Online Recommender Systems

Saurabh Mehra
Department of Industrial and Systems Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060
samehra@vt.edu

Shalini Ragothaman
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060
rshalini@vt.edu

Subhash Holla Hosakoppa Sukumar
Department of Industrial and Systems Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060
subbu23@vt.edu

Abstract

Recommendation Systems are ubiquitous and are riddled with challenges such as the cold-start. We evaluate two models designed for Recommendation Systems - one of which is based on a Reinforcement Learner namely Online Particle Thompson Sampler, and the other employs a Deep Neural Network architecture, Deep-ICF, to provide recommendations to users. A detailed study of the two models is conducted on several real-world datasets on various evaluation metrics and it is observed that the Deep-ICF model performs better on a cold start scenario.

1 Introduction

Recommender Systems play a vital role for most E-commerce platforms to recommend items to users based on their interests by leveraging the power of data and predicting the rating a user would give to these items. Based on the predictions made for the ratings of the items that would be recommended to users, E-commerce companies ensure that the users are actively engaged with their platforms by recommending the right products, movies, music, news articles based on users' interests while simultaneously boosting company's revenue.

Recommendation is a problem that has been extensively studied by the research community. Among a variety of algorithms that are being devised that meet the need for recommender systems, a class of algorithms are called collaborative filters.

Collaborative Filtering: These systems provide recommendations to users based on a similarity measure between the user and the item. These systems make suggestions based on the historic users' preferences on items. In this method of recommendation, the system either finds users with similar interests, analyzes their behavior and then recommends the current or new user the same items; or the system looks at items that are similar to the ones which the user has bought or rated well earlier and the system recommends those products or items to them. Collaborative Filtering, thus, has two types of recommendation systems:

User Based Collaborative Filtering (UCF):

A user-item rating matrix is created that consists of the preferences of the users with respect to the items that are being suggested to them. The UCF recommends items by finding similar users to the active user to which items are being recommended to. At every step, the user-item matrix is filled to consist of the predictions of ratings of items that a new user has not seen, based on the similarity of the user.

Item Based Collaborative Filtering (ICF):

Item-based CF represents a user with their historically interacted items, using the similarity between the target item and interacted items to estimate the user-item relevance. ICF is more computationally expensive when building the item-item similarity matrix and these calculations are computed offline; however the online prediction requires very little computational cost. It is also seen that the composability of ICF for user preference modeling makes it sufficiently easier to implement online personalization **[8].**

Recommender systems are devised as offline or online learning systems - offline evaluation is done by recording the items that the users have interacted with, hiding a few of these user-item interactions (test set) and training algorithms on the remaining information to assess the accuracy of the recommender system.

Click-Through Rate (CTR) indicates the ratio between the interactions on recommended items and the total of recommended items produced to the users proves a valuable measure of the system. Another metric of evaluation for the system is given by *Rewards* which are the observed ratings obtained from the users for items recommended that needs to be maximized to provide better recommendations.

An issue that is prevalent when accurate recommendations cannot be provided for new users or items with little or no information is the **Cold Start Problem**. This is particularly present in collaborative filtering systems that rely on user-item interactions.

The report is organized as follows: **Section 2** describes the Literature survey that was undertaken, **Section 3** details the Methodology explaining the two models selected for experimentation and further study, **Section 4** outlines the results and inferences obtained during the running and the creation of the Recommender System.

2 Literature Review

The paper **A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking [1]** reviews the optimal Bayesian algorithms for nonlinear and non-Gaussian tracking problems, with a large focus on Particle Filters. The paper describes Particle Filters as the sequential Monte Carlo methods that are based on a point mass or a particle which represents the probability densities that can be applied to any state-space model. The particle filter helps in generalizing the Kalman filter that assumes a Gaussian posterior density at every time step. The authors describe the Sequential Importance Sampling Algorithm which is a Monte Carlo method that forms the basis for most MC filters designed.

