IS6101 Project Report: Replicating the Hidden Markov Model for Collaborative Filtering

Lu Chung I

1. Introduction

The project consists of building the Hidden Markov Model (HMM) as described in (Sachoo et al 2012) and testing the model against the Netflix Prize dataset. The purpose of the model is for collaborative filtering, specifically, recommending a set of movies for a single time period after the training period of the model.

The rest of the report is organised as follows. In the next section, I describe the model specifics and the objective. This is then followed by a description of the dataset and the pre-processing required to prepare the data for the model. Section 4 will detail the implementation of the model in Python including the inference and optimisation approach as well as the challenges faced. Finally, I show the results of experimenting with different hyperparameters and dataset slicing before finally drawing some conclusions.

2. The Model and Objective

The key contribution of the original paper was to remove the assumption that user preferences were static which was common in the literature at the time. Therefore, the proposed model is an HMM which is a mixed membership model to a discrete set of latent classes that changes across time periods. There are typically two types of changes that can happen to users' preferences. One type is due to idiosyncratic causes which are usually difficult to identify from the available data. The other type would stem from systematic causes which have a repeating pattern across users. For example, a new subscriber to Netflix may go through an initial exploratory phase before settling to a more recognised profile. It is the latter type of changes that can be modelled by the HMM from the data as global transition probabilities that are shared across all users.

Parameters of the Model

The latent classes are represented by the variable Z which can take on K discrete values. K is a hyperparameter that would require tuning. Z_u^t is the latent class of user u at time period t. Each user will have degrees (probability) of membership to each of the latent classes in a particular time period. If we know the membership of the user to the latent classes, then it is sufficient to know the distribution of the movies that will be rated in the time period. User class memberships change from period to period based on the global transitional probability matrix A of size K by K. The element A_{ij} is the probability of transitioning from latent class i to latent class j. The prior distribution of a user's class membership is represented by the vector π of size K.

For each latent class, there is a set of parameters *(a, b)* which parameterise a Negative Binomial Distribution (NBD). That NBD supports non-negative integers hence is suitable to model the number of movies a user rates in a time period. This is represented by the variable N_u^t. The NBD is formulated as a Gamma-Poisson mixture. The NBD is a Poisson(λ) where λ follows a Gamma distribution parameterised by shape parameter a and rate parameter b.

The specific movies rated by a user in a time period is represented the variable I_u^t and follows a multinomial distribution. The distribution is conditioned on N_u^t as the number of trials and a set of probabilities for each movie in the available library. There are K sets of these probabilities for each latent class represented by the K by $|I|$ matrix θ where $|I|$ is the number of unique movies.

The Objective

The objective is to produce a set of recommended movies for one time period after the available data. The recommended list contains the movies most likely to appear in the prediction period. In order to do this, the parameters in the model must be inferred from the available data using a Bayesian approach. In essence, we seek to maximise the likelihood of the data by varying the parameters A, π, a, b and θ. This would be done using the Expectation Maximisation (EM) algorithm starting with a random initialisation. Once a local maximum has been obtained, the inferred parameters will be used to compute the probability of a movie appearing in the next time period as follows.

$$P(i \in I_u^{T+1}) = \sum_{k=1}^{K} P(Z_u^{T+1} = k) P(i \in I_u^{T+1} | a_k, b_k, \theta_k) \tag{1}$$

where T is the final period in the training data.

The first term in the summation is the latent class probability which can be calculated as follows.

$$P(Z_u^{T+1} = k) = \sum_{i=1}^{K} P(Z_u^T = i | I_u^{1:T})\, P(Z_u^{T+1} = k | Z_u^T = i)$$

The first term is the posterior of the latent class for a user in the final period of the data and the second term is the transitional probability obtained from the matrix A.

The latter term of equation (1) can be calculated as the joint distribution of N_u^{T+1} and I_u^{T+1} then marginalising N_u^{T+1} as follows.

$$P(i \in I_u^{T+1} | a_k, b_k, \theta_k) = \sum_{N_u^{T+1}=0}^{\infty} P(N_u^{T+1} | a_k, b_k) P(i \in I_u^{T+1} | N_u^{T+1}, \theta_{ki}) \tag{2}$$

The latter term of equation (2) can be calculated as one minus the probability that movie i does not appear in the period. The probability that the movie i does not appear is simply any movie other than movie i appearing for N_u^{T+1} times.

$$P(i \in I_u^{T+1} | N_u^{T+1}, \theta_{ki}) = 1 - P(i \notin I_u^{T+1} | N_u^{T+1}, \theta_{ki}) = 1 - (1 - \theta_{ki})^{N_u^{T+1}} \tag{3}$$

Equations (2) and (3) allow us to simply equation (1) as follows using the fact that summation of probabilities of all values of N_u^{T+1} equals one.

$$\begin{aligned} P(i \in I_u^{T+1} | a_k, b_k, \theta_k) &= \textstyle\sum_{N_u^{T+1}=0}^{\infty} P(N_u^{T+1} | a_k, b_k) \left[1 - (1 - \theta_{ki})^{N_u^{T+1}}\right] \\ &= 1 - \textstyle\sum_{N_u^{T+1}=0}^{\infty} P(N_u^{T+1} | a_k, b_k)(1 - \theta_{ki})^{N_u^{T+1}} \end{aligned}$$

The latter term can be expressed as the moment generating function of a NBD, $M(s) = [1 + b(1 - e^s)]^{-a}$ where $s = log(1 - \theta_{ki})$. This allows us to simplify equation (1) into the final form as follows.

$$P(i \in I_u^{T+1} | a_k, b_k, \theta_k) = 1 - \textstyle\sum_{k=1}^{K} P(Z_u^{T+1} = k)\ (1 + b_k \theta_{ki})^{-a_k} \tag{4}$$

Equation (4) allow us to rank each movie based on the probability it would appear in the next time period. The ranking can be done using $R(i, u) = -\sum_{k=1}^{K} P(Z_u^{T+1} = k)\ (1 + b_k \theta_{ki})^{-a_k}$ without any change in order compared with equation (4). From this ranked list of movies, one can select the top N movies to be recommended to the user.

