2022 2학기 강화학습특론 **Project Final Presentation**

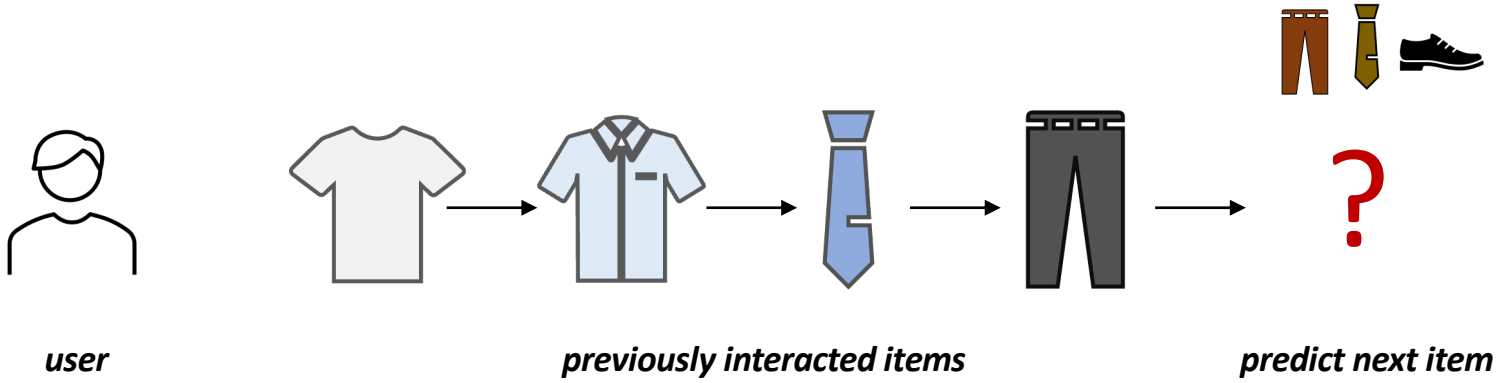
Sequential Recommendation with Reinforcement Learning

김수경 tnrud929@g.skku.edu
양희윤 chri0220@skku.edu

목차

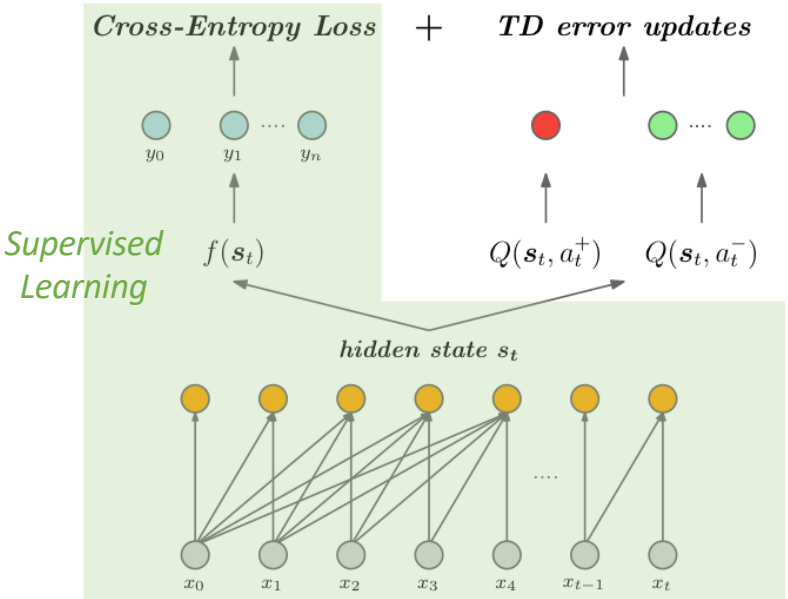
1. Introduction

What is sequential recommendation?



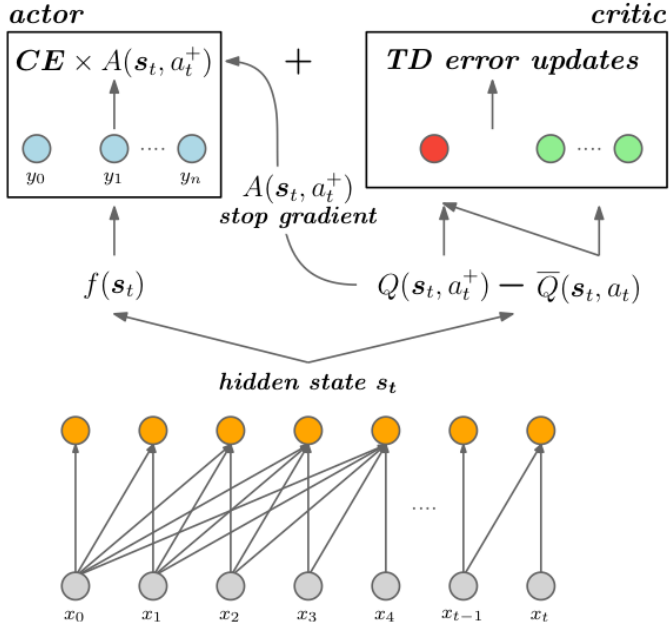
1. Introduction

Baseline: The Learning Framework Architectures of SNQN and SA2C



(a) SNQN architecture.

Supervised Negative Q-Learning



(b) SA2C architecture. CE is short for cross-entropy.