Similar concepts of Particle filtering and Monte Carlo methods were reviewed in the paper **A Tutorial on Particle Filtering and Smoothing: Fifteen years later [4]** that delineates the basic and advanced particle filtering methods developed in the literature, that can be interpreted as special instances of the generic Sequential Monte Carlo algorithm, such as the auxiliary particle filters, particle filtering with MCMC moves, block sampling strategies and the Rao-Blackwellised particle filters. The authors detail how a numerical approximation of the distribution:

$$p(x_{1:n}|y_{1:n})_{n1}$$

can be computed sequentially in time and a direct application of the SMC methods described to a set of target distributions that results in a popular class of particle filters.

The combined use of the Rao-Blackwellised particle filtering with Thompson Sampling in the use of Matrix Factorization Recommendation was analysed in the paper **Efficient Thompson Sampling for Online Matrix-Factorization Recommendation [7]**. A computationally efficient algorithm was extensively studied in a Matrix Factorization Recommendation Bandit that recommends items for users and maximizes the accumulated rewards based on the user and the observed ratings. The

algorithm designed allows to simultaneously update the posterior probabilities of U and V - which are the factors of the matrix factorization M=UV$\hat{T}$ while also minimizing the cumulative regret in the online setting.

Upon studying further on item based and user based collaborative filtering, the paper **Deep Item-based Collaborative Filtering for Top-N Recommendation [8]** describes the numerous advantages of ICF over UCF as the ICF is able to encode more signal in its input that the UCF signal that only used an ID to represent the user. This is found to provide more potential by the ICF to improve both the accuracy and the interpretablity than the UCF models for the top-N recommendations. The paper discusses a predictive model, for a given user-item pair (u, i), a method by which the prediction value ŷ_ui is estimated is described. A general framework for the higher-order item relation modeling with neural networks is presented in the paper along with discussing the two instantiations under the framework — DeepICF that uses standard average pooling on second-order interactions and DeepICF+a that uses an adaptive pooling strategy with attention on second-order interactions.

3 Methodology

The goal of the project was to evaluate models that recommend movies, articles,ads, etc.(items) with a feedback from the customer through a click or rating for the same. To this regard we decided to evaluate two models: The Online Particle Thompson Sampler [7], is a user based matrix factorizer where the user information, especially their history of interaction with other items, is used to give recommendations. The DeepICF [8] model, is an item based collaborative filter which is claimed to perform better, especially in an online setting and supposedly is more efficient with the "cold start" problem.

3.1 Online Particle Thompson Sampler

In a typical online recommendation system, users and observed ratings (also called rewards) arrive over time, and the task of the system is to recommend item for each user so as to maximize the accumulated expected rewards. So the system is required to learn over time what are the best items to recommend based on the existing user ratings as well sufficiently explore all the items. recommended. The quality of the recommendation system is measured by its expected cumulative regret:

$$CR = E\left[\sum_{t=1}^{n}[r_t^o - max_j U_{it}^{*T} V_j^*]\right] \tag{1}$$

where U^* and V^* are user and product latent features, r_o^t is the observed user rating. An efficient Rao-Blackwellized particle filter (RBPF) is designed to exploit the specific structure of the probabilistic matrix factorization model. Let $\theta = (\sigma, \alpha, \beta)$ be the control parameters and let posterior at time t be $p_t = Pr(U, V, \sigma_u, \sigma_v | r_{1:t}^o, \theta)$. The RBPF algorithm maintains the posterior distribution p_t as follows. Each of the particle conceptually represents a point-mass at V, σ_u (U and σ_v are integrated out). Thus, $p_t(V, \sigma_u)$ is approximated by $p_t = \frac{1}{D}\sum_{d=1}^{D} \delta_(V_{(d)}, \sigma_U^{(d)})$ where D is the number of particles [7]. Considering U and σ_v as auxiliary variables, effectively sample $U, \sigma_v | V, \sigma_u \sim p_t(U, \sigma_v | V, \sigma_u)$ and then sample $V^{'}, \sigma_u^{'} | U, \sigma_v \sim p_t(V^{'}, \sigma_u^{'} | U, \sigma_v)$. However, this move would be highly inefficient due to the number of variables that need to be sampled at each update. Therefore, the key to an efficient implementation is that the latent features for all users except the current user U_{-it} are independent of the current observed rating $r_t^o : p_t(U_{-it}|V, \sigma_u) = p_{t-1}(U_{-it}|V, \sigma_u)$, therefore at time t only sample U_{it} as there is no need to resample U_{-it}. Furthermore, it suffices to resample the latent feature of the current item V_{jt}.

