$$

For a given regularization factor μ, the objective function (1) could be seen as $f(X) + h(X)$, where

$$f(X) = \frac{1}{2}\|P \odot (X - M)\|_F^2, \quad \nabla f(X) = P \odot (X - M)$$

and

$$\mathrm{prox}_{t_k h}(X) = U\mathbf{diag}\left(\max\{|d| - t_k\mu, 0\}\right) V^T.$$

Here $X = U\mathbf{diag}(d)V^T$ is the *thin SVD* of X. In this way, we can derive the iteration formula for minimizing (1) using the Proximal Gradient Method:

$$\begin{aligned} Y^k &= X^k - t_k P \odot \left(X^k - M\right), \\ X^{k+1} &= \mathrm{prox}_{t_k h}\left(Y^k\right). \end{aligned} \tag{2}$$

In combination with the penalty function method, Problem 2 could be solved using the following algorithm.

Algorithm 2 Nuclear Norm Minimization via Penalty Function and Proximal Gradient Method

```
Input: Initial point $X^0$, ultimate factor $\mu^*$, initial factor $\mu_0$, $\gamma \in (0, 1)$.
  $k \leftarrow 0$
  while $\mu_k \geqslant \mu^*$ do
      Minimize the penalty function (1) using Algorithm 1. Let $X^{k+1}$ be the optimal solution.
      if $\mu_k = \mu^*$ then
          return $X^{k+1}$
      else
          $\mu_{k+1} \leftarrow \max\{\mu^*, \gamma\mu_k\}$
          $k \leftarrow k + 1$
      end if
  end while
```

4 Numerical Experiment

After the theoretical analysis of the above two models: HMM Algorithm and Low-Rank Matrix Completion, we will simulate them and test their recommendation accuracy and running time.

First, our dataset comes from a well-known movie recommendation website: Movielens. We downloaded more than 100k reviews of movies (943 users, 1682 movies) from this website, each rating including: user number, movie number, and users' rating of the movie.

We will use cross-validation to calculate the recommendation accuracy of the algorithm, that is, to randomly select 10 items from each user's ratings as the test set, and the remaining ratings as the training set. We first use the training set to train our model, and then use the trained model to recommend movies to users in the test set, and compare it with the test set to obtain the correct number of recommendations. To calculate the accuracy of the model recommendation, we use the following standard:

In our standard, we use Recall Rating to represent the accuracy of the recommendation. Its physical meaning represents the rate of movies the model recommends that should be recommended to users. Its numerical definition is as follows:

$$\mathrm{REC} = \frac{1}{N - M}\sum_i \frac{R_i}{T_i}$$

Where the variables are defined as below:

- N: Total number of users.
- M: Number of users without data in the test set.
- R_i: For user i with data in test set, the correct number which is recommended.

- T_i: Number of true recommendations to user i.

The result of numerical experiment is as below:

	HMM	Low-Rank Matrix Completion
Accuracy(Recall Rate)	25%	22%
Running Time	1 hour 40 minutes	41 seconds

where the run time represents the sum of the training time and the prediction time.
From the table above, we can see the conclusion:

- HMM Algorithm: slightly high accuracy but extremely long running time.
- Low-Rank Matrix Completion: Almost same accuracy as traditional HMM Algorithm but extremely fast running speed.

To conclude the experimental results, we can see that the new low-rank matrix recovery method is very close to the traditional HMM method in the recommendation accuracy (recall rate), but for the running time, the low-rank matrix recovery method is much better than the traditional HMM method. This also verifies the superiority of the low-rank matrix recovery method and conforms to the improvement of the time complexity mentioned in Motivation.

5 Conclusion

We replicate a simple HMM based movie recommendation system, and test its initial efficiency under two different scale datasets. It is noticed that starting a simple HMM model using random initialization and manual hyperparameter settings on a real large database requires unacceptable pretrain time.

To overcome the shortcoming, we proposed a Low-Rank Matrix Completion based movie recommendation method for high efficiency recommending and feature mining. Different from the HMM algorithm, Low-Rank Matrix Completion is based on optimization theory and focuses more on theoretical calculation, so it can be seen as a new horizon in the recommendation system.

And in the numerical experiment, the results show that the proposed method: Low-Rank Matrix Completion is much faster than the HMM based method and has no inferior recall rate. Moreover, it is noted that the proposed method can help determine the initial probability and hyperparameters of HMM, which significantly reduces the amount of training required for HMM.

To conclude, in this paper, we provide an optimization algorithm that can replace the traditional HMM algorithm: Low-Rank Matrix Completion. Its superior performance (mainly due to extremely short running time) makes it possible in many cases to replace the once widely used HMM algorithm for recommendation in industries such as movies. Although HMM based recommendation system is no longer the most mainstream method, we believe that the proposed method: Low-Rank Matrix Completion is still a valuable idea for other systems relying on HMM training.

References

[1] C.-G. Chiru, C. Preda, V.-N. Dinu, and M. Macri, "Movie recommender system using the user's psychological profile," in *2015 IEEE international conference on intelligent computer communication and processing (ICCP)*, pp. 93–99, IEEE, 2015.

[2] R. Hande, A. Gutti, K. Shah, J. Gandhi, and V. Kamtikar, "Moviemender-a movie recommender system," *International Journal Of Engineering Sciences & Researchtechnology ISSN*, pp. 2277–9655, 2016.

[3] M. Kumar, D. Yadav, A. Singh, and V. K. Gupta, "A movie recommender system: Movrec," *International Journal of Computer Applications*, vol. 124, no. 3, 2015.

[4] A. Bilge, C. Kaleli, I. Yakut, I. Gunes, and H. Polat, "A survey of privacy-preserving collaborative filtering schemes," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 08, pp. 1085–1108, 2013.

[5] P. Li and S. Yamada, "A movie recommender system based on inductive learning," in *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, vol. 1, pp. 318–323, IEEE, 2004.

[6] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[7] G. Arora, A. Kumar, G. S. Devre, and A. Ghumare, "Movie recommendation system based on users' similarity," *International journal of computer science and mobile computing*, vol. 3, no. 4, pp. 765–770, 2014.

[8] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[9] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 665–674, 2013.

[10] B. Vandereycken, "Low-rank matrix completion by riemannian optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1214–1236, 2013.

[11] F. J. Király, L. Theran, and R. Tomioka, "The algebraic combinatorial approach for low-rank matrix completion.," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1391–1436, 2015.

[12] H. Liu, J. Hu, Y. Li, and Z. Wen, "Optimization modeling, algorithms and theory," 2020.