Supervised Advantage Actor-Critic

Xin, Xin, et al. "Supervised Advantage Actor-Critic for Recommender Systems." *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022.

1. Introduction

Our Goal: Advanced Sequential Recommendation with RL

- **Disadvantage of SNQN, SA2C**
  - 현재 state를 고려하지 않고 negative sample 을 무작위로 선정하기 때문에 학습에 도움이 되지 않을 수 있음
  - 목적함수로 추천의 정확도만을 고려하여 사용자의 추천 만족도를 달성하지 못할 수 있음 (ex. 다양성)

- **SNQN, SA2C의 문제점을 개선 → 사용자의 추천 만족도를 높일 수 있는 모델 제안**
  - 아이템 또는 사용자의 부가적인 정보를 활용 → 더 나은 representation 학습
  - 정확도와 다양성을 고려한 multi-objective learning 도입
  - 현재 상태를 고려한 negative sampling 방식 도입

2. Proposed Method

Reinforcement Learning Formulation of Sequential Recommendation

Agent: recommender system (RS)

Environment: user

State $\mathcal{S}$: user state space based on previously interacted items

Action $\mathcal{A}$: items available for recommendation

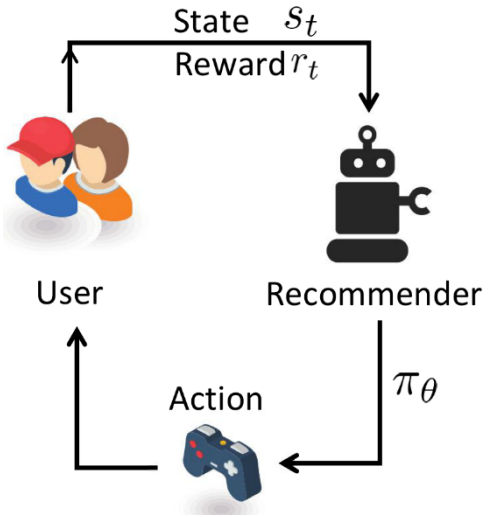
Reward $\mathcal{R}$: user feedback on the recommended item

(if the user clicks, reward = 1 , otherwise reward = 0)

Transition probability $\mathcal{P}$: state transition probability

(assumption: only positive items can affect user state)

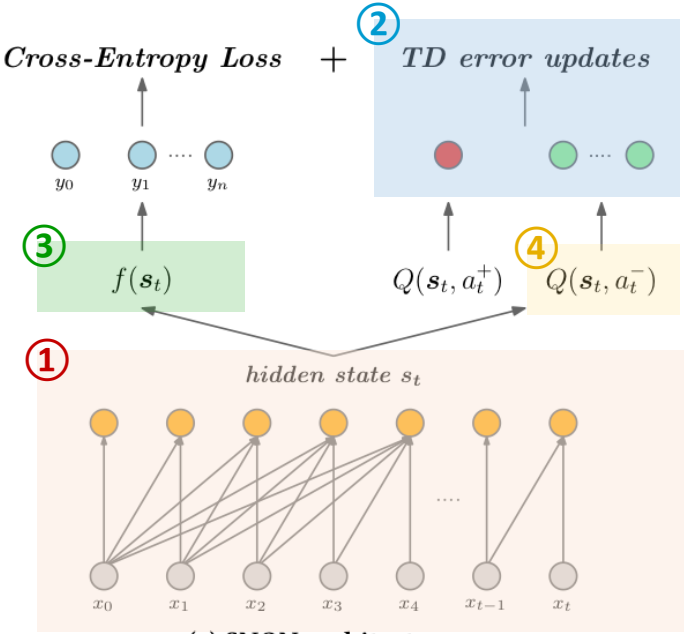
Discount factor γ : discount factor for future rewards



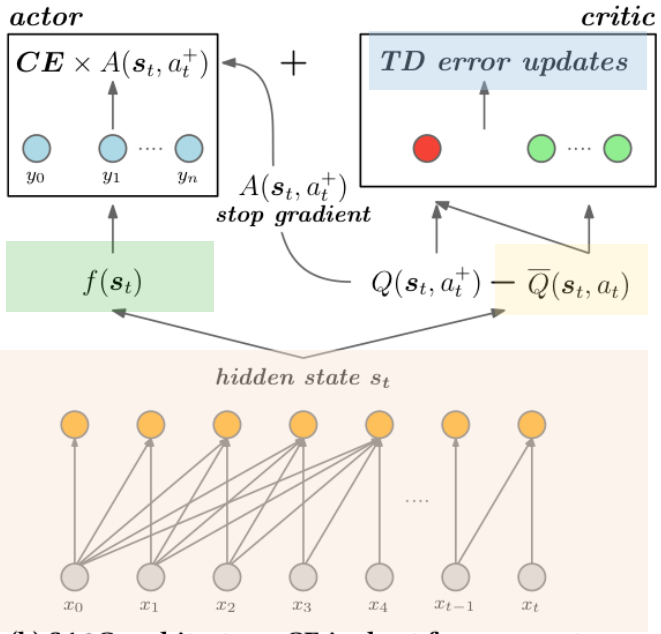
Usually, offline learning is performed from the logged implicit feedback.

2. Proposed Method

Model Framework with ours



(a) SNQN architecture.



(b) SA2C architecture. CE is short for cross-entropy.