Below are the expressions for the conditional probabilities of interest. Supposed that V and U are known. Then the vectors U_i are independent for each user i. Let $rts(i) = \{j | r_{ij} \in R^o\}$ be the set of items rated by user i, observe that the ratings $\{R_{ij}^o | j \in rts(i)\}$ are generated i.i.d. from U_i following a simple conditional linear Gaussian model. Thus, the posterior of U_i has the closed form

$$Pr(U_i|V, R^o, \sigma, \sigma_U) = Pr(U_i|V_{rts(i)},\ R_{i,rts(i)}^o, \sigma_U, \sigma) = \mathcal{N}(U_i|\mu_i^u,\ (\Lambda_i^u)^{-1}) \tag{2}$$

$$where\ \ \mu_i^u = \frac{1}{\sigma^2}(\Lambda_i^u)^{-1}\zeta_i^u;\ \ \Lambda_i^u = \frac{1}{\sigma^2}\sum_{j \in rts(i)} V_j V_j^T + \frac{1}{\sigma_u^2} I_K;\ \ \zeta_i^u = \sum_{j \in rts(i)} r_{ij}^o V_j \tag{3}$$

Algorithm 1 Algorithm 1 Particle Thompson Sampling for Matrix Factorization (PTS)
Global control params: $\sigma, \sigma_U, \sigma_V$; for Bayesian version (PTS-B)σ, α, β

$\hat{p}_0 \leftarrow \text{InitializeParticles}()$
$R^o = \emptyset$
for $t = 1, 2 \ldots$ **do**
  $i \leftarrow$ current user
  Sample $d \sim \hat{p}_{t-1}.w$
  $\leftarrow \hat{p}_{t-1}.V^{(d)}$
  [If PTS-B] $\sigma_U \leftarrow \hat{p}_{t-1}.\sigma_U^{(d)}$
  Sample $\tilde{U}_i \sim \Pr(U_i|\tilde{V}\ , \sigma_{U,\sigma,r^o_{1:t-1}})$ ▷ sample new U_i due to Rao-Blackwellization
  $\hat{j} \leftarrow \arg\max_j \tilde{U}_{ij}$
  Recommend $\hat{j}$ for user i and observe rating r.
  $r_t^o \leftarrow (i, \hat{j},\ r)$
  $\wedge t \leftarrow \text{UpdatePosterior}(\hat{p}_{t-1},\ r^o_{1:t})$
procedure UPDATEPOSTERIOR($\hat{p},\ r^o_{1:t}$)
    ▷ $\hat{p}$ has the structure (w, particles) where particles $[d] = (U^{(d)},\ V^{(d)},\ \sigma_U^{(d)},\ \sigma_V^{(d)})$.
  $(i,\ j,\ r) \leftarrow r_t^o$
  $\forall d, \Lambda_i^{u(d)} \leftarrow \Lambda_i^u(V^{(d)},\ r^o_{1:t-1})\ , \zeta_i^{u(d)} \leftarrow \zeta_i^u(V^{(d)},\ r^o_{1:t-1})$ ▷ see Eq. (3)
  $\forall d, w_d \propto \Pr(R_{ij} = r|V^{(d)},\ \sigma_U^{(d)},\ \sigma,\ r^o_{1:t-1}), \sum w_d = 1$ ▷ Reweighting; see Eq.(5)
  $\forall d, i \sim \hat{p}.w; \hat{p}'.\text{particles}\ [d] \leftarrow \hat{p}.\text{particles}\ [i]; \forall d, \hat{p}'.w_d \leftarrow \frac{1}{D}$ ▷ Resampling
  for all d **do** ▷ Move
    $\Lambda_i^{u(d)} \leftarrow \Lambda_i^{u(d)} + \frac{1}{\sigma^2} V_j V_j^{\neg -}; \zeta_i^{u(d)} \leftarrow \zeta_i^{u(d)} + rV_j$
    $\wedge \prime.U_i^{(d)} \sim \Pr(U_i|\hat{p}'.V^{(d)}, \hat{p}'.\sigma_U^{(d)},\ \sigma,\ r^o_{1:t})$▷ see Eq. (2)
    [If PTS-B] Update the norm of $\hat{p}'.U^{(d)}$
    $\Lambda_j^{v(d)} \leftarrow \Lambda_j^v(V^{(d)},\ r^o_{1:t})\ , \zeta_j^{v(d)} \leftarrow \zeta_i^u(V^{(d)},\ r^o_{1:t})$
    $\wedge \prime.V_j^{(d)} \sim \Pr(V_j|\hat{p}'.U^{(d)}, \hat{p}'.\sigma_V^{(d)},\ \sigma,\ r^o_{1:t})$
    [If PTS-B] $\hat{p}'.\sigma_U^{(d)} \sim \Pr(\sigma_U|\hat{p}'.U^{(d)},\ \alpha,\ \beta)$▷ see Eq.(4)
  end for
  return $\hat{p}'$
end procedure