It is important to note that equation (4) does not exclude movies that have appeared in a user's history. In fact, it is quite likely for movies that have appeared in a user's history to rank high based on equation (4). However, while it is not impossible for a user to re-rate a movie, such occurrences would be rare and more importantly, unhelpful in a recommendation context. Therefore, it would make sense to remove any movies from a user's history from the final recommended list of movies. This particular action is not explicit in the original paper but would be done for the experiments in this report.

Algorithm

The full algorithm for generating the recommendation list is summarised as follows. Details on key steps are in section 4.

1. Collect data for the training period

2. Initialise $\pi,\ A,\ \theta, (a, b)$ to random values
3. E Step: Compute posterior probabilities $P(Z_u^t | I_u^{1:T})$ and $P(Z_u^{t-1}, Z_u^t | I_u^{1:T})$ for $t = 1, 2, \dots, T$ using the forward-backward algorithm with current set of parameters
4. M Step: Update parameters using the posterior probabilities
5. If expected log likelihood has not converged, go back to step 3
6. For each user u, compute ranking $R(i,\ u)$ of each item i for time T+1
7. Recommend top N movies with highest $R(i,\ u)$

3. Data and Pre-processing

The model will be tested against the Netflix Prize dataset. This dataset was provided by Netflix back in October 2006 for an open competition to find the best collaborative filtering algorithm. The dataset can be summarised as follows.

- >100m movie rating entries
- Each entry contains 4 pieces of data: user ID, movie ID, movie rating and date stamp of the entry
- 17,770 unique movie titles in movie IDs from 1 to 17,770
- 480,189 unique users in non-sequential user IDs
- Movie ratings are integers from 1 to 5
- Date stamped from Oct 1998 to Dec 2005

Although the specifications indicated that the data starts from Oct 1998, it was found that the earliest date stamp was Dec 1999 instead. While movie ratings were provided which indicated the degree of preference of the user for the movie, the model works on an implicit rating assumption. In other words, ratings are ignored and all movies rated in a time period are on an equal footing. Therefore, the pre-processing of the data consists of converting all ratings belonging to a user in a time period to a binary vector. The size of the vector is the number of unique movies $|I|$. Each movie rated in the period corresponds to a value of 1 e.g. if movie ID 10 was rated then the 10th element of the vector will have a value of 1. Each time period is specified as a calendar month hence there is a total of 73 periods in the dataset.

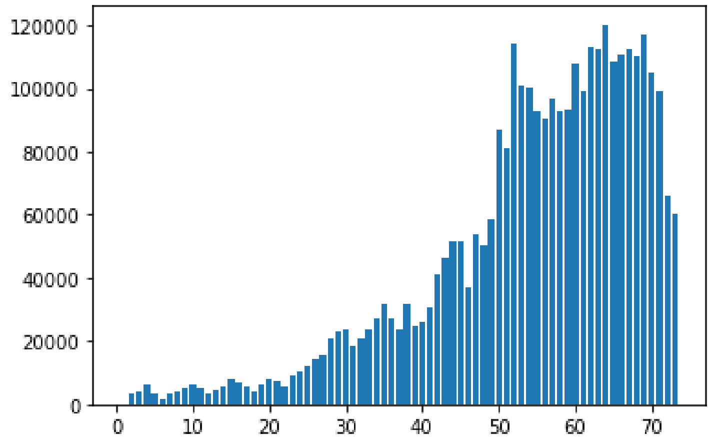

Figure 1 - Bar plot of number of ratings in each time period based on the subset with 1,212 users that have at least 2,000 ratings. Y-axis is the number of ratings in aggregate while X-axis is the time period (calendar month)

As detailed in the next section on implementation, there were initially memory issues when running the algorithm on the full dataset. Therefore, a subset was created that consisted of users with at least 2,000 ratings. This subset has 1,212 users and 17,768 movies with at least one rating across all users and time periods. In the original paper, there was also an experiment for users with at least 2,000 ratings. Although the number of users was the same at 1,212, the number of movies was reduced to 5,264 in the paper. There is lack of information in the paper on the criteria used to obtain this reduction. From analysis of this subset, even excluding movies with less than 100 ratings would result in 7,349 movies. It is odd to reduce the available movies to less than one third of the original offering for collaborative filtering purposes as it significantly limits the potential recommendations. Therefore, this discrepancy between the two datasets remains as experiments detailed in the penultimate section will use the full subset with 17,768 movies.

Further analysis of this subset shows that the earlier periods are significantly more sparse compared to the latter periods as illustrated below in Figure 1. The datasets were stored in two formats, in a 3-dimensional dense array in *numpy* array format (.npy) and as a list of sparse matrices in in pickle file format (.pkl). Links to these files are available in the appendix.

4. Implementation

The model was implemented in Python. Table 1 summarises the packages used and the usage purpose.