① Sequential Encoding with features

② BCQ (Batch-Constrained Deep Q-Learning)

③ Normalized Recommendation

④ Batched Negative Sampling

2. Proposed Method

① *Sequential Encoding with item features – item features*

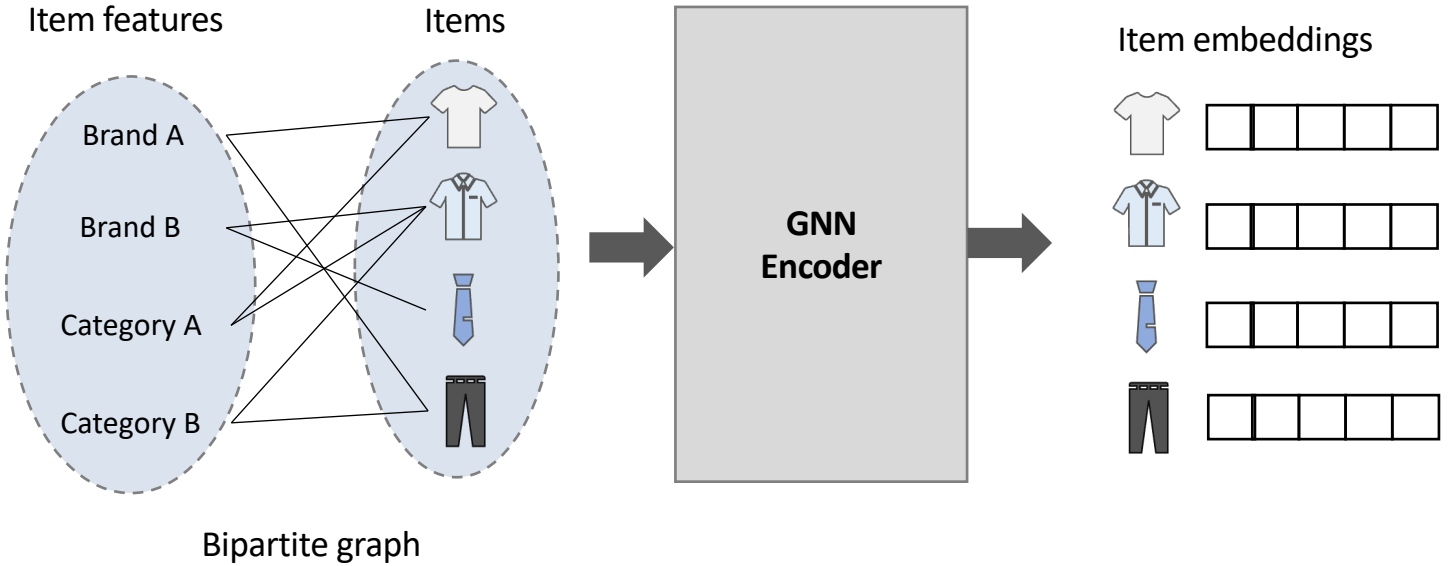
- **아이템의 부가적인 정보를 활용 → 더 나은 representation 을 학습**
  - ✓ 아이템 정보가 포함된 데이터셋 선정
  - ✓ LastFM: 음악 아티스트 추천 데이터로, 아이템의 속성으로 아티스트 이름/장르 포함
  - ✓ Amazon Beauty: Amazon review 에서 수집된 데이터로, 아이템의 속성으로 상품 카테고리/브랜드 포함

Dataset	LastFM	Amazon Beauty
# Users	1,090	22,363
# Items	3,646	12,101
# Interactions	52,551	198,502
# Sparsity	98.68%	99.93%
# Attributes	388	1,221
# Avg.Attributes / Item	31.5	5.1

2. Proposed Method

① *Sequential Encoding with item features – item features*

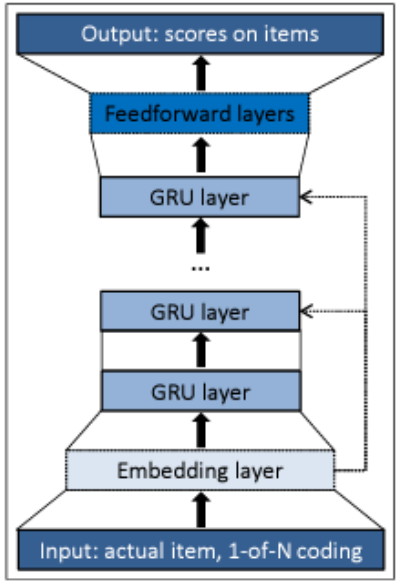
- **아이템의 부가적인 정보를 활용 → 더 나은 representation 을 학습**
  - ✓ How to encode item features into item embedding? GNN encoder



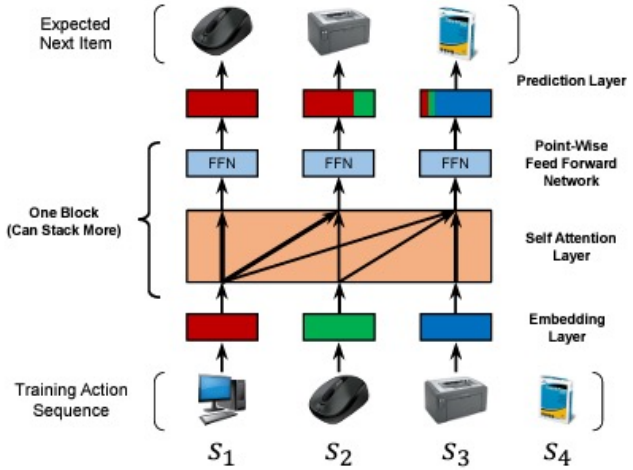
2. Proposed Method

① *Sequential Encoding with item features – Sequential Encoder*

- **Sequential Modeling 에 용이한 GRU 와 Self Attention 기반 Encoder 사용**



[1] GRU4Rec



[2] SASRec

Hidashi et al. "Session-based Recommendations With Recurrent Neural Networks," ICRL '16
Kang et al. "Self-Attentive Sequential Recommendation," ICDM '18

