SURGE

Tags	
Column	
Column 1	

Sequential recommendation :users' historical behaviors을 leverage하여 고객의 다음 interaction을 predict.

main challenges

1) rich historical sequences에서의 user behaviors는 종종 implicit 하고 noisy preference signals. > users' actual preferences 를 sufficiently reflect 못함

2) users' dynamic preferences는 종종 change rapidly하기 때문에 historical sequence 에서 user patterns을 capture 하는 것은 어려움

- SURGE 제안(short for SeqUential Recommendation with Graph neural nEtworks) : long-term user behaviors into clusters와 the graph by re-constructing loose item sequences into tight item-item interest graphs based on metric learning의 different types of preferences를 integrates.
    - 이는 users' core interests를 explicitly distinguish하는데 도움을 줄 것(interest graph에서 forming dense clusters 를 통해)
    - cluster-aware and query-aware graph convolutional propagation 과 graph pooling on the constructed graph를 perform
    - noisy user behavior sequences에서도 dynamically fuses 와 extracts users' current activated core interests .
    - public과 proprietary industrial datasets에서 실험. > SOTA 달성

1. introduction

- sequential recommandation : news, video, advertisements, 등과 같은 modern online information systems 사용.
- traditional recommendation tasks : static fashion의 model user preferences만을 반영,

- sequential recommendation : user's evolved 와 dynamic preferences를 capturing 가능

(예) user may prefer to watch soccer news only during the period of World Cup, which can be
regarded as a kind of **short-term preference**.

- fastchanging short-term preferences, 3개의 관점에서 접근

1) early efforts : human designed rules 또는 attention mechanism to assign time-decaying
weights to historically interacted items

2) leverages recurrent neural networks to summarize the behavioral sequences : capturing users' dynamic interests에서 suffer from the short-term bottleneck. difficulty of modeling long-range dependencies동안

3) model long-term and short-term interests to avoid forgetting longterm interests, but the division and integration of long/short-term interests are still challenging : aforementioned works commonly concentrate more on user behaviors of recent times, 그리고 are not capable of fully mining older behavior-sequences to accurately estimate their current interests.

- User behaviors in long sequences reflect implicit and noisy preference signals
- User preferences are always drifting over time due to their diversity.
    - based method with graph convolutional networks 는 implicit preference signals를 extract
    - dynamic graph pooling : dynamics of preferences를 capture

1) loose item sequence 를 tight item-item graph 와 design a attentive graph convolutional network 으로 convert > gather weak signals 에서 strong ones하게 되면 (accurately reflect user preferences 가능)

- new perspective으로 sequential recommendation를 접근. implicit-signal behaviors 와 fast-changing preferences를 고려한
- user behaviors의 implicit signals를 aggregate 하여 explicit ones으로 변환. > constructed item-item interest graphs의 designing graph neural network-based models > recommendation에 대해 activated core preferences를 reserving 하고 dynamic-pooling for filtering 를 design
- 실제 applications의 two large-scale datasets를 실험하여 conduct extensive. experimental results는 state-of-the-art 라는significant performance

improvements 를 보여줌. long behavioral sequences effectively 하고 efficiently하게 model 가능

2. Problem formultation

- $x \in \mathcal{X}$ item, item set
- sequential interaction sequence with items $\{x_1, x_2, \cdots, x_n\}$
- n : the number of interactions
- x_i : i-th item that the user has interacted with
- x_{n+1} : predict the next item. user's preferences.
- input : The interaction history for each user $\{x_1, x_2, \cdots, x_n\}$
- output : A recommendation model that estimates the probability that a user with interaction history $\{x_1, x_2, \cdots, x_n\}$ will interact with the target item x_t at the (n + 1)-th step

3. Methology

1) interest Graph Construction : loose item sequences를 tight item-item interest graphs based on metric learning를 통해 re-constructing. explicitly integrate 하고 long-term user behaviors에서 distinguish different types preferences.
2) Interest-fusion Graph Convolutional Layer : constructed interest graph dynamically에서 graph convolution propagation. fuses the user's interests, strengthening important behaviors, and weakening noise behaviors.
3) Interest-extraction Graph Pooling Layer : 다른 시점에서, users' different preferences를 고려, dynamic graph pooling operation은 dynamically activated core preferences를 adaptively conducted위해 수행.

4) Prediction Layer : pooled graphs 후에 reduced sequences로 flattened, model the evolution of the enhanced interest signals하고 predict the next item that the user has high
probability to interact with.

