A Constrained Contrastive Causal Discovery Network for Attributed Dynamic Interaction Graph

Abstract

Temporal graph networks (TGNs) have gained prominence as models for embedding dynamic interactions. But as a temporal model, TGNs are not designed to handle the heterogeneity of the temporal distribution of interactions, which means in different timestamp, even two interactions with the same features can produce very different effect on the memory or on the users and items. This phenomenon may be caused by the overall environment change, which cannot be handled by simply adding a temporal feature. In this paper, we propose a novel model, Constrained Contrastive Causal Discovery Network (C3DN), to address this problem. However, C3DN cannot be used directly to finish the interaction graph task. It's aimed to learn a domain-invariant representation of each user, item and interaction by discovering the causal instruction among users, attributers and items. and then these representations can be used to train the down-stream TGNs. To discovery the causal intruction, C3DN uses SVD-basd Graph Fourier Transform to generate positive and negative sample for further Constrastive learning.

1 Introduction

2 Related Work

3 Preliminary

Notations. We denote the k-th interaction as a vector $\mathbf{x_k} = (u_k, \mathbf{a}_k, i_k, t_k), k = 1, 2, \ldots, n$, where u_k and i_k are the user id and item id respectively, $\mathbf{a}_k = (a_{k,1}, a_{k,2}, \ldots, a_{k,d})$ is the attribute vector with dimention d, and t_k is the timestamp. $\mathbf{X}$ is denoted as the set of all interactions, which we call it interaction matrix. Furthermore, we only consider the discrete timestamp case, i.e. $\{t_k\}_{k=1}^n \subset \{T_1, T_2, \ldots, T_m\}$. We split the total matrix $\mathbf{X}$ into m sub-matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m$, where $\mathbf{X}_i$ contains all the interactions with timestamp T_i. We also denote $\tilde{\mathbf{X}}_\mathbf{i}$ as the time-eliminated version of $\mathbf{X}_i$, namely:

$$\mathbf{X}_i = \begin{pmatrix} u_{k_i} & \mathbf{a}_{k_i} & i_{k_i} & T_i \\ u_{k_i+1} & \mathbf{a}_{k_i+1} & i_{k_i+1} & T_i \\ \vdots & \vdots & \vdots & \vdots \\ u_{k_{i+1}-1} & \mathbf{a}_{k_{i+1}-1} & i_{k_{i+1}-1} & T_i \end{pmatrix}, \qquad \tilde{\mathbf{X}}_\mathbf{i} = \begin{pmatrix} u_{k_i} & \mathbf{a}_{k_i} & i_{k_i} \\ u_{k_i+1} & \mathbf{a}_{k_i+1} & i_{k_i+1} \\ \vdots & \vdots & \vdots \\ u_{k_{i+1}-1} & \mathbf{a}_{k_{i+1}-1} & i_{k_{i+1}-1} \end{pmatrix}$$

The user and item set are denoted as $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ and $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$ respectively. Similarity, based on the timestamp, we can also define the user set $\mathcal{U}_i$ and item set $\mathcal{I}_i$ for each timestamp T_i.

3.1 Causal Graph for Attributers

As the main body of a single interaction, the attributer vector $\mathbf{a}$ totally determines the effect of the interaction. Rather than simply putting the vector $\mathbf{a}$ into the model, we think it's better to consider how this effect is produced. In other words, we want to find the causality among attributers, users and items.

More precisely, we want to find a unweighted directed acyclic graph $\mathbf{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{\mathbf{user}, \mathbf{attributer}_1, \ldots, \mathbf{attributer}_d, \mathbf{item}\}$. It's necessary to mention that unweighted here means the weight of each edge is 0 or 1. However, in this paper we'll loosen the condition such that cyclic graph is allowed.

Here is an example. Suppose Alice bought some math books from Bob's bookstore. Then the attributer vector $\mathbf{a}$ may contain the price of these books, the type of these books, the number of the

books and the publisher of the books. There causality correlation is like the fig **??**. It's obviously that price influence both Alice and Bob, but the type of the books only influence Alice. Also, the number of the books only influence Bob and the publisher infomation accually doesn't matter. In the above toy example, find the causality can give us a better understanding of the attributers.

(*skipped*)

There are three specific reasons for this consideration. First, the attributer vector's dimention a may be very large, but not all of them are equally useful. Finding the causality can reduce the unnecessary dimensions' influence. Also, as we saw above, it can help us recognize the different effect for users and items.

Second, after determining the causality, this skeleton constraint would help increase the model's interpretability

Third, and most importantly, the causality can help us to find the domain-invariant representation of the attributer vector, namely a representation that won't change with the change of the environment. Furthermore, since the characteristics of each user and item, the memory we storaged in GTN model and the node mebedding vectors are all generated and updated through the attributer vector step by step, we actually find the domain-invariant representation of the whole model.