2. Proposed Method

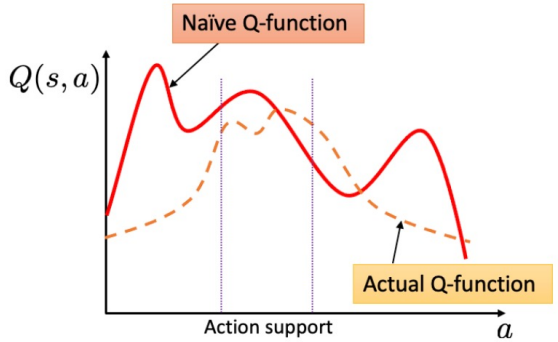
② *BCQ (Batch-Constrained Deep Q-Learning)*

- **사용자의 추천 만족도를 높이기 위해 정확도와 다양성을 함께 고려**

  ✓ SNQN의 Q-loss

$$L_q = \underbrace{(r(\mathbf{s}_t, a_t^+) + \gamma \max_{a'} Q(\mathbf{s}_{t+1}, a') - Q(\mathbf{s}_t, a_t^+))^2}_{L_p\text{: positive TD error}} + \underbrace{\sum_{a_t^- \in N_t} (r(\mathbf{s}_t, a_t^-) + \gamma \max_{a'} Q(\mathbf{s}_t, a') - Q(\mathbf{s}_t, a_t^-))^2}_{L_n\text{: negative TD error}},$$

  ✓ Offline learning의 Q-value overestimation 문제가 발생할 수 있음



https://bair.berkeley.edu/blog/2020/12/07/offline/

2. Proposed Method

② *BCQ (Batch-Constrained Deep Q-Learning)*

- **사용자의 추천 만족도를 높이기 위해 정확도와 다양성을 함께 고려 – 정확도 향상을 위한 BCQ**

  ✓ Discrete Batch-Constrained Deep Q-learning(BCQ): generative model $G_\omega(a|s)$ 을 이용하여 batch 내에 저장되어 있는 (s,a) 조합과 유사한 action sampling

$$\pi(s) = \underset{a|G_\omega(a|s)/\max_{\hat{a}} G_\omega(\hat{a}|s)>\tau}{\operatorname{argmax}} Q_\theta(s,a). \qquad G_\omega(a|s) \approx \pi_b(a|s)$$

Algorithm 1 BCQ

Input: Batch $\mathcal{B}$, number of iterations T, target_update_rate, mini-batch size N, threshold τ.
Initialize Q-network Q_θ, generative model G_ω and target network $Q_{\theta'}$ with $\theta' \leftarrow \theta$.
for $t = 1$ **to** T **do**
  Sample mini-batch M of N transitions (s, a, r, s') from $\mathcal{B}$.
  $a' = \operatorname{argmax}_{a'|G_\omega(a'|s')/\max_{\hat{a}} G_\omega(\hat{a}|s')>\tau} Q_\theta(s', a')$
  $\theta \leftarrow \operatorname{argmin}_\theta \sum_{(s,a,r,s')\in M} l_\kappa\left(r + \gamma Q_{\theta'}(s', a') - Q_\theta(s, a)\right)$
  $\omega \leftarrow \operatorname{argmin}_\omega - \sum_{(s,a)\in M} \log G_\omega(a|s)$
  If t mod target_update_rate $= 0$: $\theta' \leftarrow \theta$
end for

SNQN의 supervised loss

$$L_s = -\sum_{i=1}^{n} Y_i log(p_i), \text{where } p_i = \frac{e^{y_i}}{\sum_{i'=1}^{n} e^{y_{i'}}}$$

Fujimoto, Scott, et al. "Benchmarking batch deep reinforcement learning algorithms." Deep RL Workshop NeurIPS '19.