Package name	Purpose
pandas	Reading, pre-processing of data and outputting results of experiments
sklearn.preprocessing	MultiLabelBinarizer was used to convert labels into multi-label binary vectors
numpy	Data pre-processing and generally the data container used throughout the algorithm. Built in functions for manipulating the data and performing mathematical operations
scipy.stats	Dirichlet module used for generating random samples
scipy.special	logsumexp used on log probability arrays gammaln for calculating log probability of NBD and Multinomial distribution digamma and polygamma functions used for the optimisation of parameter a under generalised Newton's method

Table 1 - Packages used in implementation

For the initial implementation, the dataset was stored in 3-dimensional *numpy* arrays of size (# users, # time periods, # movies). Computations were naturally vectorised using this type data representation. However, this caused memory issues when the dataset has more than 1,000 users. As the data sparsity was high, it made sense to use sparse data containers. Unfortunately, there was no readily available sparse data container library for arrays with more than 2 dimensions. Therefore, a list of sparse matrices was used instead. While this solved the memory issue, it changed the implementation into a loop over users which slowed down the execution speed. Both implementations are enclosed with the report and linked in the appendix.

The code was written on *Jupyter* notebooks and ran on *Google Colab*. As *Google Colab* had a GPU option, there was an attempt to re-write portions of the code using *cupy* which is a GPU enabled library built on top of *numpy*. However, there was negligible speed up as the dataset was not large enough for the GPU to make a meaningful difference and not all functions required were available in the *cupy* library.

Initialisation

For the initialisation of the parameters, all except the NBD parameters utilised a Dirichlet prior parameterised by a single value for convenience. In other words, all parameters of the Dirichlet distributions have the same value. This is a hyperparameter that will be tuned as part of the experiments in the next section. For the parameters of the NBD, the initialisation had a wide range for the initial experiments that were not a part of the final results. The range for the random initialisation was narrowed down for subsequent tests based on observations of the local optima obtained for these early runs.

E-Step

At the heart of the implementation is the EM algorithm which consist of the E-step and M-step. The E-step consists of calculating the posterior of the latent class for each user given the data. The most efficient way known to do this is the forward-backward algorithm. It starts by calculating the emission probabilities given the current set of parameters. All probabilities are calculated in log probability for numerical reasons.

$$P(I_u^t|Z_u^t = i) = P(N_u^t|a_i, b_i)P(I_u^t|N_u^t, \theta_i) \quad (5)$$

For $P(I_u^t|N_u^t,\theta_i)$, there was substantial computation cost savings by storing the calculations of the combinatorics portion i.e. $\frac{\log N_u^t!}{\sum \log I_u^t!}$ for each user which is the same in each iteration. All factorials were calculated using the gammaln function.

Next, we need the message passing functions α and β. The classic way of defining these functions are as follows.

$$\alpha_u^t(i) = P(I_u^{1:t}, Z_u^t = i|\Theta)$$

$$\beta_u^t(i) = P(I_u^{t+1:T}|Z_u^t = i, \Theta)$$

where Θ represents the full set of parameters of the model. These functions can then be combined to compute the required posteriors of the latent class.

$$P(Z_u^t = i|I_u^{1:T}) = \frac{\alpha_u^t(i)\beta_u^t(i)}{\sum_{i=1}^K \alpha_u^t(i)\beta_u^t(i)}$$

$$P(Z_u^t = i, Z_u^{t+1} = j|I_u^{1:T}) = \frac{\alpha_u^t(i)A_{ij}P(I_u^{t+1}|Z_u^{t+1} = j)\beta_u^{t+1}(j)}{\sum_{i=1}^K\sum_{j=1}^K \alpha_u^t(i)A_{ij}P(I_u^{t+1}|Z_u^{t+1} = j)\beta_u^{t+1}(j)}$$

However, using the above definitions and calculating log probabilities resulted into numerical stability issues. It was necessary to calculate the posterior as defined in the paper where normalising factors in the posterior were broken up and absorbed into the α and β functions.

$$\alpha_u^t(i) = P(Z_u^t = i|I_u^{1:t})$$

$$\beta_u^t(i) = \frac{P(I_u^{t+1:T}|Z_u^t = i, \Theta)}{P(I_u^{t+1:T}|I_u^{1:t})}$$

The posteriors are then computed as follows.

$$P(Z_u^t = i|I_u^{1:T}) = \alpha_u^t(i)\beta_u^t(i) = \gamma_u^t(i) \tag{6}$$

$$P(Z_u^t = i, Z_u^{t+1} = j|I_u^{1:T}) = \frac{\alpha_u^t(i)A_{ij}P(I_u^{t+1}|Z_u^{t+1} = j)\beta_u^{t+1}(j)}{P(I_u^{t+1}|I_u^{1:t})} = \xi_u^t(i) \tag{7}$$

The α and β functions are calculated recursively with the starting point defined as follows.

$$\alpha_u^1(i) = P(I_u^1|Z_u^1 = i)\theta_i$$

$$\beta_u^T(i) = 1$$

where $P(I_u^1|Z_u^1 = i)$ is obtained from equation (5).

M-Step

With the posteriors, we can optimise the parameters in the M-step. The vector π and each row of matrix A and θ all parameterise a multinomial distribution hence it was natural to use the Dirichlet distribution, which is the conjugate distribution, as the prior distribution. We can update these parameters using Maximum-a-posteriori (MAP) as follows where $\gamma_u^t(i)$ and $\xi_u^t(i)$ are defined in equations (6) and (7).

$$\pi_i^{new} = \frac{\sum_u \gamma_u^1(i) + n_i - 1}{\sum_u \sum_{i=1}^K \gamma_u^1(i) + \sum_{i=1}^K n_i - K} \quad \textit{for i=1,...,K} \tag{8}$$

$$A_{ij}^{new} = \frac{\sum_u \sum_{t=1}^T \xi_u^t(i,j) + n_i - 1}{\sum_u \sum_{t=1}^T \sum_{j=1}^K \xi_u^t(i,j) + \sum_{i=1}^K n_i - K} \quad \textit{for j=1,..,K for each i=1,..,K} \tag{9}$$

$$\theta_{ij}^{new} = \frac{\sum_u \sum_{t=1}^T \gamma_u^t(j) I_{ui}^t + n_i - 1}{\sum_u \sum_{t=1}^T \gamma_u^t(j) N_u^t + \sum_{i=1}^{|I|} n_i - K} \quad \textit{for i=1,...,|I| for each j=1,...,K} \tag{10}$$

For the first iteration, the MAP update could result in a negative number in the numerator when the Dirichlet prior parameter is less than 1. For this situation, the update will calculate the expected value instead of the mode i.e. remove the negative number in the numerator and denominator of the update.

