A Comparative Evaluation of Active Learning Methods in Deep Recommendation

Kai Luo, kluo@mie.utoronto.ca

May 2019

1 Motivation

There is little work on pushing active learning to latent space in recommendation tasks. The goal of this project is to show how deep recommender systems can actively learn from sequential user feedback and recommend personalized items to users.

2 Related Work

2.1 A Multi-armed Bandit Formulation

The problem of personalized recommendation can be naturally modeled as a multi-armed bandit problem with context information, which is called a contextual bandit. Formally, a contextual-bandit algorithm A proceeds in discrete trials $t = 1, 2, 3, \ldots$ In trial t [3]:

1. The algorithm observes the current user u_t and a set $\mathcal{A}_t$ of arms (or actions) together with their feature vectors $\boldsymbol{x}_{t,a}$ for $a \in \mathcal{A}_t$. The vector $\boldsymbol{x}_{t,a}$ will be referred to as the context and summarizes information of both the user u_t and arm a.
2. Based on observed reward in previous trials, A chooses an arm $a_t \in \mathcal{A}_t$, and receives reward r_{t,a_t}, the expectation of which depends on both the user u_t and the arm a_t.
3. The algorithm then improves its arm-selection strategy with the new observation, $(\boldsymbol{x}_{t,a_t}, a_t, r_{t,a_t})$.

2.2 Thompson Sampling

In Thompson sampling, there is a set of past observations D which is made of triplets $(\boldsymbol{x}_{t,a_t}, a_t, r_{t,a_t})$ where the likelihood function $p(r_{t,a_t}|a_t, \boldsymbol{x}_{t,a_t}, \theta)$ depends on some parameters θ. Given some prior distribution $p(\theta)$ on these parameters, the posterior distribution of these parameters is given by equation 1 as Bayes rule [1].

$$p(\theta|D) \propto \prod p(r_{t,a_t}|a_t, \boldsymbol{x}_{t,a_t}, \theta)p(\theta) \tag{1}$$

In practice, the reward is a stochastic function of the action, context and the unknown, true parameter θ^*. Ideally, we would like to choose an arm a_t maximizing the expected reward, $\max_{a_t} \mathbb{E}(r_{t,a_t}|a_t, \boldsymbol{x}_{t,a_t}, \theta^*)$.

In an exploration / exploitation setting, the probability matching heuristic consists of randomly selecting an arm a_t according to its probability of being optimal. That is, arm a_t is chosen with probability as shown in equation 2, where $\mathbb{1}$ is the indicator function. Note that the integral does not have to be computed explicitly: it suffices to sample from parameter θ at each round.

$$\int \mathbb{1}\Big[\mathbb{E}(r_{t,a_t}|a_t, \boldsymbol{x}_{t,a_t}, \theta) = \max_{a'_t} \mathbb{E}(r_{t,a_t}|a'_t, \boldsymbol{x}_{t,a_t}, \theta)\Big]p(\theta|D)d\theta \tag{2}$$

2.3 Upper Confidence Bound

In trial t, upper confidence bound algorithms estimate both the mean payoff $\hat{\mu}_{t,a}$ of each arm a as well as a corresponding confidence interval $c_{t,a}$, so that $|\hat{\mu}_{t,a} - \mu_a| < c_{t,a}$ holds with high probability, where μ_a is the true value. They then select the arm that achieves a highest upper confidence bound (UCB for short): $a_t = \text{argmax}_a(\hat{\mu}_{t,a} + c_{t,a})$ [3].

2.4 Variational Autoencoders for Collaborative Filtering

We will use variational autoencoders for collaborative filtering (VAE-CF) [4] as the core deep recommender system. It extends variational autoencoders to include collaborative filtering for implicit feedback, and models user-item implicit feedback using multinomial conditional likelihood.