2. Proposed Method

② *BCQ (Batch-Constrained Deep Q-Learning)*

- **사용자의 추천 만족도를 높이기 위해 정확도와 다양성을 함께 고려 – 정확도 향상을 위한 BCQ**

✓ 적용 아이디어

- SNQN의 supervised part을 generative model로 사용

$G_w(a|s) = P_{sup}(a|s)$

- Our new q-loss

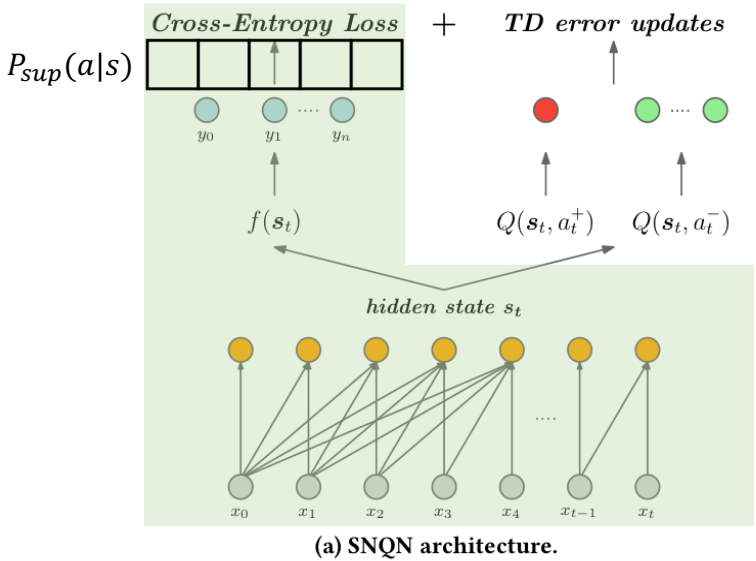
$$\max_{a'|P_{sup}(a'|s_{t+1})/max\,\hat{a}\,P_{sup}(\hat{a}|s_{t+1})>\tau} Q\,(s_{t+1}, a')$$

$$L_q = (r(\mathbf{s}_t, a_t^+) + \gamma \max_{a'} Q(\mathbf{s}_{t+1}, a') - Q(\mathbf{s}_t, a_t^+))^2$$

- $\tau = 0$: Q-learning

$\tau = 1$: Behavior cloning

해당 state에서 각 item을 클릭할 확률

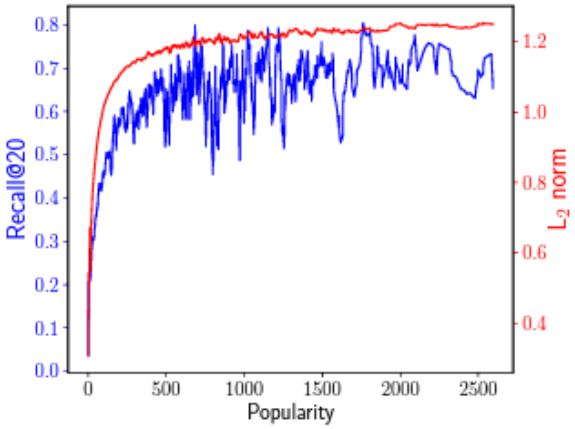


(a) SNQN architecture.

2. Proposed Method

③ *Normalized Recommendation*

○ **Recommendation 과정에서 발생할 수 있는 Popularity Bias 문제 완화를 위해 Normalizing 적용**



→ ***Recall*@20 and L_2 norm of learned item embedding decrease with decreasing popularity**

✓ **Original Recommendation Probability**

: inner product of item embeddings and sequence embedding

$$p_k(\mathbf{s}) = \hat{\mathbf{y}}_k = \frac{\exp(\mathbf{i}_k^T \mathbf{s})}{\sum_{j=1}^{m} \exp(\mathbf{i}_j^T \mathbf{s})}.$$

✓ **Normalized Recommendation Probability**

: replace embeddings to normalized embeddings

$$\hat{\mathbf{y}}_k = \frac{\exp(\sigma \tilde{\mathbf{i}}_k^T \tilde{\mathbf{s}})}{\sum_{j=1}^{m} \exp(\sigma \tilde{\mathbf{i}}_j^T \tilde{\mathbf{s}})}. \quad \tilde{\mathbf{i}}_k = \frac{\mathbf{i}_k}{||\mathbf{i}_k||_2}$$

➔ **In our case,**

$\hat{\mathbf{y}}_k = \tilde{\imath}^\top \tilde{u}$, $L_s = CrossEntropy(\hat{\mathbf{y}}, \mathbf{y})$

[3] *Gupta et al. NISER: Normalized Item and Session Representation to Handle Popularity Bias, CIKM Workshop '19*

2. Proposed Method

④ *Batched Negative Sampling*

- **현재 상태를 고려한 Negative sampling 방식 도입**
  - ✓ SNQN, SA2C 의 경우에는 해당 시점의 action이 아닌 것을 제외한 **모든 다른 action 중 10개를 random sampling 하여 negative action으로 사용**
  - ✓ 예측 시점에서는 사용자의 현재 상태 뿐만 아니라, 이전에 사용자가 상호작용 했던 아이템들에 대한 선호 정보도 **모두 포함**
    → 이전 시점에서 상호작용했던 아이템들까지 **negative item 후보군에 포함**하는 것은 모델에 **혼란**을 줄 수 있음
  - ✓ 그렇기 때문에, negative reward 의 변화와 개수에 따라서 **유의미한 성능 변화가 이루어지지 않았다**고 추측됨