There is no known conjugate distribution for the NBD hence Maximum Likelihood Estimation (MLE) is used instead. A generalised Newton method is used to optimise the parameter a (Minka 2002) iteratively as follows.

$$\frac{1}{a_i^{new}} = \frac{1}{a_i} + \frac{\frac{\sum_t E[\log \lambda_i^t]}{T} - \log \frac{\sum_t E[\lambda_i^t]}{T} + \log a_i - \Psi(a_i)}{a_i^2 \left[\frac{1}{a_i} - \Psi'(a_i)\right]} \tag{11}$$

where Ψ is the digamma function, $E[\lambda_i^t] = \frac{(N_i^t + a_i)b_i}{b_i+1}$ and $E[\log \lambda_i^t] = \Psi(N_i^t + a_i) + \log\left(\frac{b_i}{b_i+1}\right)$. Then, parameter b is updated as follows.

$$b_i^{new} = \frac{\sum_u \sum_{t=1}^{T} \gamma_u^t(i) N_u^t}{TUa^{new}} \tag{12}$$

When relating parameter b to the probability of success in a typical NBD formulation, it becomes clearer that it is simply updating the probability p as a ratio of number of successes divided by total number of trials i.e. $p = \frac{b}{b+1} = \frac{\overline{N_\iota}}{\overline{N_\iota}+a}$ where $\overline{N_\iota}$ is the average number of movies rated in a time period.

In the implementation which had a loop over each user, it required a variable to store the per user state probabilities. For the updating of parameter a and b, this presented an issue since the values required in the numerator of equation (11), specifically $E[\log \lambda_i^t]$, would have to be recalculated for each user for every iteration. This was computationally expensive and it was found not to have a material impact if instead the values for $E[\lambda_i^t]$ and $E[\log \lambda_i^t]$ were kept constant in each iteration of the generalised Newton method.

Check for Convergence

The convergence check is performed by calculating the expected log likelihood to see if it has changed by more than a threshold value. The expected log likelihood is a lower bound for the log likelihood of the data given the parameters. This lower bound will equal the log likelihood of the data when the expectation is taken under the posterior of the latent variable. Therefore, it is calculated after one final E-step to update the posterior. The expected log likelihood of the HMM has three components which are the initial state log likelihood, transition log likelihood and the emission log likelihood as shown in equation (13).

$$\begin{aligned} P(X|\Theta) = \quad & \sum_u \sum_k P(Z_u^1 = k | I^{1:T}, \Theta^{new}) \log \pi_k \\ & + \sum_u \sum_{t=1}^{T-1} \sum_j \sum_k P(Z_u^t = j, Z_u^{t+1} = k | I^{1:T}, \Theta^{new}) \log A_{jk} \\ & + \sum_u \sum_{t=1}^{T} \sum_k P(Z_u^t = k | I^{1:T}, \Theta^{new}) \log P(N_u^t, I_u^t | a_k, b_k, \theta_k) \end{aligned} \tag{13}$$

Prediction

Finally, prediction can be done with the parameters for one time period after the training period. This is done as described in section 2 where the ranking of each movie in terms of likelihood of appearing in the period is calculated according to the negative component of equation (4) i.e. $R(i,u) = -\sum_{k=1}^{K} P(Z_u^{T+1} = k)\ (1 + b_k\theta_{ki})^{-a_k}$. Then, movies in the list which have appeared in the user's history are removed. Finally, the top N movies in the list can be used as the recommendation list.

5. Experiment Results

From the original paper, it was clear that the algorithm performed better when the data had higher density. At the same time, due to time constraints, it was not possible to perform the experiments on larger datasets especially since the implementation using loops over users was around 40-50% slower. For these reasons, all the experiments were done on the subset of the data with users that had at least 2,000 ratings. This subset has 1,212 users over 73 time periods.

There were four objectives for the experiments. The first was to compare the performance of the model as shown in the original paper. Note that the discrepancy in the data, namely the larger number of movies, meant that a direct comparison was not possible since it should result in worse prediction performance. However, it would still provide a guidance on the feasibility of the results as seen in the paper.

The second was to investigate the effect of adjusting the Dirichlet prior parameter. In the original specifications, the authors used $\frac{100}{K}$ for all the Dirichlet prior parameters. As K was tested in the range from 5 to 100, this results in probabilities that are fairly even. However, since the model implicitly expects users to fall into different categories combined with the fact that the EM algorithm only guarantees a local maximum, it makes sense to try to induce sparsity in the latent class memberships by specifying a Dirichlet prior parameter less than 1. This results in initial distributions with higher probability mass in one of the latent classes, transitional states or movies. The different setups should result different local maxima with the EM algorithm.

The third objective was to study whether truncating the earlier periods of the data which had lower density would hurt or improve the performance. Finally, the last objective was to explore whether setting a threshold on the movie ratings would hurt or improve performance. Since a low rating would indicate a dislike, it is reasonable to assume that if only movies rated 3 to 5 were considered as positive data points, the true user preferences would be better captured by the model.