In Figure 1, $\boldsymbol{x}$ represents user click history, q is the encoder parameter, p is the decoder parameter, $\boldsymbol{\epsilon}$ is a standard Gaussian distribution, $\boldsymbol{z}$ represents the user latent representation, $\boldsymbol{\mu}$ is mean of $\boldsymbol{z}$ and $\boldsymbol{\sigma}$ is standard deviation of $\boldsymbol{z}$.

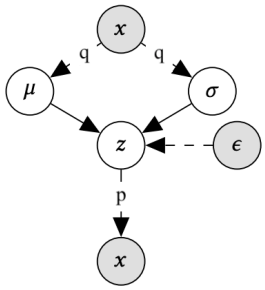

Figure 1: VAE-CF. The dotted arrows denote a sampling operation.

3 Experiments

Now we proceed to evaluate the performance of doing active learning using the VAE-CF model in order to answer the following questions:

- Is active learning helping to boost recommendation general performance across?
- Is doing active learning iteratively on active set increasing recommender system's general performance?

3.1 Datasets and Statistics

We evaluate VAE-CF on the MovieLens-1M and Yelp datasets. We binarize the rating dataset with a rating threshold, ϑ, defined to be the upper half of the range of the ratings. This ensures all users feedback is implicit. To be specific, the threshold is $\vartheta > 3$ out of 5. Table 1 summarizes the properties of the binarized metrics where m and n are the number of users and items.

Table 1: Summary of dataset.

Dataset	m	n	$\lvert r_{i,j} > \vartheta \rvert$	Sparsity
ML1M	6,038	3,533	575,281	2.69×10^{-2}
Yelp	6,075	3,999	195,385	0.80×10^{-2}

3.2 Experiment Steps

1. Randomly choose half of users and their ratings in the rating dataset as test set.
2. Split the rest of positive ratings evenly into train set and validation set by timestamp.

3. After tuning the hyperparameters on the validation set, we merge the train set and validation set to be the train set.
4. Split the test set into active set and test set by timestamp with ratio 7:3.
5. Train the VAE-CF model using the train set.
6. Perform active learning on the active set with completely cold start.
7. Merge the train set and positive items recovered from active learning to be the train set.
8. Re-train the VAE-CF model using the train set and evaluate recommendation performance on the test set.

3.3 Compared Algorithms

The core active learning algorithms empirically evaluated in our experiments are as follows:

- `greedy (Gr)`: This policy chooses items of the highest expected reward.
- `Thompson sampling (Th)`: This policy randomly picks each arm conditioned on its probability of being optimal.
- `UCB`: This policy picks items of the highest upper confidence bound.

Other factors that can be combined with active learning algorithms are described as follows:

- `non-iterative (NI)`: Active learning algorithms select items from active set in one time.
- `iterative (Iter)`: Active learning algorithms iteratively select items from active set.
- `select only from positive samples (Pos)`: Active learning algorithms only select items liked by users in active set.
- `select from positive and negative samples (All)`: Active learning algorithms select items from all items in active set.

3.4 Performance Metric

For Top-N recommendation performance, we evaluate the proposed methods on five metrics: MAP@N, Precision@N, Recall@N, R-Precision, and NDCG [2].

3.5 Experimental Results

Table 2: Results of performing active learning iteratively and sampling from all data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Model	C	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
Gr	0	**0.1695**	**0.2331**	0.5679	0.5253	0.5258	0.4567	0.0379	0.0650
Th	0	0.1671	0.2305	0.5613	0.5206	0.5219	0.4535	0.0375	0.0646
UCB	0.5	0.1681	0.2319	0.5683	0.5253	**0.5317**	**0.4574**	**0.0383**	**0.06525**
UCB	1	0.1691	0.2330	0.5699	0.5261	0.5267	0.4559	0.0379	0.0649
UCB	2	0.1686	0.2329	**0.5731**	**0.5284**	0.5315	0.4573	0.03826	0.06524