3.2 Constrained Method

There are many methods to find the causality graph. Here we use a adjusted version of the PC algorithm.

The PC algorithm is a classical method to find the causality in a Bayesian network. It's based on the conditional independence test. The algorithm consists of three steps. First, initialize the graph as a complete graph. Second, test the conditional independence of each pair of nodes, and remove the edge if the test result is positive. Third, use some cretiria to determine the other edges. In C3DN, we replace the third step with a GAT network, and only use the first two steps to find a causality pre-skeleton. It's necessary to mention that through this method the causality graph we get finally is not a DAG, but a directed graph with some cycles. However, we still call it a causality graph. We call the first two-step PC algorithm as quick-PC algorithm. The detail is shown in the Appendix.

For convenience, we denote the quick-PC as a operator $\mathcal{QPC}$:

$$\mathcal{QPC} : \mathbb{R}^{N\times p} \to \mathbb{R}^{p\times p}, \quad \mathbf{X} \mapsto \mathbf{G}^{(pre)} \tag{1}$$

3.3 SVD based Graph Fourier Transform

It's well known that the graph Fourier transform is a powerful tool to analyze the graph structure, especially for the graph convolutional network. However, the graph Fourier transform is defined on the symmetric Laplacian matrix, which is not suitable for the causality graph. [] proposed a method to define the graph Fourier transform on the asymmetric Laplacian matrix, which is based on the SVD decomposition:

Given a causality graph G, we denote its adjacent matrix as $\mathbf{A}$ which is asymmetric, and the degree matrix as $\mathbf{D}$. Then the Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. We can get the SVD decomposition of $\mathbf{L}$ as $\mathbf{L} = \mathbf{U}\Sigma\mathbf{V}^T$. Then the graph Fourier transform and its inverse of a signal $\mathbf{x}$ is defined through the following equations:

$$\mathbf{F} := \frac{1}{\sqrt{2}}\begin{pmatrix}\mathbf{U} & \mathbf{U}\\ \mathbf{V} & -\mathbf{V}\end{pmatrix}, \qquad \begin{cases}\mathcal{F}\mathbf{x} := \mathbf{F}^T\begin{pmatrix}\mathbf{x}/\sqrt{2}\\ \mathbf{x}/\sqrt{2}\end{pmatrix} = \begin{pmatrix}(\mathbf{U}^T+\mathbf{V}^T)\mathbf{x}/2\\ (\mathbf{U}^T-\mathbf{V}^T)\mathbf{x}/2\end{pmatrix}\\ \mathcal{F}^{-1}\begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix} := \begin{pmatrix}\frac{\mathbf{I}}{\sqrt{2}} & \frac{\mathbf{I}}{\sqrt{2}}\end{pmatrix}\mathbf{F}\begin{pmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{pmatrix} = \frac{1}{2}(\mathbf{U}(\mathbf{z_1}+\mathbf{z_2})+\mathbf{V}(\mathbf{z_1}-\mathbf{z_2}))\end{cases} \tag{2}$$

The detail of the construction is shown in the Appendix.

More importantly, during the experiments we find that given a signal matrix $\mathbf{X}$,then after the graph Fourier transform based on a causality graph G, doing some pertuation on its low frequency components and transforming it back to time domain, we could get a new signal matrix $\mathbf{X}'$. if G is actually the correct causality graph, then the "difference" between $\mathbf{X}$ and $\mathbf{X}'$ is very small, but if this is not the case for G, then the "difference" would be very large. Here the 'difference' is defined

as the statistic correlation among signal's features. The detail of this experiment will be shown in the Appendix.

This experiment gives us a new way to determine the causality graph. We can use the graph Fourier transform and low frequency pertuation to generate a new signal matrix $\mathbf{X}'$, and disturb the high frequency components to generate a new signal matrix $\mathbf{X}''$. Then our target is to maximize the similarity between $\mathbf{X}$ and $\mathbf{X}'$, and minimize the similarity between $\mathbf{X}$ and $\mathbf{X}''$.

4 Constrained Contrastive Causal Discovery Network

4.1 Constrained Causal Module

In this module we'll determine the causality pre-skeleton using the quick-PC algorithm.

Specifically, for each timestamp T_i, we have the time-eliminated interaction matrix $\tilde{\mathbf{X}}_\mathbf{i}$. If the dimension of the attributer vector is d, then we can use the quick-PC algorithm to find the causality pre-skeleton $\mathcal{QPC}(\tilde{\mathbf{X}}_\mathbf{i}) = \mathbf{G}_i^{(pre)} \in \mathbb{R}^{(d+2)\times(d+2)}$. The reason for calling it a pre-skeleton is that it's not directed. The task of determining the direction of the edges will be done in latter module.

Finally we get a sequence of causality pre-skeletons $\{\mathbf{G}_i^{(pre)}\}_{i=1}^T$.