3.1 Interest Graph Construction

- 두 item 간의 co-occurrence relationship는 reasonable construction criterion이지만, 문제는 co-occurrence relationship의 sparseness가 각 사용자에 대해 연결된 그래프를 생성하기에 충분하지 않음. > metric learning : automatically construct graph structures for each interaction
  sequence to explore the distribution of its interests.

3.1.1. Raw graph construction

undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}\}$ for each iteration sequence, $\mathcal{E}$: set of graph edeg to learn

$A \in \mathbb{R}^{n \times n}$: corresponding adjacency matrix. Each vertex $v \in \mathcal{V}$ with $|\mathcal{V}| = n$ interacted item

- 목표 : adjacency matrix A의 학습. 각 edge $(i, j, A_{i,j}) \in \mathcal{E}$: item i 와 연관된 item j
- each user's interaction history를 graph로 표현함으로써, user의 core하고 peripheral interests를 쉽게 distinguish
- core interest node는 유사한 관심사를 더 많이 연결하기 때문에 peripheral interest node보다 정도가 높고, 유사한 관심사의 빈도가 높을수록 subgraph가 더 촘촘하고 커짐.
- 이러한 방식으로, priori framework가 구성. (neighbor nodes들간 유사, dense subgraphs는 core interests of users.

3.1.2 Node similarity metric learning

- priori graph가 필요하여 graph learning problem가 node similarity metric learning 문제로 변환.
- downstream recommendation task와 함께 훈련.
- 이 graph construction method 장점 : general, easy to implement, and able to perfectly cope with inductive learning (with new items during testing)
- Metric learning : kernel-based 와 attention-based methods으로 분류 가능 > improve expressiveness 와 acceptable complexit 학습할 수 있어야 함
  - 1) kernel-based : cosine distance, Euclidean distance, Mahalanobis distance 가 포함
  - 2) attention-based methods
- weighted cosine similarity

$$M_{ij} = \cos(\vec{\mathbf{w}} \odot \vec{h}_i, \vec{\mathbf{w}} \odot \vec{h}_j), \qquad (1)$$

$\odot$: Hadamard product, $\vec{w}$: $\vec{h_i}$ 와 $\vec{h_j}$ 의 item embeddings의 다른 dimensions 을 adaptively highlight하기 위한 trainable weight vector

- expressive power를 높이고 learning process를 안정시키기 위해, similarity metric function을 multi-head metric으로 확장 가능[5, 25].
- 특히, 위의 similarity metric function를 사용하여 ϕ (the number of heads) independent similarity matrices (각각 one representing one perspective)를 계산하고 이들의 평균을 final similarity로 취하기 위해 ϕ (the number of heads) 가중치 벡터를 사용

$$M_{ij}^{\delta} = \cos(\vec{\mathbf{w}}_{\delta} \odot \vec{h}_i, \vec{\mathbf{w}}_{\delta} \odot \vec{h}_j), \quad M_{ij} = \frac{1}{\delta} \sum_{\delta=1}^{\phi} M_{ij}^{\delta}, \qquad (2)$$

M_{ij}^{δ} : δ head 에 대한 두 item embedding $\vec{h_i}$ 와 $\vec{h_j}$ 간 similarity metric. 각 head는 different perspective of semantics를 capture

3.1.3 Graph sparsification via ε-sparseness.

- adjacency matrix elements는 음수가 아니어야 하지만 [-1, 1] 사이의 메트릭 범위에서 계산한 코사인 값 M_{ij}. Simply normalizing는 graph sparsity에 어떠한 제약도 가하지 않으며 fully connected adjacency matrix를 산출 가능.
- 이는 계산 비용이 많이 들고 noise(즉, unimportant edge)를 도입할 수 있으며, subsequent graph구성이 그래프의 가장 관련성이 높은 측면에 초점을 맞출 수 없을 만큼 충분히 희소하지 않음.
- 가장 중요한 연결을 가진 노드 쌍만 고려하여 M에서 symmetric sparse non-negative adjacency matrix A를 추출. 임계값의 hyper parameter를 extraction하게 만들고, graph's sparsity distribution를 파괴하지 않기 위해 entire graph의 relative ranking strategy 를 채택. 특히, 우리는 음이 아닌 임계값보다 작은 M의 element를 mask(0으로 설정)하며, M의 metric 값을 순위화.

$$A_{ij} = \begin{cases} 1, & M_{ij} >= \mathbf{Rank}_{\varepsilon n^2}(M); \\ 0, & \text{otherwise}; \end{cases} \qquad (3)$$

$Rank_{\varepsilon n^2}(M)$: Metric Matrix M에서 εn^2-th largest value. n : the number of nodes, ε : overall sparsity of the generated graph.