Table 3: Results of performing active learning iteratively and sampling from only positive data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Model	C	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
Gr	0	**0.2337**	**0.3011**	0.6254	0.5841	0.5809	0.5227	0.0424	0.0759
Th	0	0.2323	0.2997	0.6215	0.5827	0.5822	0.5223	0.0425	0.0759
UCB	0.5	0.2313	0.3001	**0.6318**	**0.5897**	**0.5888**	**0.5279**	**0.0431**	**0.0768**
UCB	1	0.2325	0.3001	0.6241	0.5840	0.5820	0.5228	0.0426	0.0760
UCB	2	0.2323	0.3000	0.6254	0.5855	0.5825	0.5241	0.0427	0.0762

Table 4: Results of performing active learning non-iteratively and sampling from all data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Model	C	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
Gr	0	0.1610	**0.2243**	**0.5595**	**0.5162**	**0.5191**	**0.4467**	**0.0374**	**0.0636**
Th	0	0.1609	0.2175	0.5184	0.4746	0.4704	0.4073	0.0338	0.0578
UCB	0.5	**0.1613**	0.2221	0.5488	0.5030	0.4998	0.4306	0.0359	0.0613
UCB	1	0.1612	0.2212	0.5414	0.4982	0.4989	0.4304	0.0359	0.0613
UCB	2	0.1603	0.2181	0.5315	0.4845	0.4825	0.4111	0.0345	0.0585

Table 5: Results of performing active learning non-iteratively and sampling from only positive data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Model	C	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
Gr	0	**0.2233**	**0.2939**	0.6364	**0.5995**	**0.6030**	**0.5391**	**0.0442**	**0.0785**
Th	0	0.2188	0.2857	0.6136	0.5736	0.5732	0.5110	0.0420	0.0748
UCB	0.5	0.2214	0.2918	**0.6383**	0.5977	0.5986	0.5317	0.0438	0.0775
UCB	1	0.2205	0.2897	0.6263	0.5877	0.5868	0.5267	0.0432	0.0770
UCB	2	0.2198	0.2887	0.6266	0.5872	0.5893	0.5225	0.0433	0.0764

Table 6: Results of greedy algorithm and sampling from all data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1610	0.2243	0.5595	0.5162	0.5191	0.4467	0.0374	0.0636
True	**0.1695**	**0.2331**	**0.5679**	**0.5253**	**0.5258**	**0.4567**	**0.0379**	**0.0650**

Table 7: Results of greedy algorithm and sampling from only positive data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.2233	0.2939	**0.6364**	**0.5995**	**0.6030**	**0.5391**	**0.0442**	**0.0785**
True	**0.2337**	**0.3011**	0.6254	0.5841	0.5809	0.5227	0.0424	0.0759

Table 8: Results of Thompson sampling and sampling from all data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1609	0.2175	0.5184	0.4746	0.4704	0.4073	0.0338	0.0578
True	**0.1671**	**0.2305**	**0.5613**	**0.5206**	**0.5219**	**0.4535**	**0.0375**	**0.0646**

Table 9: Results of Thompson sampling and sampling from only positive data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.2188	0.2857	0.6136	0.5736	0.5732	0.5110	0.0420	0.0748
True	**0.2323**	**0.2997**	**0.6215**	**0.5827**	**0.5822**	**0.5223**	**0.0425**	**0.0759**

Table 10: Results of UCB (C=0.5) and sampling from all data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1613	0.2221	0.5488	0.5030	0.4998	0.4306	0.0359	0.0613
True	**0.1681**	**0.2319**	**0.5683**	**0.5253**	**0.5317**	**0.4574**	**0.0383**	**0.0652**

Table 11: Results of UCB (C=0.5) and sampling from only positive data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.2214	0.2918	**0.6383**	**0.5977**	**0.5986**	**0.5317**	**0.0438**	**0.0775**
True	**0.2313**	**0.3001**	0.6318	0.5897	0.5888	0.5279	0.0431	0.0768