The conditional posterior of $V, Pr(V|U,\ R^o, \sigma_V, \sigma)$ is similarly factorized into $j = 1\mathcal{N}(V_j|\mu_j^v,\ (\Lambda_j^v)^{-1})$ where the mean and precision are similarly defined. The posterior of the precision $\lambda_U = \sigma_U^{-2}$ given U (and simiarly for λ_V) is obtained from the conjugacy of the Gamma prior and the isotropic Gaussian

$$Pr(\lambda_U|U, \alpha, \beta) = \Gamma(\lambda_U|\frac{NK}{2} + \alpha, \frac{1}{2}\|U\|_F^2 + \beta) \tag{4}$$

Although not required for Bayesian PMF, we give the likelihood expression

$$Pr(R_{ij} = r|V,\ R^o, \sigma_U, \sigma) = \mathcal{N}(r|V_j^T \mu_i^u, \frac{1}{\sigma^2} + V_j^T \Lambda_{U,i} V_j) \tag{5}$$

The advantage of the Bayesian approach is that uncertainty of the estimate of U and V are available which is crucial for exploration in a bandit setting. However, the bandit setting requires maintaining online estimates of the posterior as the ratings arrive over time. Using the above closed form updates an efficient Rao-Blackwellized particle filter is designed for maintaining the posterior over time.

3.2 DeepICF

While the online particle thompson sampling method uses matrix factorization which represents a user and a projection of the user in the same embedding space; then the relevance score between