- entire graph의 absolute threshold strategy와 relative ranking strategy of the node neighborhood 는 다르다.
  - entire graph의 absolute threshold strategy : adjacency matrix에서 smaller elements를 제거하기 위한 absolute threshold를 설정. 임베딩이 지속적으로 업데이트됨에 따라 hyperparameter가 잘못 설정되면 metric value distribution도 변경되며 그래프를 생성하거나 전체 그래프를 생성하지 못할 수 있음.
  - relative ranking strategy of the node neighborhood : adjacency matrix에서 각 행의 fixed number of maximum values의 indices 를 반환, 이는 generated graph가 same degree를 갖도록 함. uniform sparse distribution를 적용하면 downstream GCN은 graph의 dense or sparse structure information을 완전히 활용 불가

3.2 Interest-fusion Graph Convolutional Layer

- separate diverse interests를 구분하는 learnable interest graphs. 핵심 관심사와 주변 관심사는 각각 큰 cluster와 작은 cluster를 형성하고, 다른 유형의 관심사는 다른 cluster를 형성. 또한 사용자 선호도를 정확하게 반영할 수 있는 강한 신호를 수집하기 위해 구성된 그래프에 정보를 종합.

3.2.1 Interest fusion via graph attentive convolution

- 정보 수집 중에 사용자의 핵심 관심사(즉, cluster center에 위치한 item)와 쿼리 관심사(즉, 현재 대상 항목)를 인식할 수 있는 cluster- and query-aware graph attentive convolutional layer를 제안.
- input : node embedding matrix $\{\vec{h_1}, \vec{h_2}, \cdots, \vec{h_n}\}, \vec{h_i} \in \mathbb{R}^d$, n : 노드의 개수(즉, 사용자 상호 interaction sequence length) 이며, d는 각 노드에 임베딩되는 차원.
- Layer 는 $\{\vec{h_1}', \vec{h_2}', \cdots, \vec{h_n}'\}, \vec{h_i}' \in \mathbb{R}^{d'}$ 잠재적으로 다른 차원 d'의 출력으로 새로운 노드 embedding node matrix 를 생성. applying a residual connection and a nonlinearity function을 적용한 후 every node에 대한 output embeddings

$$\vec{h}_i' = \sigma\left(\mathbf{W_a} \cdot \mathbf{Aggregate}\left(E_{ij} * \vec{h}_j | j \in \mathcal{N}_i\right) + \vec{h}_i\right). \qquad (4)$$

aggregation function : Mean, Sum, Max, GRU, etc.

- attention mechanism의 learning process를 안정화시키기 위해 [25, 26]과 유사한 multi-head attention을 사용. 정확히는, ϕ : execute the above transformation하는 independent attention mechanisms . 그 임베딩은 다음과 같은 출력 표현으로 표현.

$$\vec{h}_i' = \Big\Vert_{\delta=1}^{\phi} \sigma\left(\mathbf{W_a}^{\delta} \cdot \mathbf{Aggregate}\left(E_{ij}^{\delta} * \vec{h}_j | j \in \mathcal{N}_i\right) + \vec{h}_i\right), \qquad (5)$$

$||$: concat. E_{ij}^{δ} : normalized attention coefficients obtained by the δ-th attention head, W_a^{δ} : corresponding linear transformation's weight matrix.

최종 output $\vec{h}'$ 은 각 노드에 대해 (d'가 아닌) dimension embed 에 해당한다는 점에 유의

3.2.2 Cluster- and query-aware attention

- interest 통합 시 important signals를 강화하고 noise signals를 약화시키기 위해 cluster and query-aware attention mechanism을 제안. attention coefficients를 사용하여 message passing 과정에서 edge information에 weights를 재분배.

1) target node v_i의 neighborhood가 cluster를 형성하고 그래프에서 target node를 클러스
터 $c(v_i)$의 중간체로 간주한다고 가정.