Table 12: Results of UCB (C=1.0) and sampling from all data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1612	0.2212	0.5414	0.4982	0.4989	0.4304	0.0359	0.0613
True	**0.1691**	**0.2330**	**0.5699**	**0.5261**	**0.5267**	**0.4559**	**0.0379**	**0.0649**

Table 13: Results of UCB (C=1.0) and sampling from only positive data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.2205	0.2897	**0.6263**	**0.5877**	**0.5868**	**0.5267**	**0.0432**	**0.0770**
True	**0.2325**	**0.3001**	0.6241	0.5840	0.5820	0.5228	0.0426	0.0760

Table 14: Results of UCB (C=2.0) and sampling from all data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1603	0.2181	0.5315	0.4845	0.4825	0.4111	0.0345	0.0585
True	**0.1686**	**0.2329**	**0.5731**	**0.5284**	**0.5315**	**0.4573**	**0.0383**	**0.0652**

Table 15: Results of UCB (C=2.0) and sampling from only positive data on ML1M dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.2198	0.2887	**0.6266**	**0.5872**	**0.5893**	0.5225	**0.0433**	**0.0764**
True	**0.2323**	**0.3000**	0.6254	0.5855	0.5825	**0.5241**	0.0427	0.0762

Table 16: Results of performing active learning iteratively and sampling from all data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Model	C	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
Gr	0	**0.1067**	0.1356	0.2995	0.2573	0.2578	0.1978	**0.0212**	0.0312
Th	0	0.0922	0.1205	0.2784	0.2480	0.2511	0.2000	0.0198	0.0312
UCB	0.5	0.1036	0.1356	0.3097	0.2678	0.2578	**0.2078**	0.0210	**0.0330**
UCB	1	0.1053	**0.1380**	**0.3252**	**0.2685**	0.2511	0.2000	0.0202	0.0322
UCB	2	0.0992	0.1291	0.2955	0.2569	**0.2600**	0.1978	0.0209	0.0312

Table 17: Results of performing active learning iteratively and sampling from only positive data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Model	C	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
Gr	0	0.1803	0.2197	0.3685	0.3578	0.3444	0.3422	0.0289	0.0572
Th	0	0.1847	**0.2269**	**0.4131**	**0.3816**	**0.3778**	0.3478	**0.0314**	0.0578
UCB	0.5	0.1842	0.2257	0.3777	0.3760	0.3756	**0.3656**	0.0313	**0.0616**
UCB	1	0.1825	0.2193	0.3748	0.3558	0.3600	0.3300	0.0301	0.0553
UCB	2	**0.1856**	0.2263	0.3860	0.3707	0.3733	0.3478	0.0312	0.0582

Table 18: Results of performing active learning non-iteratively and sampling from all data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Model	C	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
Gr	0	**0.0768**	**0.1015**	**0.2522**	**0.2201**	**0.2156**	**0.1767**	**0.0161**	**0.0262**
Th	0	0.0630	0.0785	0.1638	0.1443	0.1467	0.1211	0.0112	0.0190
UCB	0.5	0.0746	0.0980	0.2327	0.1999	0.1911	0.1533	0.0151	0.0244
UCB	1	0.0660	0.0852	0.2017	0.1757	0.1733	0.1400	0.0137	0.0220
UCB	2	0.0615	0.0769	0.1621	0.1410	0.1422	0.1144	0.0113	0.0176

Table 19: Results of performing active learning non-iteratively and sampling from only positive data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Model	C	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
Gr	0	**0.1733**	**0.2142**	0.4064	0.3816	0.3689	**0.3300**	0.0304	0.0544
Th	0	0.1671	0.2033	0.3696	0.3411	0.3333	0.2989	0.0270	0.0488
UCB	0.5	0.1693	0.2101	0.3957	0.3659	0.3667	0.3233	0.0305	0.0533
UCB	1	0.1705	0.2112	**0.4079**	**0.3840**	**0.3978**	**0.3300**	**0.0327**	**0.0545**
UCB	2	0.1663	0.2025	0.3535	0.3391	0.3489	0.3011	0.0286	0.0498