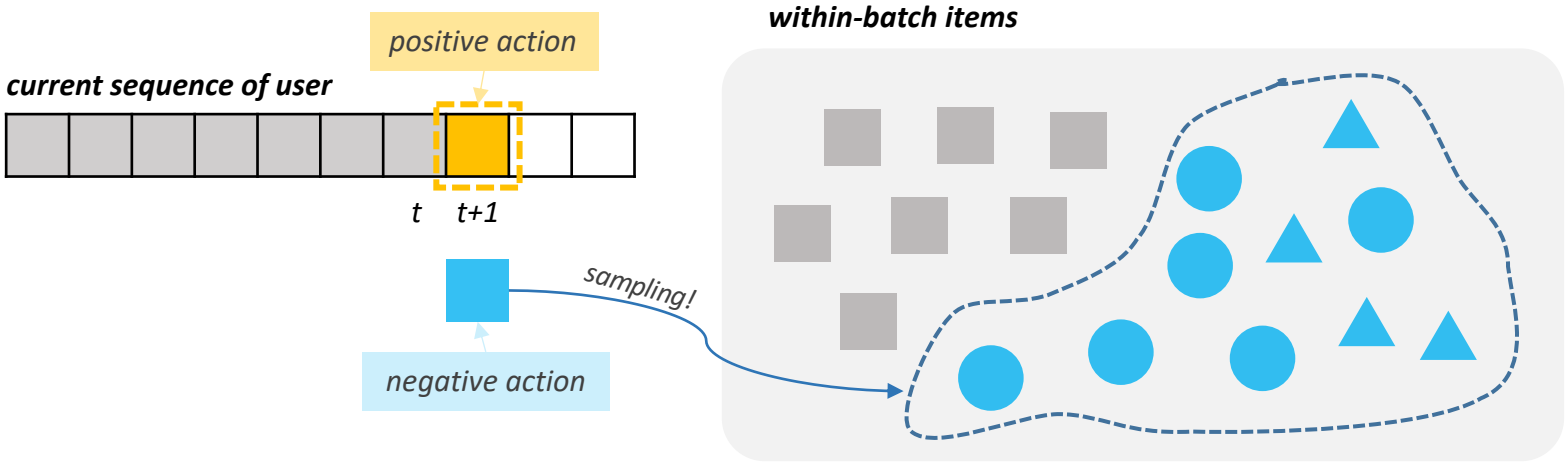
2. Proposed Method

④ *Batched Negative Sampling*

- **현재 상태를 고려한 Negative sampling 방식 도입**

✓ 사용자 정보를 고려하여, 예측하고자 하는 state의 **사용자가 상호작용했던 아이템을 배치 내에서 등장하는 아이템**에서 **제외**한 뒤, random sampling 하는 방식으로 진행

computational efficiency ↑



3. Experiments

Evaluation metrics

1) **HR (Hit Ratio):** measuring whether the ground truth item is in the top-k recommendation list.

$$HR(click) = \frac{\#hits\ among\ clicks}{\#clicks\ in\ test}$$

2) **nDCG (Normalized Discounted Cumulative Gain):** rank sensitive metric which assign higher score to top position in the recommendation list.

$$nDCG = \frac{DCG}{IDCG}, \quad DCG = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \quad IDCG = \sum_{i=1}^{k} \frac{rel_i^{opt}}{\log_2(i+1)}$$

3) **Coverage**

$$Coverage@K = \frac{|\cup_{s \subset S} L_K(s)|}{|I|}, \quad L_K(s) = [i_1, i_2, \ldots, i_k]$$ ← *Top-K recommendation list.*