- target node v_i의 k-hop neighborhood을 cluster의 receptive field로 정의.
- $\vec{h}_{i_c}$cluster 내 모든 nodes' embedding 평균 값은 cluster의 평균 정보를 표현. target node가 cluster의 중심인지 확인하기 위해 target node embedding 및 해당 cluster embedding이 following attention score를 계산하는 데 사용

$$\alpha_i = \mathbf{Attention}_c(\mathbf{W_c}\vec{h}_i \parallel \vec{h}_{i_c} \parallel \mathbf{W_c}\vec{h}_i \odot \vec{h}_{i_c}), \qquad (6)$$

W_c : transformation matrix, $||$: concatenation operation, $\odot$: Hadamard product, $Attention_c$:

two-layers feedforward neural network with the LeakyReLU as activation function

2) downstream dynamic pooling method을 제공, user interest's independent evolution의 target interest 를 학습하기 위해서는 source node embedding $\vec{h}_j$ 과 target item embedding $\vec{h}_t$간의

- 상관 관계도 고려해야함. source node 가 query item과 연관있으면, target node에 대한 aggregation 에서 weight 가 더 중요하며, 그 반대의 경우도 마찬가지.

- relevant behaviors 만이 final prediction에 역할을 할 수 있으므로 relevant information만 보관하고 irrelevant information는 aggregation중 폐기

$$\beta_j = \mathbf{Attention}_q(\mathbf{W_q}\vec{h}_j \parallel \vec{h}_t \parallel \mathbf{W_q}\vec{h}_j \odot \vec{h}_t), \qquad (7)$$

W_q : transformation matrix, $||$: concatenation operation, $\odot$: Hadamard product

- attention mechanism Attention$q$ 는 LeakyReLU nonlinearity를 적용한 a two-layers feedforward neural network
- cluster와 query의 factors를 동시에 고려하기 위해 additive attention mechanism[1]을 따름. target node's cluster score와 source node's query score를 소스 노드 j 에 대한 update weight로
  합산. 서로 다른 nodes에서 coefficients를 쉽게 비교할 수 있도록 softmax function를 사용하여
  j의 모든 선택에서 정규화. attention coefficients E_{ij}는 다음과 같이 계산.

$$E_{ij} = \mathrm{softmax}_j(\alpha_i + \beta_j) = \frac{\exp(\alpha_i + \beta_j)}{\sum_{k \in N_i} \exp(\alpha_i + \beta_k)}, \qquad (8)$$

node i의 neighborhood N_i 는 node i itself를 포함. self-loop propagation (when j equals i) 를 포함하는 context 에서 α_i : target node가 받을 수 있는 information 양 controls , β_j : source node가 보낼 수 있는 information 양 controls

3.3 Interest-extraction Graph Pooling Layer

- explicit interest signals에 대한 implicit interest signals의 fusion은 interest graph에서 information aggregation를 수행하여 완료.
- 그래프 풀링 방법[17, 22, 37]을 사용하여 fusion information 를 추가로 추출. CNN의 pooling에서 feature map의 downsamling과 유사하게, graph pooling은 graph 를 합리적으로 downsize하는 것을 목표. structured graph structure의 조율을 통해 loose interest이 tight interest로 전환되고 분포가 유지.

3.3.1 Interest extraction via graph pooling.

- pooled graph를 얻으려면 cluster assignment matrix이 필요[22, 37]. soft cluster assignment matrix $S \in \mathbb{R}^{n \times m}$가 존재한다고 가정하면 node information를 cluster information로 pooling 가능.
- m : 풀링의 정도를 반영하는 사전 정의된 model hyperparameter($m < n$)

- node embeddings $\{\vec{h_1}, \vec{h_2}, \cdots, \vec{h_n}\}$ 와 raw graph의 node core $\{\gamma_1, \gamma_2, \cdots, \gamma_n\}$이 주어지면 cluster embeddings 및 coarsened graph를 생성 가능

$$\{\vec{h}_1^*, \vec{h}_2^*, \ldots, \vec{h}_m^*\} = S^T\{\vec{h}_1', \vec{h}_2', \ldots, \vec{h}_n'\}, \quad (9)$$

$$\{\gamma_1^*, \gamma_2^*, \ldots, \gamma_m^*\} = S^T\{\gamma_1, \gamma_2, \ldots, \gamma_n\}, \quad (10)$$