The convergence criteria for all runs are set with an epsilon value of 0.01% i.e. if the percentage change in expected log likelihood is less than epsilon then the EM algorithm is terminated. As per the original paper, the evaluation will be done by calculating the precision and recall of the top 5 and top 10 recommended movies. The precision is defined as the percentage of the recommended movies that are a positive match in the actual data. The recall is the percentage of movies in the actual data that are recommended.

Initially, the experiments were run with the last period being held out as the test set. There were 60,499 ratings by users in this test period. The training period utilised all 72 prior periods. This set up resulted in inferior performance compared to that seen in the original paper. This was expected as the number of movies in the corpus is 3 times higher than the dataset used in the paper. Figure 2 summarises the results of executing 10 runs of each permutation of hypermeters K from 5 to 25 in steps of 5 and Dirichlet parameter using both $100/K$ and 0.9. The lowest AIC score was achieved with the model using 25 latent classes with 0.9 as the Dirichlet prior parameter. However, it does not necessarily translate to a better precision or recall as that particular run results in precision of 5.7% and 5.0% and recall of 0.57% and 1.0% for the top 5 and top 10 respectively.

It is clear from Figure 2 that using a Dirichlet prior parameter of 0.9 results in better performances especially for a larger number of latent classes. This was further confirmed by analysing the log likelihood achieved by the 2 sets of parameters as shown in Figure 3. The use of 0.9 as the parameter value consistently results in convergence to a better local maximum in terms of log likelihood.

Additional tests were done using different holdout periods but with 5 runs for each permutation. Figure 4 shows the results of using the penultimate period for prediction and using all 71 prior periods for training. Since it had been established that using Dirichlet prior parameter of 0.9 was superior, all permutations only used this value. In this prediction period, there were 65,884 movies rated which was slightly more than the last period. This should result in higher precision but lower recall. However, both precision and recall were better and significantly so for precision. A similar result was seen using the third last period for prediction. There was a total of 99,038 movie ratings in this period which is significantly higher. The precision was indeed higher but the recall was surprisingly comparable to using the last period for prediction as shown in Figure 5. Possible explanations include users exhibiting greater than expected changes in the last period or the EM algorithm is not able to converge close enough to the optimal solution.

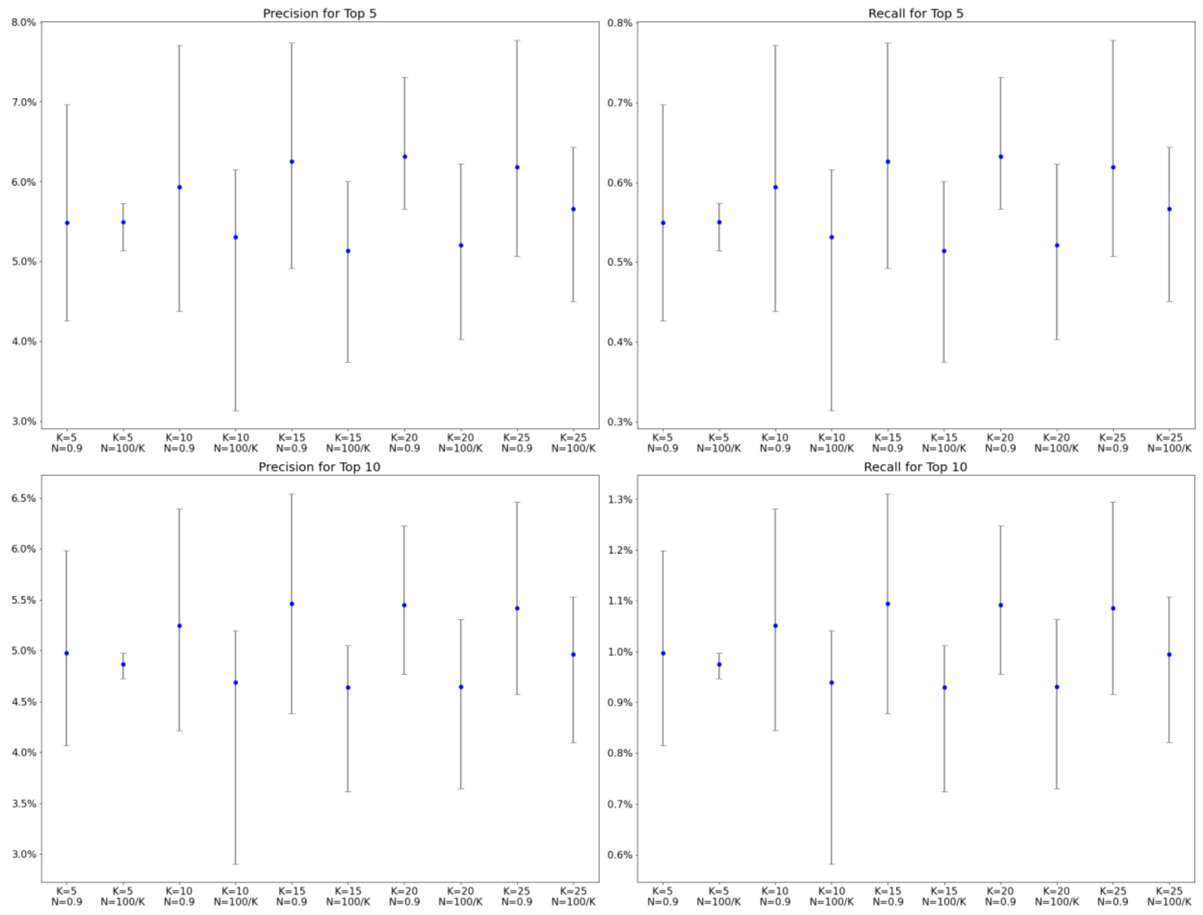

Figure 2 - Results showing precision and recall of Top 5 and Top 10 movies to recommend for 10 permutations of hyperparameters. The blue dot is the average of the 10 runs while the top and bottom of the grey lines show the maximum and minimum seen in the 10 runs.