4) **Total reward**

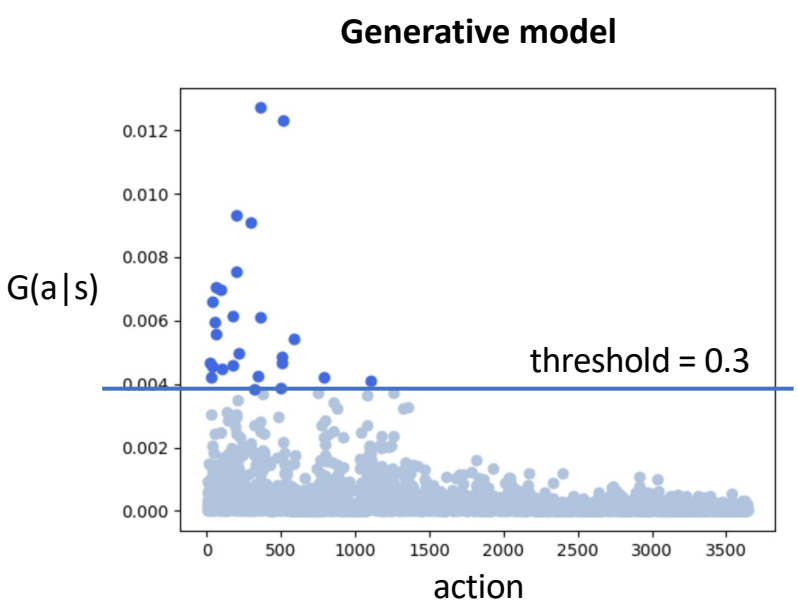
3. Experiments

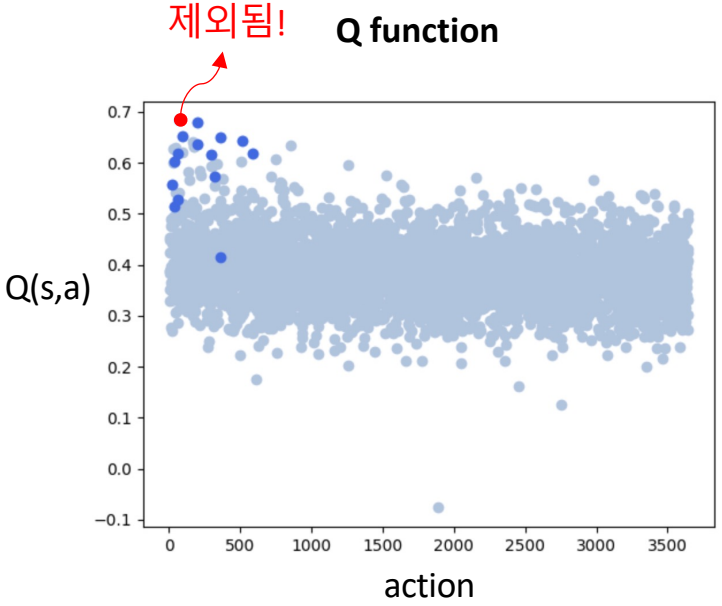
Performance comparison

	LastFM				Beauty			
Models	**HR@20**	**nDCG@20**	**Coverage**	**Total Rewards**	**HR@20**	**nDCG@20**	**Coverage**	**Total Rewards**
GRU	17.533	8.085	99.451	-	11.008	5.340	80.101	-
GRU+SNQN	17.990	8.288	**99.369**	827	11.620	5.654	**76.985**	2298
GRU+SNQN+ours	**23.798**	**10.251**	72.902	**1094**	**14.219**	**6.448**	50.269	**2812**
GRU+SA2C	13.300	5.999	51.509	835	10.944	5.371	**39.972**	2194
GRU+SA2C+ours	**15.929**	**6.913**	**65.579**	**1000**	**12.131**	**5.774**	36.518	**2432**
SASRec	22.602	10.827	99.013	-	13.395	6.872	69.358	2649
SASRec+SNQN	21.296	10.094	**96.654**	979	12.540	5.610	17.420	2480
SASRec+SNQN+ours	**25.582**	**11.000**	82.913	**1176**	**15.802**	**7.389**	**48.318**	**3125**
SASRec+SA2C	4.699	1.715	0.576	295	9.852	4.507	5.239	1921
SASRec+SA2C+ours	**14.161**	**5.826**	**22.655**	**889**	**12.266**	**5.689**	**9.131**	**2459**

4. Experiments

Analysis - BCQ





3. Experiments

Analysis – time efficiency

○ Batched Negative Sampling 기법으로 인한 time complexity 감소

```python
def neg_sampler(self, actions, n_samples):
    negatives = []
    for idx in range(n_samples):
        negative_list = []
        for i in range(self.n_neg):
            neg = np.random.randint(self.item_num)
            while neg == actions[idx]:
                neg = np.random.randint(self.item_num)
            negative_list.append(neg)
        negatives.append(negative_list)
    return torch.LongTensor(np.array(negatives)).to(self.device)
```

$O(n^2)$

```
------------------------------------------------------------
Epoch:  0
start training:  2022-12-03 22:21:59.487948
[40/40] Loss: 8.671  Time: 43.77
                Total Loss:     349.005
Metric          HR@10   NDCG@10         Cov@10  Total Reward
Value           1.147   0.448           1.783   72.0
Metric          HR@20   NDCG@20         Cov@20  Total Reward
Value           1.752   0.603           2.825   110.0
Time elapse : 0.06936383247375488
------------------------------------------------------------
```

```python
def neg_sampler_new(self, states, n_samples):
    states = states.to('cpu').numpy()
    negatives = []
    batch_items = np.unique(states)
    for idx in range(n_samples):
        negatives_list = np.random.choice(np.setdiff1d(batch_items, states[idx]),
                                          size=self.n_neg, replace=False)
        negatives.append(negatives_list)
    return torch.LongTensor(np.array(negatives)).to(self.device)
```

$O(n) + O(n)$

```
Epoch:  0
start training:  2022-12-03 22:20:17.817244
[40/40] Loss: 8.956  Time: 7.77
                Total Loss:     348.052
Metric          HR@10   NDCG@10         Cov@10  Total Reward
Value           0.924   0.392           1.920   58.0
Metric          HR@20   NDCG@20         Cov@20  Total Reward
Value           2.103   0.683           2.962   132.0
Time elapse : 0.06982231140136719
```

3. Experiments

Ablation study

Encoder = GRU	LastFM			
Models	**HR@20**	**nDCG@20**	**Coverage**	**Total Rewards**
SNQN	17.533	8.085	99.451	806
SNQN+feature	22.188	9.236	89.084	1020
SNQN+feature+BCQ	22.101	9.520	90.620	1016
SNQN+feature+NS	22.710	**9.970**	**90.784**	1044
SNQN+feature+Normalize	**22.972**	9.587	62.095	**1056**
SNQN+feature+BCQ+NS+Normalize	21.209	8.850	59.517	975
SA2C	13.300	5.999	51.509	330
SA2C+feature	14.702	6.311	37.740	923
SA2C+feature+BCQ	**15.929**	**6.913**	**65.579**	**1000**
SA2C+feature+NS	13.523	5.648	42.320	849
SA2C+feature+Normalize	10.752	4.134	23.560	675
SA2C+feature+BCQ+NS+Normalize	12.711	5.279	48.246	798

3. Experiments

Hyperparameter study

Models	# of negative samples	HR@20	nDCG@20	Coverage	negative reward	HR@20	nDCG@20	Coverage
SNQN + ours	1	**24.799**	**10.595**	66.237	0	**23.798**	**10.251**	**72.902**
	5	22.698	9.763	60.943	-0.5	22.254	9.378	53.236
	10	23.798	10.251	**72.902**	-1.0	16.511	6.738	18.788
	50	19.230	8.045	59.956	-1.5	16.402	6.672	15.332
	100	17.903	7.460	46.654	-2.0	6.178	2.404	3.62
SA2C + ours	**1**	**19.114**	**8.373**	**98.875**	0	**15.929**	**6.913**	65.597
	5	15.945	7.089	93.993	-0.5	15.642	6.724	**79.100**
	10	15.929	6.913	65.579	-1.0	14.447	6.181	65.836
	50	5.989	2.268	2.276	-1.5	13.221	5.489	50.302
	100	3.918	1.427	27.948	-2.0	13.141	5.425	60.121