Table 20: Results of greedy algorithm and sampling from all data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.0768	0.1015	0.2522	0.2201	0.2156	0.1767	0.0161	0.0262
True	**0.1067**	**0.1356**	**0.2995**	**0.2573**	**0.2578**	**0.1978**	**0.0212**	**0.0312**

Table 21: Results of greedy algorithm and sampling from only positive data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1733	0.2142	**0.4064**	**0.3816**	**0.3689**	0.3300	**0.0304**	0.0544
True	**0.1803**	**0.2197**	0.3685	0.3578	0.3444	**0.3422**	0.0289	**0.0572**

Table 22: Results of Thompson sampling and sampling from all data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.0630	0.0785	0.1638	0.1443	0.1467	0.1211	0.0112	0.0190
True	**0.0922**	**0.1205**	**0.2784**	**0.2480**	**0.2511**	**0.2000**	**0.0198**	**0.0312**

Table 23: Results of Thompson sampling and sampling from only positive data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1671	0.2033	0.3696	0.3411	0.3333	0.2989	0.0270	0.0488
True	**0.1847**	**0.2269**	**0.4131**	**0.3816**	**0.3778**	**0.3478**	**0.0314**	**0.0578**

Table 24: Results of UCB (C=0.5) and sampling from all data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.0746	0.0980	0.2327	0.1999	0.1911	0.1533	0.0151	0.0244
True	**0.1036**	**0.1356**	**0.3097**	**0.2678**	**0.2578**	**0.2078**	**0.0210**	**0.0330**

Table 25: Results of UCB (C=0.5) and sampling from only positive data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1693	0.2101	**0.3957**	0.3659	0.3667	0.3233	0.0305	0.0533
True	**0.1842**	**0.2257**	0.3777	**0.3760**	**0.3756**	**0.3656**	**0.0313**	**0.0616**

Table 26: Results of UCB (C=1.0) and sampling from all data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.0660	0.0852	0.2017	0.1757	0.1733	0.1400	0.0137	0.0220
True	**0.1053**	**0.1380**	**0.3252**	**0.2685**	**0.2511**	**0.2000**	**0.0202**	**0.0322**

Table 27: Results of UCB (C=1.0) and sampling from only positive data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1705	0.2112	**0.4079**	**0.3840**	**0.3978**	**0.3300**	0.0327	0.0545
True	**0.1825**	**0.2193**	0.3748	0.3558	0.3600	**0.3300**	**0.0301**	**0.0553**

Table 28: Results of UCB (C=2.0) and sampling from all data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.0615	0.0769	0.1621	0.1410	0.1422	0.1144	0.0113	0.0176
True	**0.0992**	**0.1291**	**0.2955**	**0.2569**	**0.2600**	**0.1978**	**0.0209**	**0.0312**

Table 29: Results of UCB (C=2.0) and sampling from only positive data on Yelp dataset. We omit the error bars since the confidence interval is in 4th digit.

Iter	R-Precision	NDCG	MAP@5	MAP@10	Precision@5	Precision@10	Recall@5	Recall@10
False	0.1663	0.2025	0.3535	0.3391	0.3489	0.3011	0.0286	0.0498
True	**0.1856**	**0.2263**	**0.3860**	**0.3707**	**0.3733**	**0.3478**	**0.0312**	**0.0582**

3.5.1 Comparison of Recommendation Performance using Active Learning for All Algorithms

Tables 2 to 5 and 16 to 19 show the comparison of general recommendation performance using active learning between all algorithms on ML1M and Yelp datasets. In the tables, we make the following key observations:

1. When sampling from all data, UCB outperforms the rest if active learning is done iteratively, whereas greedy algorithm has the best performance when active learning is done in one shot.
2. Thompson sampling only performs the best when doing active learning iteratively and sampling from only positive data on Yelp dataset.