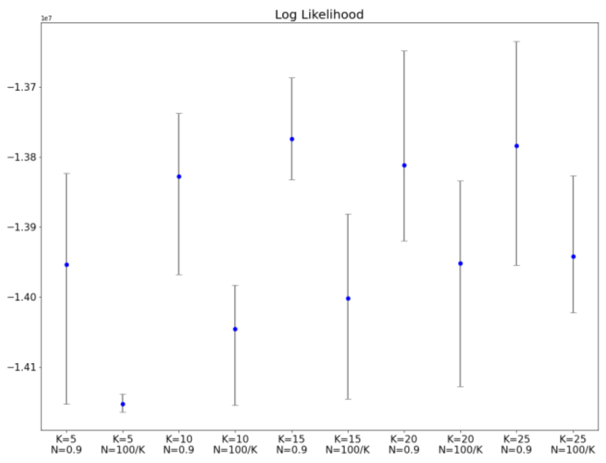

Figure 3 - Log likelihood of the same permutations shown in Figure 2

For the third objective, tests were done using only the prior 36 periods for training as the data density was significantly higher for these periods as seen in Figure 1. As the algorithm only filters out movies rated during the training periods, it should have a negative impact on the result. Despite this negative impact, the results were still better than using all available training periods. The results are shown in Figure 6. Similar tests were done using the penultimate and third last period for prediction as shown in the appendix and the results were either slightly better or comparable to using all available periods for training.

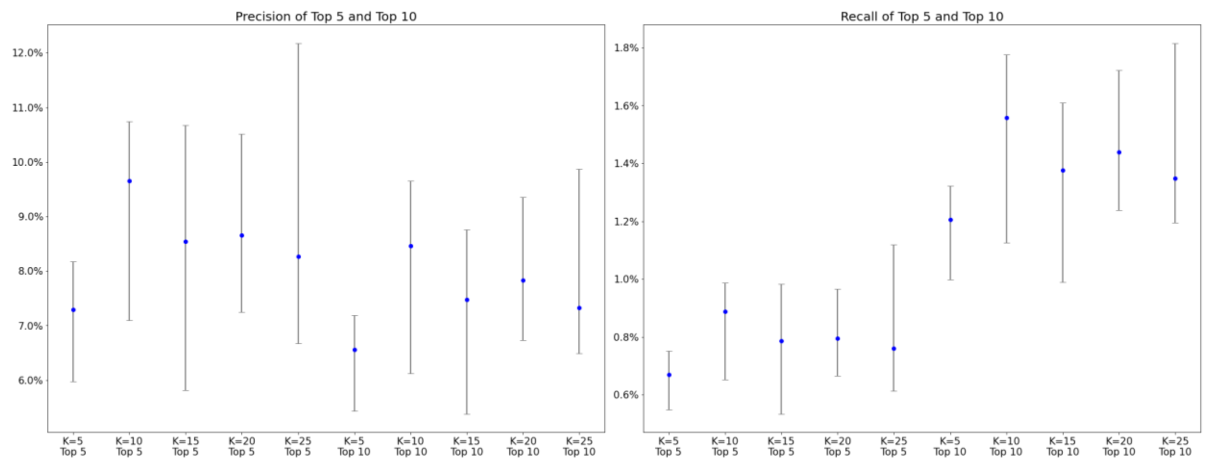

Figure 4 – Results using the penultimate period for prediction period and all prior periods for training with Dirichlet prior parameter value of 0.9

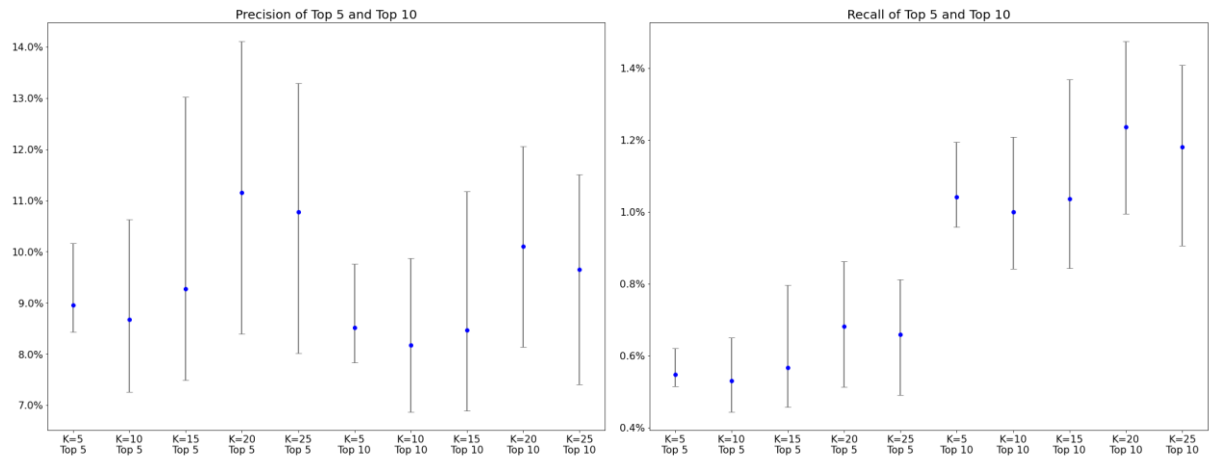

Figure 5 – Results using third last period for prediction and all prior periods for training with Dirichlet prior parameter value of 0.9