3. Experiments

Hyperparameter study

Models	γ for Q loss	HR@20	nDCG@20	Coverage	threshold of BCQ	HR@20	nDCG@20	Coverage
SNQN + ours	0.5	**24.148**	**10.403**	89.468	0.1	**24.578**	10.421	90.291
	1	22.101	9.520	**90.620**	**0.3**	22.101	9.520	**90.620**
	5	16.614	6.769	47.614	0.5	24.466	**10.480**	89.834
	10	11.054	4.241	18.596	0.7	24.275	10.464	90.181
	20	4.603	1.751	3.127	0.9	15.355	6.510	25.891
SA2C + ours	0.5	**16.215**	**7.156**	**94.899**	0.1	15.084	6.505	65.935
	1	15.929	6.913	65.579	**0.3**	15.929	**6.913**	65.579
	5	9.286	3.688	9.901	0.5	**16.343**	6.979	64.948
	10	4.858	1.839	0.823	0.7	15.913	6.946	**72.161**
	20	4.763	1.752	1.481	0.9	16.056	7.049	67.087

4. Conclusion

- Sequential recommendation에서 강화학습을 적용하여 사용자의 추천 만족도를 높일 수 있는 모델을 제안
- 추천에서 아이템의 특징을 포함시키는 것이 성능 향상에 도움이 되는 것을 실험적으로 증명
- BCQ, Batched Negative Sampling 을 통해 거대한 action space (전체 아이템)을 좁혀서 효율적인 학습이 이루어질 수 있도록 함
- 추천의 정확도 뿐만 아니라, 인기도 편향을 완화하여 추천의 다양성을 향상시키기 위한 normalized 기법 적용

Q&A

1. 6 페이지에서 action은 일정 길이의 아이템들을 추천하는 건가요? 가령 top-20 조건이라면, 20개의 아이템을 한 번에 출력하는 건가요? 혹은 20번에 걸쳐서 하나씩 출력하나요?

기준이 되는 state에 대해서 20개의 아이템을 한 번에 출력하는 방식으로 진행하였습니다.

2. Reward 는 어떻게 설계 하였나요?

Positive action에 대해서는 reward 를 1로 설정하였고, negative action에 대해서는 reward를 0으로 설정하였습니다. 기존 계획 부분에서는 coverage 부분을 담아서 사용자의 추천 만족도를 높일 수 있는 종합적인 reward를 설계하고자 하였으나, 그 부분이 미흡하게 되어서 아쉽지만, 추후에 시도해보고 합니다.

3. Coverage 평가 항목은 어떻게 계산되는 것인가요?

Coverage는 발표 자료에서도 말씀 드렸던 것처럼, 아래 식과 같이 계산할 수 있습니다. 모델이 예측한 추천 리스트들에서 겹치지 않는 고유한 아이템의 개수를 전체 아이템 개수로 나눕니다.

$$Coverage@K = \frac{|\cup_{s \subset S} L_K(s)|}{|I|}, \quad L_K(s) = [i_1, i_2, \ldots, i_k] \leftarrow \textit{Top-K recommendation list.}$$

Q&A

4. 21 페이지 ablation 분석에서 feature 항목이 item에 대한 GNN 임베딩 한 것을 의미하나요? feature 항목이 가장 주요한 영향을 미치는 것으로 보이네요. ablation 분석으로 SASRec 포함되는 경우는 일부로 제외 한 것인가요?

네 맞습니다. 저희가 분석하기에도 아이템에 대한 특성 정보를 사용하는 것이 모델 학습에 더 도움이 되는 것으로 보았습니다. 추가로 SASRec 모델과 관련된 부분은 아래 표와 같습니다.

Encoder = SASRec	k=20			
Models	**HR**	**NDCG**	**Coverage**	**Total Reward**
SA2C	4.699	1.715	0.576	295
SA2C+feature	8.872	3.474	4.526	557
SA2C+feature+BCQ	4.667	1.73	0.603	293
SA2C+feature+NS	**14.161**	**5.826**	**22.655**	**889**
SA2C+feature+Normalization	4.667	1.712	0.549	293
SA2C+feature+BCQ+NS+Normalization	11.962	4.915	13.11	751
SNQN	13.157	6.547	32.113	2602
SNQN+feature	15.165	6.669	25.362	2999
SNQN+feature+BCQ	15.266	6.922	37.055	3019
SNQN+feature+NS	13.612	5.93	21.378	2692
SNQN+feature+Normalization	14.77	6.475	24.122	2921
SNQN+feature+BCQ+Normalization	**15.802**	**7.389**	**48.318**	**3125**

Comments

negative sampling에 대한 의미/필요성이 아직 확실히 이해되지 못한 점이 아쉽군요. 레퍼런스 논문에서는 BCQ 같은 offline RL기법을 사용하지 않았기에 overestimation 문제가 크게 발생할 수 있는 것을 negative sampling 으로 도움을 받은게 아닐까 싶습니다. 반면, BCQ를 사용하면 이미 distribution shift 같은 이슈를 이미 줄였기 때문에 negative sampling 효과가 줄어들지 않았을까요. 그리고 coverage 같은 성능을 높이기 위해 RL 학습시에 reward 설계 등으로 반영해보면 좋을 듯 합니다.

THANK YOU.

감사합니다 .