γ_i : β_i 에 softmax를 적용해 얻음. i-th node의 importance score를 표현

- assignment matrix S의 각 행은 n node중 하나에 해당하고, 각 열은 m 클러스터 중 하나에 해당.
  - 각 node를 해당 클러스터에 soft assignment 가능.
  - cluster assignment S에 따라 node embeddings과 scores를 집계하여 각 clusters에 대한 새로운 embeddings 과 scores를 생성.
- 다음으로, node 에 대한 differentiable soft clusters assignment S을 학습하는 방법에 대해 논의
  . assignment matrix를 생성하기 위해 GNN 아키텍처[37]를 사용.
  - assignment mapping의 probability matrix는 adjacency matrix와 node embedding을 기반으로 standard message passing 및 softmax function를 통해 얻음.

$$S_{i:} = \text{softmax}\left(\mathbf{W_p} \cdot \textbf{Aggregate}\left(A_{ij} * \vec{h}_j' | j \in \mathcal{N}_i\right)\right), \quad (11)$$

W_p : maximum number of clusters m,

- 소프트맥스 함수는 i번째 노드가 m 클러스터 중 하나로 분할될 확률을 구하는 데 사용
- 클러스터 간의 연결을 보장하기 위해 $S^T AS$을 수행하여 pooled graph의 adjacency matrix A^*을 얻을 수 있다는 점에 주목.
- 방정식의 반복은 관심 있는 hierarchical compression을 달성하기 위해 multi-layer pooling을 수행할 수 있음.

3.3.2 Assignment regularization

- downstream recommendation task인 gradient signal만 사용하여 cluster assignment matrix S를훈련시키기는 어려움.

- non-convex optimization문제는 early training stage에서 local optimum 상태에 빠지기 쉬움[37].
- 각 node embedding의 relative position $\{\vec{h_1}', \vec{h_2}', \cdots, \vec{h_n}'\}$ 에 포함된 각 node 의 relative position는 interaction의 시간 순서와 일치.
- pooled cluster embedding matrix $\{\vec{h_1}^*, \vec{h_2}^*, \cdots, \vec{h_n}^*\}$ 에서 사용자의 관심을 반영하는 클러스터 사이의 시간 순서는 보장되기 어려움 > 정규화 조건을 제시

1) Same mapping regularization

- 연결 강도가 더 높은 두 node가 동일한 cluster에 더 쉽게 매핑되도록 하기 위해 첫 번째 정규화

$$L_{\mathrm{M}} = \|A, SS^T\|_F, \qquad (12)$$

$||\cdot||_F$: Frobenius norm. adjacency 매트릭스 A의 각 element는 두 node 사이의 연결 강도를 나타내며, SS^T의 각 원소는 두 node가 동일한 클러스터에 분할될 확률

2) Single affiliation regularization

- 각 클러스터의 연관성을 명확하게 정의하기 위해, 각 행을 $S_{i:}$로 만듬. assignment matrix에서 entropy를 정규화하여 one-hot vector를 approch

$$L_{\mathrm{A}} = \frac{1}{n}\sum_{i=1}^{n} H\left(S_{i:}\right), \qquad (13)$$

$H(\cdot)$: mapping distribution의 불연속성을 줄일 수 있는 entropy function, optimal situation : i 번째 node가 하나의 cluster에만 mappling, $H(S_{i:})$ 가 0

3) Relative position regularization

- downstream interest evolution modeling을 위해 pooling 전후의 사용자 interest 의 시간 순서를 유지해야함. pooled cluster embedding matrix $\{\vec{h_1}^*, \vec{h_2}^*, \cdots, \vec{h_n}^*\}$ 에서 인덱스를
  스왑하는 작업은 구별 불가.
- pooling 동안 clusters 사이의 시간적 순서를 보장하기 위해 position regulazation을 설계.

$$L_{\mathrm{P}} = \|P_n S, P_m\|_2, \qquad (14)$$

P : position encoding vector $\{1, 2, \cdots, n\}$이고, P_m : position encoding vector $\{1, 2, \cdots, m\}$. L2 norm을 최소화하면 S에서 0이 아닌 element의 position이 main diagonal elements에 더 가까워
짐. Intuitively, 원래 시퀀스에 front position이 있는 node의 경우 assigned cluster의 position index는 front에 있는 경향.