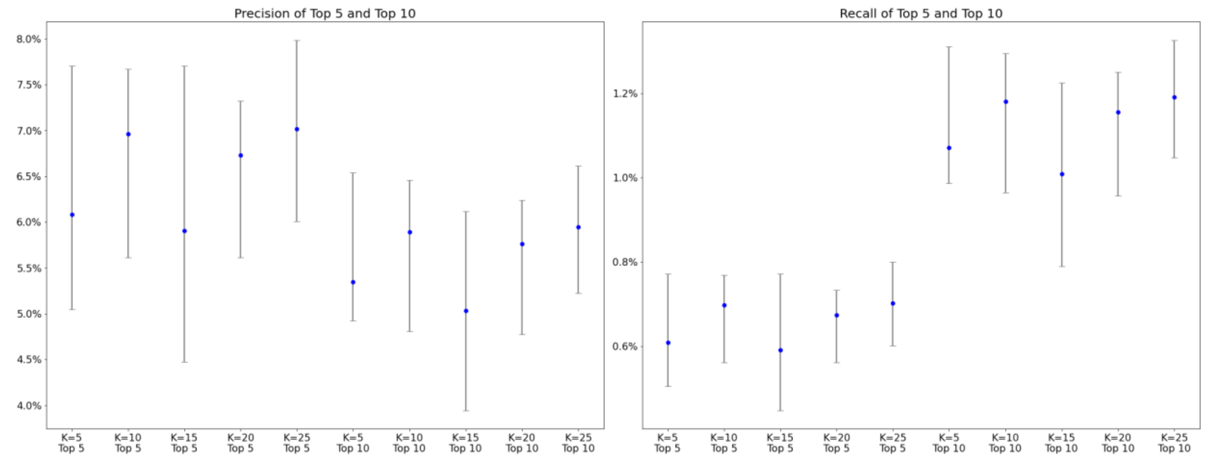

Figure 6 – Results using the last period for prediction period and prior 36 periods for training with Dirichlet prior parameter value of 0.9

Finally, setting a threshold on the movie rating to be 3 or higher was expected to produce a better result. Unfortunately, experiments proved otherwise as the prediction scores were clearly inferior as shown in Figure 7. A possible explanation is that the evaluation is predicting whether a movie would be rated instead of the likelihood of a high rating. These can be distinct as we all have chosen movies we ended up disliking based on a valid selection criteria.

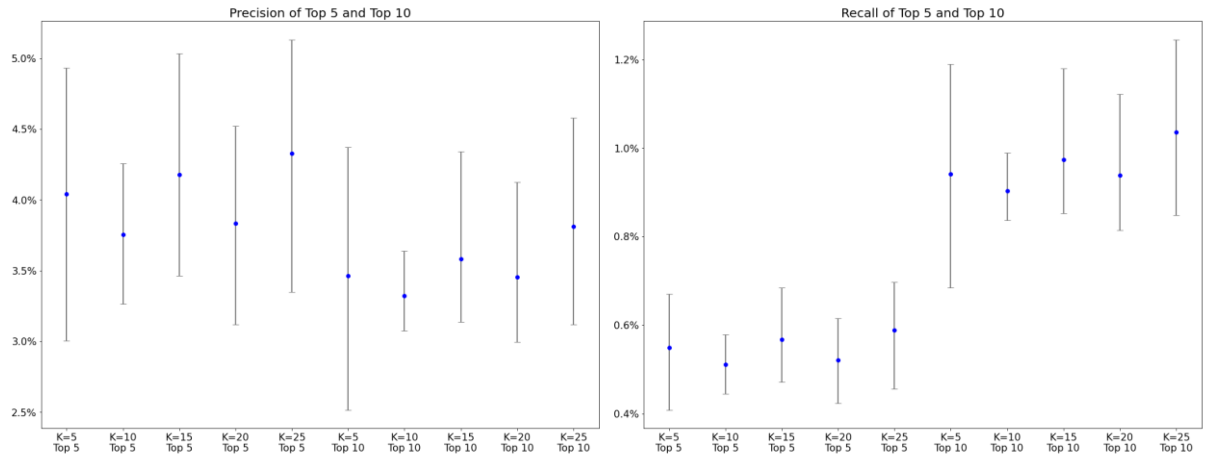

Figure 7 - Results using the last period for prediction period and all prior periods for training with Dirichlet prior parameter value of 0.9 and rating threshold of 3 or higher

6. Conclusion

The project was to replicate the Hidden Markov Model for Collaborative Filtering. The implementation revealed several important numerical considerations. The α and β functions in the forward-backward algorithm had to be carefully defined for numerical stability. The experimentation with the Dirichlet prior parameter also required a modification of the MAP formula to account for potential negative values that arise due to the parameter starting with a value less than 1.

As the EM algorithm only guarantees convergence to a local maximum, the initialisation procedure is an important ingredient to obtain a solution close to the global optimal. A key part of the initialisation procedure involved the Dirichlet prior parameter which was set to $\frac{100}{K}$ in the original paper. No explanation was given for this choice which would likely result in fairly uniform values especially when K is small. The natural alternative to try would be a value less than 1. Intuitively, this induces sparsity in the latent class memberships and should work well if users had more distinct characteristics. Indeed, the experiments showed that this choice for the Dirichlet prior parameter results in consistently better solutions both in terms of log likelihood and prediction scores. For similar settings, one should experiment with both choices as the outcome is likely dependent on the data.

Generating the recommended list of movies raised an important issue not addressed in the paper. In certain context, such as recommending movies, it does not make sense to include movies that appear in the user's history. However, in a music recommendation context, users might often return to the same songs and artist and hence including historical items is logical. Since the experiments done in this report are on the Netflix dataset, it was crucial to remove movies that are in the user's history from the recommendation list.