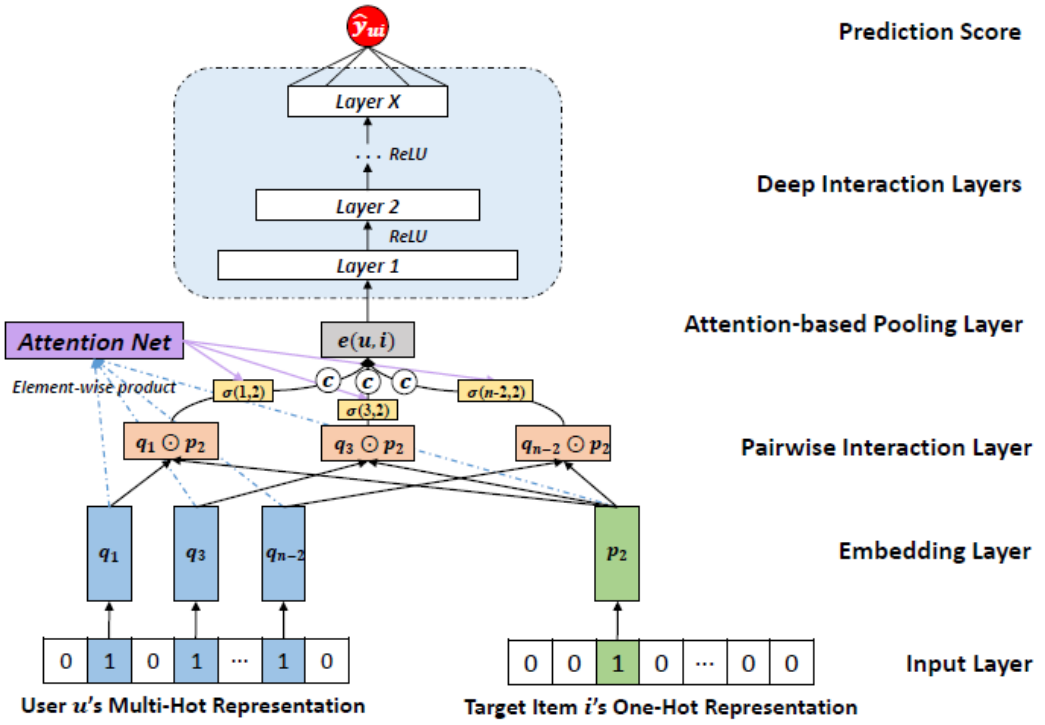


Figure 1: DeepICF Architecture

a user-item pair is estimated as the inner product of the user embedding and item embedding. In contrast, item-based collaborative filtering represents a user with their historically interacted items, using the similarity between the target item and interacted items to estimate the user-item relevance.

DeepICF is one such implementation of the item-based collaborative filtering. This uses a deep neural network architecture (see Figure 1) similar to the neural collaborative filering (NCF) [6] with 2 major differences. First, in the input layer that represents a user (bottom left), distinct from NCF that applies one-hot encoding on the user's ID, multi-hot encoding is used on the user's interacted items. This naturally leads to the difference in the embedding layer — rather than using a vector to represent the user, a set of vectors is used where each vector represents an interacted item of the user. Second, instead of designing a holistic NCF layers to model the interaction between user and item,the interaction modeling is divided into two components — 1) pairwise interaction layer that models the interaction between each historical item and the target item, and 2) deep interaction layers that model higher-order interaction among all historical items and the target item.

The vanilla DeepICF suffers from the lack of uniform optimal setting for the normalization hyper-parameter (α) used in the pooling layers. Hence attention based pooling is used in the DeepICF+a network inspired by the attention network design in ACF [2] and NAIS[5]. Figure 2 illustrates the attention network. The pooling is defined as follows,

$$f_{att}(V_{ui}) = \frac{1}{|V_{ui}|^{\alpha}} \sum_{v \in V_{ui}} a(v).v = \frac{1}{(|R_u^+| - 1)^{\alpha}} \sum_{j \in R_u^+ \setminus i} a(q_j \odot p_i).(q_j \odot p_i)$$

where a(v) is the attention function that takes vector v as input, and outputs the importance of v in the weighted average pooling.

Upon reproducing the work of [8], using the code that they had provided on Github we evaluated the model on the same metrics that were used by the authors.

- **Hit Ratio (HR)**: This is a common metric used in top-K recommender systems where if a test item appears in the recommended list, it is deemed a hit. HR is calculated as:

$$HR@K = \frac{Numberof Hits@K}{|GroundTruth|}$$

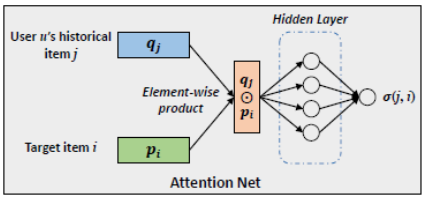


Figure 2: Attention Network

- **Normalized Discounted Cumulative Gain (NDCG)**: This is used to evaluate the accuracy of the recommender by assigning higher importance to results at top ranks, scoring successively lower ranks with marginal fractional utility.

$$NDCG@K = Z_K \sum_{i=1}^{K} \frac{2^{r_i} - 1}{log_2(i+1)}$$

where Z_K is the normalizer to ensure the perfect ranking has a value of 1; r_i is the graded relevance of item at position i. We use: r_i = 1 if the item is in the test set, and 0 otherwise.

4 Results

4.1 Dataset

The datasets that we used were: the Movie Lens dataset which had ratings with respect to a specific user, movie and a given timestamp; the pinterest dataset which had a click through indicated by a 1 for a given item ID. Table 1 gives the brief summary statistics for the datasets used for both models.