3.5.2 Comparison of Recommendation Performance using Active Learning Iteratively per Algorithm

Tables 6 to 15 and 20 to 29 show the comparison of general recommendation performance using active learning iteratively for each algorithm on ML1M and Yelp datasets. In the tables, we make the following key observations:

1. Sampling from all data in active learning always boosts recommendation performance.
2. Given sampling from only positive data, performing active learning iteratively does not help increase recommendation performance except for Thompson sampling on ML1M dataset.
3. Doing active learning iteratively always helps to increase performance on Yelp dataset.
4. To summarize the above three observations, we can conclude that doing active learning iteratively improves general performance when the number of positive samples is limited (learns from all data). If the model sees enough positive samples (sample from positive data only), it depends on the dataset.

3.5.3 Comparison of Recommendation Performance Between Two Datasets

Compared to the metrics results from Yelp dataset under the same active learning settings and model, the metrics results from ML1M dataset are generally better. This is very likely because Yelp dataset is much sparse and largely localized.

References

[1] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[2] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[3] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[4] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 689–698. International World Wide Web Conferences Steering Committee, 2018.

4 Appendix

4.1 VAE-CF in math

VAE-CF optimizes equation 3 where β is the regularization term.

$$\mathcal{L}_\beta(\boldsymbol{x}_{t,a}) \equiv \mathbb{E}_{q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})}[\log p(\boldsymbol{x}_{t,a}|\boldsymbol{z}_{t,a})] - \beta \cdot KL(q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})||p(\boldsymbol{z}_{t,a})) \tag{3}$$

4.2 Objective in math

Let $p(a_t|\boldsymbol{x}_{t,a})$ represent observed reward. We can factorize it in equation 4. However, computing $p(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})$ is extremely expensive.

By leveraging VAE-CF, we are able to easily approximate $p(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})$ with $q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})$ using Kullback-Leiber divergence as shown below. It is also worth mentioning that $p(a_t|\boldsymbol{z}_{t,a})$ can be considered as the decoder in equation 4.

$$\begin{aligned}
\log p(a_t|\boldsymbol{x}_{t,a}) &= \log \int_z p(a_t|\boldsymbol{z}_{t,a}) p(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a}) dz && (4)\\
&= \log \int_{\mathbf{z}} q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a}) \frac{p(a_t|\boldsymbol{z}_{t,a}) p(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})}{q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})} dz && (5)\\
&\geq \int_{\mathbf{z}} q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a}) \log \frac{p(a_t|\boldsymbol{z}_{t,a}) p(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})}{q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})} dz && (6)\\
&= \int_{\mathbf{z}} q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a}) \log p(a_t|\boldsymbol{z}_{t,a}) dz && (7)\\
&\quad + \int_{\mathbf{z}} q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a}) \log \frac{p(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})}{q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})} dz && (8)\\
&= \mathbb{E}_{q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})} \log p(a_t|\boldsymbol{z}_{t,a}) && (9)\\
&\quad - KL(q(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a})||p(\boldsymbol{z}_{t,a}|\boldsymbol{x}_{t,a}))) && (10)
\end{aligned}$$

This is commonly known as the evidence lower bound (ELBO).

VAE-CF assumes:

- $q(\boldsymbol{z}_{t=0,a=\emptyset}) \sim \mathcal{N}(0, 1)$. In other words, user latent representation z has a standard Gaussian as prior which is a tractable distribution. It also means that VAE-CF restricts user latent representation into a very small latent space, which helps in generalization.
- $q(\boldsymbol{z}_{t,a}) \sim \mathcal{N}(\mu_t, \sigma_t^2)$ means that user latent representation is stochastic since we sample user latent representation from Gaussian each iteration. This also ensures that the observed reward is stochastic.

4.3 Implementation

The pseudocode of standard Thompson sampling is shown in Algorithm 1 [1].