Finally, an important aspect of the implicit rating approach is that it effectively throws away potentially important information that helps to characterise a user's preference. The simple modification of setting a rating threshold clearly did not improve the performance. However, this does not mean that there is no useful information in the explicit ratings that can improve prediction performance. The current model only admits binary entries for rated and unrated. A potential extension would be to add a categorical distribution that models the explicit rating given a latent class. The issue with this approach is that it would increase the number of parameters by $|I| \times R \times K$ where R is the range of ratings allowed. Incorporating explicit ratings that would improve performance in a computationally feasible manner would be the next step to explore as an extension to the model.

References

Bishop, C. M. 2006. “Pattern recognition and machine learning”, Springer.
Minka, T. P. 2002. “Estimating a Gamma Distribution,” unpublished paper, (http://research.microsoft.com/en-us/um/people/minka/papers/minka-gamma.pdf).
Sahoo, N., Singh, P. V., & Mukhopadhyay, T. 2012. “A hidden Markov model for collaborative filtering.” *MIS quarterly*, 1329-1356.
Tsun, A. 2020. “Probability & Statistics with Applications to Computing”, self-published. (https://www.alextsun.com/files/Prob_Stat_for_CS_Book.pdf)

Appendix

Links to dataset with users having at least 1,000 movie ratings

3D dense array of shape (# users, # periods, # movies): Link

List of sparse matrices of length (# users) and matrices of shape (# periods, # movies): Link

Dataset with rating threshold of 3 or higher in sparse format: Link

Links to implementations in Jupyter notebook format

Implementation based on 3D dense arrays: Link

Implementation based on list of sparse arrays: Link

Additional Experiment Results

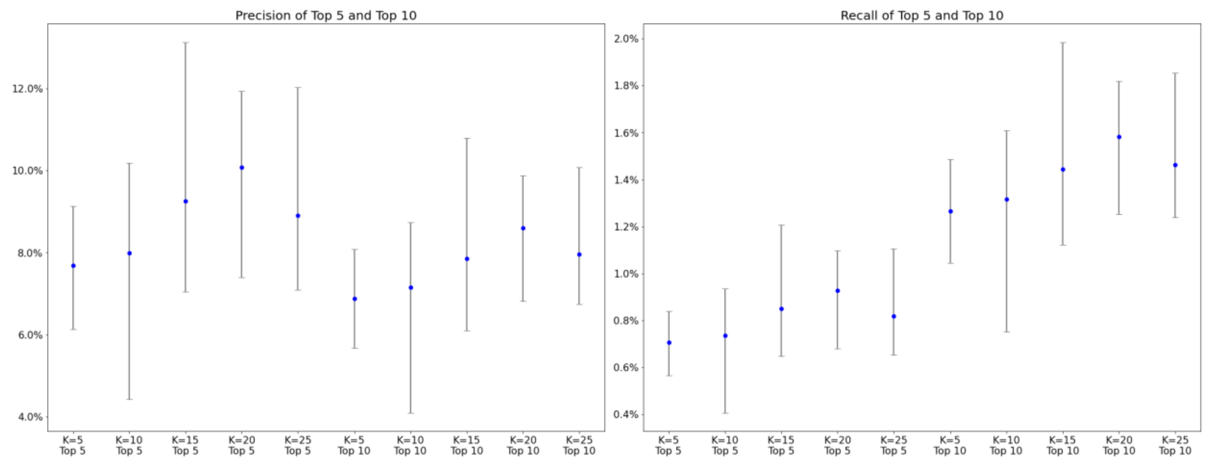

Figure 8 - Penultimate period for prediction and prior 36 periods for training with Dirichlet prior parameter value of 0.9

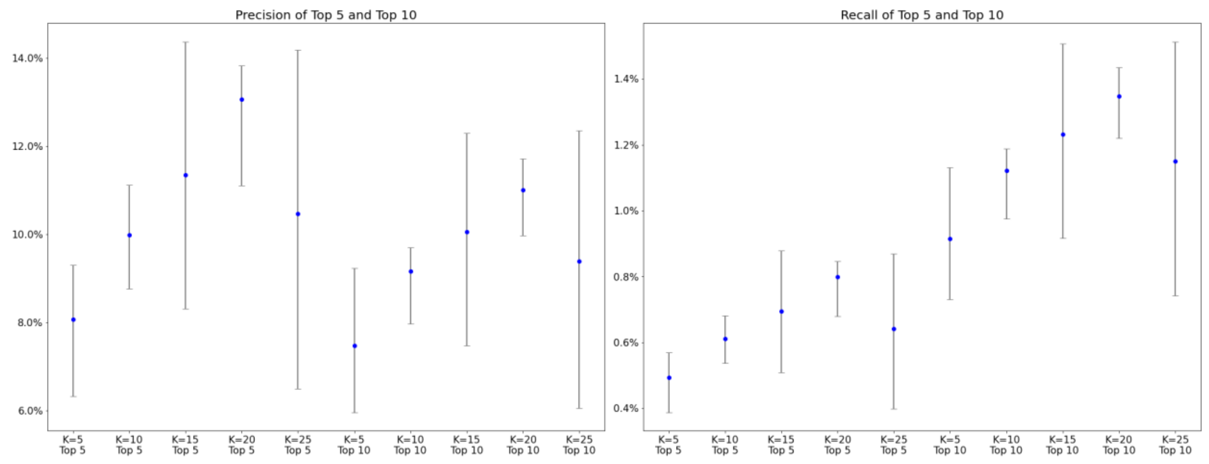

Figure 9 - Third last period for prediction and prior 36 periods for training with Dirichlet prior parameter value of 0.9