Dataset	#Interactions	#Users	#Items	Density
MovieLens 1 Million	1,000,209	3,706	6,040	4.47%
MovieLens 100 Thousand	100,072	943	1682	6.3%
Pinterest	1,500,809	9,916	55,187	0.27%

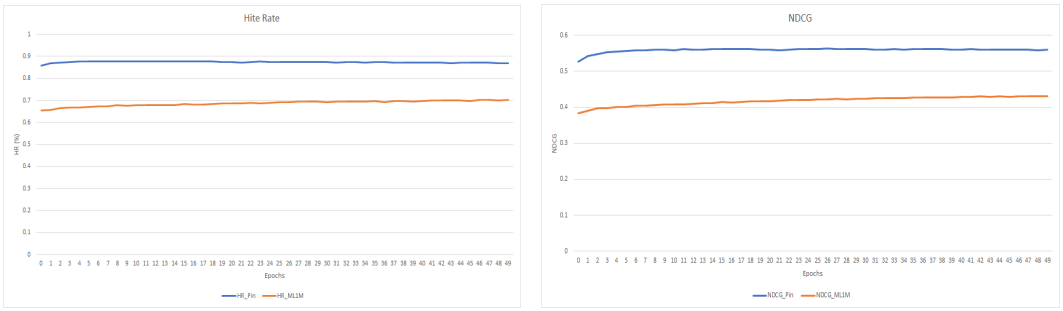
Table 1: Data summary statistics

4.2 DeepICF

Upon running the DeepICF model on both the datasets for 50 epochs we observe the hit rate and NDCG as shown in Figure 3. This along with the train-error rate (see Figure 4.) indicate that the model performed better in the pinterest dataset as apposed to the movielens dataset. This is because the pinterest dataset has more features and a higher user to feature ratio as compared to that of movie lens. This also helps us conclude that the density of the data has no effect on the performance of this item-based collaborative filtering method. This could mean that no matter how spare the data is and how many data values are missing. If we have a high item-user ratio we can expect good model performance.

4.3 Online Particle Thompson Sampler

(K, P)	Mean	Std Dev	Min	Max	Avg Cum Reg
(2, 2)	0.84	0.03	0.80	0.88	12734.54
(2, 5)	1.22	0.03	1.16	1.25	18660.34
(3, 5)	1.89	0.03	1.86	1.93	29558.86
(3, 10)	2.68	0.99	2.16	4.65	27861.91
(3,20)	2.47	0.06	2.37	2.52	37447.68
(5,10)	3.94	0.04	3.89	4.00	55679.6
(5,20)	4.28	0.05	4.20	4.34	59623.57



(a) Hit Rate (b) NDCG

Figure 3: Movie-Lens 1M data

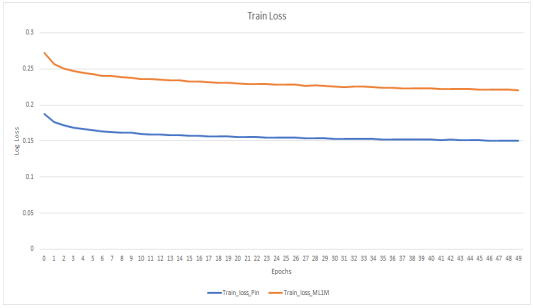


Figure 4: Train Loss Movie-Lens 1M data

For the Online Particle Thompson Sampler we used the Movielens 100k dataset for evaluating various configurations of latent features (K) and number of particles (P) which are two common hyperparameters. To empirically test how the number of latent features vs particle numbers effects performance, we run each configuration five times. We found the number of latent features to be the most significant hyperparameter for model performance as show in the table: increasing K lead to poorer performance. The number of particles is harder to explain. Somewhat counter intuitively we found that a lower number of particles leads to better results. The reason behind this may be that for this dataset most users reward preference can be approximated by 2-5 particles but more work in this area is needed.