3.3.3 Graph readout

- user's stronger interest signal를 나타내는 tightly coarsened graph $\mathcal{G}$를 얻음. 동시에, raw graph $\mathcal{G}$에 대한 weighted readout을 수행하여 각 node의 importance를 제한. propagation layer의 forward computation에서 graph-level representation $\vec{h}_g$을 생성한 후 모든 노드임베딩을 집계.

$$\vec{h}_g = \mathbf{Readout}(\{\gamma_i * \vec{h}'_i, i \in \mathcal{G}\}), \qquad (15)$$

weight는 pooling 전 각 노드의 score γ_i이고, Readout 함수는 Mean, Sum, Max, etc.과 같은
함수가 될 수 있음.

단순 합계 함수를 사용하여 permutation invariant을 보장. 이 그래프 수준 표현을 pooling layer에서 각 cluster's의 information을 추출하는 final prediction layer에 제공

3.4 Prediction Layer

3.4.1 Interest evolution modeling

- external environment과 internal cognition의 joint influence 아래 users의 핵심 관심사는 지속적으로 진화. users는 한 때 다양한 스포츠에 관심을 갖게 되고 또 다른 때는 책이 필요할 수 있음.
- 그러나 readout operation을 사용하는 것은 core interests 사이의 evolution를 고려하지 않아 시간 순서의 bias을 야기할 것.
- 관심사에 대한 최종표현을 좀 더 상대적인 역사적 정보로 공급하기 위해서는 관심사 간의 연대적 관계를 고려하는 것도 중요.
- relative position regularization의 이점을 활용하여 pooled cluster embedding matrix 는 users의 intereset의 시간 순서를 유지, 향상된 관심 신호로 pooled graph를

reduced sequence로 평탄화하는 것과 같음.

- 직관적으로,집중 관심 시퀀스를 모델링하기 위해 알려진 순차적 권장 방법을 사용할 수 있음.
- 단순성을 위해 풀링 방법의 효과를 설명하기 위해 단일 등가 모델을 사용하여 관심의 진화를 모델링

$$\vec{h}_s = \mathbf{AUGRU}(\{\vec{h}_1^*, \vec{h}_2^*, \ldots, \vec{h}_m^*\}). \quad (16)$$

GRU는 RNN의 vanishing gradients problem를 극복하고 LSTM보다 빠름 [11].

- interest extraction layer에 대한 fused interest의 importance weight γ_i^*를 더 잘 활용하기 위해 attentional update gate(AU-GRU)가 있는 GRU를 채택[45]하여 attention mechanism과 GRU를 원활하게 결합.
- AUGRU는importance weight γ_i^*를 사용하여 업데이트 범주의 모든 차원을 척도화하므로 관련 관
  심이 적어 숨겨진 상태에 미치는 영향이 적음.

3.4.2 Prediction

- interest extraction layer와 interest evolution layer의 evolution output의 graph-level representation을 사용자의 현재 관심사로 삼고 target item embedding과 연결.
- concatenated dense representation vector가 주어지면, combination of embeddings을 자동으로 학습하기 위해 완전히 연결된 레이어가 사용.
- 다음 단계에서 항목과 상호 작용하는 사용자의 확률을 추정하기 위해 예측 함수로 2계층 피드 포워드 신경망을 사용하며, 실험 부분의 모든 비교 모델은 이 인기 있는 설계를 공유할 것[39, 45, 46].

$$\hat{y} = \mathbf{Predict}(\vec{h}_s \| \vec{h}_g \| \vec{h}_t \| \vec{h}_g \odot \vec{h}_t). \quad (17)$$

- 실제 업계의 CTR(click-through rate) prediction [45, 46]에 따라 negative log-likelihood function를 loss function으로 사용하고 이 설정을 모든 비교 모델과 공유. 최적화 프로세스는 과적합을 방지하기 위해 L2 정규화 항과 함께 손실 함수를 최소화.

$$L = -\frac{1}{|O|} \sum_{o \in O} (y_o \log \hat{y}_o + (1 - y_o) \log(1 - \hat{y}_o)) + \lambda \|\Theta\|_2, \quad (18)$$

여기서 $\mathcal{O}$는 training set이고 $|\mathcal{O}|$는 training instances 수. Θ는 훈련 가능한 매개변수의 집합, λ : 페널티 강도를 제어. 레이블 $y_o = 1$ positive의 instances, $y_o = 0$은 negative의 instances를 나타냄. $\hat{y_o}$softmax layer 다음에 나오는 network의 출력, 다음 item이 클릭될 확률을 미리 예측한 값.

또한 3.3.2절의 세 가지 정규화 항을 최종 개선 목표 함수에 추가하여 더 나은 성능과 더 해석 가능한 클러스터 할당을 얻음