Algorithm 1 Thompson Sampling

$D = \emptyset$
for t = 1, 2, ..., T **do**
    Receive context $\boldsymbol{x}_{t,a_t}$
    Sample $\theta_t \sim p(\theta|D)$
    Select $a_t \leftarrow \text{argmax}_{a \in \mathcal{A}_t} \mathbb{E}_r[r_{t,a_t}|a_t, \boldsymbol{x}_{t,a_t}, \theta_t]$
    Observe reward r_{t,a_t}
    $D = D \cup (\boldsymbol{x}_{t,a_t}, a_t, r_{t,a_t})$
end for

The pseudocode of the proposed latent Thompson sampling is in Algorithm 2.

Algorithm 2 Latent Thompson Sampling for Recommendation

$D = \emptyset$
$train(model)$
for t = 1, 2, 3, ..., T **do**
    $\mathbf{z}_{u,\mu}, \mathbf{z}_{u,\sigma^2} \leftarrow encode(model, D)$

    $\mathbf{z}'_{u,\mu} \sim \mathcal{N}(\mathbf{z}_{u,\mu}, \mathbf{z}_{u,\sigma^2})$
    $\mathbb{E}_r[r_{t,a_t}|a_t, \boldsymbol{z}_{t,a_t}, \theta_t] \leftarrow decode(model, \mathbf{z}'_{u,\mu})$

    Select $a_t \leftarrow \text{argmax}_{a \in \mathcal{A}_t} \mathbb{E}_r[r_{t,a_t}|a_t, \boldsymbol{z}_{t,a_t}, \theta_t]$
    Observe reward r_{t,a_t}
    $D = D \cup (\boldsymbol{x}_{t,a_t}, a_t, r_{t,a_t})$
end for

The pseudocode of the proposed latent UCB is in Algorithm 3.

Algorithm 3 Latent UCB for Recommendation

$D = \emptyset$
$train(model)$
for t = 1, 2, 3, ..., T **do**
    $\mathbf{z}_{u,\mu}, \mathbf{z}_{u,\sigma^2} \leftarrow encode(model, D)$

    $R = \emptyset$
    for k = 1, 2, 3, ..., K **do**
        $\mathbf{z}'_{u,\mu} \sim \mathcal{N}(\mathbf{z}_{u,\mu}, \mathbf{z}_{u,\sigma^2})$
        $\mathbb{E}_r[r_{t,a_t,k}|a_{t,k}, \boldsymbol{z}_{t,a_t,k}, \theta_t] \leftarrow decode(model, \mathbf{z}'_{u,\mu})$
        $R = R \cup (\mathbb{E}_r[r_{t,a_t,k}|a_{t,k}, \boldsymbol{z}_{t,a_t,k}, \theta_t])$
    end for
    $\mathbb{E}_r[r_{t,a_t}|a_t, \boldsymbol{z}_{t,a_t}, \theta_t] = \overline{R_{t,a_t}} + C \times std(R_{t,a_t})$

    Select $a_t \leftarrow \text{argmax}_{a \in \mathcal{A}_t} \mathbb{E}_r[r_{t,a_t}|a_t, \boldsymbol{z}_{t,a_t}, \theta_t]$
    Observe reward r_{t,a_t}
    $D = D \cup (\boldsymbol{x}_{t,a_t}, a_t, r_{t,a_t})$
end for

Table 30: Non-active nearest neighbor and variational autoencoders for collaborative filtering on Yelp dataset.

Model	MAP@10	MAP@20	MAP@50	Precision@10	Precision@20	Precision@50	Recall@10	Recall@20	Recall@50
NN	0.0049	0.0078	0.0193	0.0077	0.0156	**0.0386**	0.0009	0.0019	0.0047
VAE-CF	**0.0582**	**0.0537**	**0.0464**	**0.0524**	**0.0467**	0.0381	**0.0842**	**0.1428**	**0.2769**