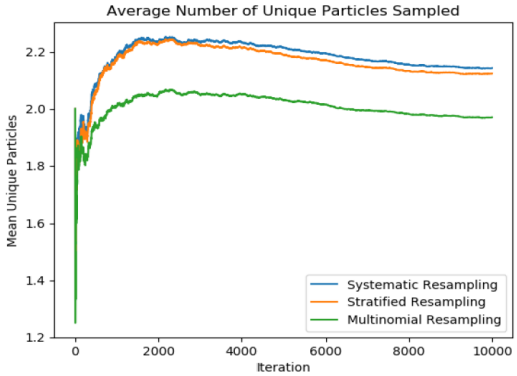
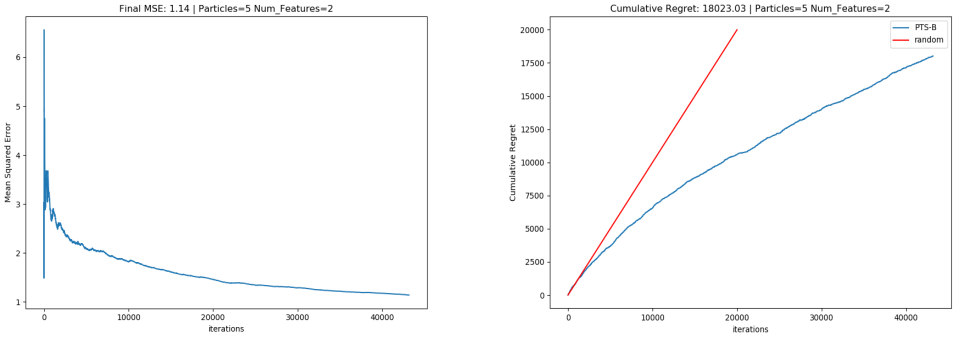


Figure 5: Comparison of resampling methods for K=2 and P=5

A common problem with particle filtering methods is degeneracy and sample impoverishment. To get a better sense for how these two common problems effected the algorithm we analyzed how the number of unique particles changed over time using three resampling methods: multinomial, stratified, and systematic resampling [3]. The plot shows that on average, systematic resampling produces more

unique particles at each iteration. Therefore, we used systematic resampling to evaluate mean squared error and cumulative regret of the model.



(a) Mean Squared Error

(b) Cumulative Regret

Figure 6: Movie-Lens 100k data

The final mean squared error using 5 particles and 2 latent features is 1.14. We compared the PTS-B cumulative regret with a random recommender system and found it to be significantly better with increasing time iterations.

Model Metric	**DeepICF**	**PTS**
HR	0.7025	0.1
NDCG	0.4315	0.031

Table 2: Comparison of the two models on the Movielens 1M dataset

From the comparison of the two models we see that the DeepICF has a better performance as compared to the PTS model when tested for a cold start scenario for various users in MovieLens 1M data.

5 Conclusion and Future Work

The goal of this project was to compare two different approaches for recommender systems - Particle Thompson Sampling [7] and DeepICF [8]. The DeepICF model yielded a higher hit rate and NDCG for the pinterest dataset than the movielens implying that such item-based collaborative filters are highly dependent on the number of items and a high item-user ratio. As for the Online Particle Thompson Sampler, we found that number of particles and number of features are two key attributes that influence the model performance and systematic resampling outperformed other resampling techniques. The algorithm outperforms all presented baselines in the paper with a lower cumulative regret compared to other popular models like Interactive Collaborative Filter, Spectral Bandit, etc. The performance of the DeepICF is comparatively better than the Online Particle Thompson Sampler on the Movielens dataset with a higher Hit Rate and NDCG on the test dataset with unseen items. This could imply that the neural network architecture with attention pooling outperforms sampling methods for a cold-start problem.

Online recommendation setting is a challenging problem with multiple hurdles like a cold-start, changing user preferences, balancing personalization and generalization, etc. The warrants future work to consider other traditional recommendation engines such as adaptive matrix factorization, plus online multi-arm bandit algorithms such as linUCB. Second, we would like to explore the interaction between users and item feature vectors during the training phase for high preference drift users. Lastly, we would also like to explore the effect of including additional user and item based features and cross embeddings based on user specific information from their profile as well as item specific information like movie genre, actors, description, etc. This could help improve recommendations by increasing the hit rate and NDCG of the recommender system.

References

[1] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[2] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344. ACM, 2017.

[3] Randal Douc and O Cappe. Comparison of resampling schemes for particle filtering. pages 64 – 69, 10 2005.

[4] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[5] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2354–2366, 2018.

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[7] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. Efficient thompson sampling for online matrix-factorization recommendation. In *Advances in neural information processing systems*, pages 1297–1305, 2015.

[8] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. Deep item-based collaborative filtering for top-n recommendation. *arXiv preprint arXiv:1811.04392*, 2018.