Absolutely the causality pre-skeletons will vary with the change of the time. However, there must be some common edges among these causality pre-skeletons. The more number of sharing, the more important and essential this edge will be. To measure this property, we introduce an weighted matrix $\mathbf{W}_0$:

$$\mathbf{W}_0 = \frac{\sum_{i=1}^T \mathbf{G}_i^{(pre)}}{T} \tag{3}$$

The closer the value of $\mathbf{W}_0$ to 1, the more important this edge will be. Furthermore, there is one more reason for introducing this weighted matrix: this matrix actually makes a priori knowledge, or a constraint for the causality graph. Then our subsequent training could be more efficient and accurate.

Additionally, we define an initial matrix $\mathbf{M}_0$:

$$\mathbf{M}_0 = \frac{1}{2}\mathrm{Sign}(\mathbf{W}_0), \qquad \mathrm{Sign}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \tag{4}$$

The utility of this matrix will be shown in the later section.

4.2 Feature Embedding Module

In C3DN, we regard each dimension of the attributer vector as a one-dim feature, but such a low dimension is not enough to capture the complex relationship among these nodes. Hence, we introduce a feature embedding module to embed the 1-dim feature into a higher dimension. This module has two parts:

User and Item For each user or item, it has a unique ID, which is a natural one-hot vector. Hence we simply using the embedding layer to embed these one-hot vectors, but with THREE times:

$$\mathbf{Emb}: \quad \mathcal{U} \cup \mathcal{I} \to \mathbb{R}^{3l}, \quad u_k \mapsto \begin{pmatrix} \mathbf{u}_k^{(Q)} \\ \mathbf{u}_k^{(K)} \\ \mathbf{u}_k^{(V)} \end{pmatrix} \tag{5}$$

Attributer For each attribute, its value is a real number, so we need do some adjustment to the above embedding method, namely multiply its value to the embedding vector:

$$\mathbf{Emb}: \mathbb{R}^d \to \mathbb{R}^{3l\times d}, \quad \mathbf{a}_k \mapsto \begin{pmatrix} a_{k,1}\mathbf{a}_1^{(Q)} & a_{k,2}\mathbf{a}_2^{(Q)} & \cdots & a_{k,d}\mathbf{a}_d^{(Q)} \\ a_{k,1}\mathbf{a}_1^{(K)} & a_{k,2}\mathbf{a}_2^{(K)} & \cdots & a_{k,d}\mathbf{a}_d^{(K)} \\ a_{k,1}\mathbf{a}_1^{(V)} & a_{k,2}\mathbf{a}_2^{(V)} & \cdots & a_{k,d}\mathbf{a}_d^{(V)} \end{pmatrix} := \begin{pmatrix} \hat{\mathbf{a}}_\mathbf{k}^{(Q)} \\ \hat{\mathbf{a}}_\mathbf{k}^{(K)} \\ \hat{\mathbf{a}}_\mathbf{k}^{(V)} \end{pmatrix} \tag{6}$$

In summary, we have an embedding matrix with shape $(3l, |\mathcal{U} \cup \mathcal{I}| + d)$ to be trained:

$$\mathbf{E} = \begin{pmatrix} \mathbf{u}_1^{(Q)} & \mathbf{u}_2^{(Q)} & \cdots & \mathbf{u}_{|\mathcal{U}\cup\mathcal{I}|}^{(Q)} & \mathbf{a}_1^{(Q)} & \mathbf{a}_2^{(Q)} & \cdots & \mathbf{a}_d^{(Q)} \\ \mathbf{u}_1^{(K)} & \mathbf{u}_2^{(K)} & \cdots & \mathbf{u}_{||\mathcal{U}\cup\mathcal{I}|}^{(K)} & \mathbf{a}_1^{(K)} & \mathbf{a}_2^{(K)} & \cdots & \mathbf{a}_d^{(K)} \\ \mathbf{u}_1^{(V)} & \mathbf{u}_2^{(V)} & \cdots & \mathbf{u}_{|\mathcal{U}\cup\mathcal{I}|}^{(V)} & \mathbf{a}_1^{(V)} & \mathbf{a}_2^{(V)} & \cdots & \mathbf{a}_d^{(V)} \end{pmatrix} \tag{7}$$

For convenience, here we neglect the notation difference between users and items.

4.3 Causal Recognization Module

For each interaction $\mathbf{x_k} = (u_k, \mathbf{a}_k, i_k, t_k)$, after the feature embedding module, it can be augmented to a high dimension matrix:

$$\mathbf{Emb}(\tilde{\mathbf{x}}_\mathbf{k}) = \begin{pmatrix} \mathbf{u}_k^{(Q)} & \hat{\mathbf{a}}_\mathbf{k}^{(Q)} & \mathbf{i}_k^{(Q)} \\ \mathbf{u}_k^{(K)} & \hat{\mathbf{a}}_\mathbf{k}^{(K)} & \mathbf{i}_k^{(K)} \\ \mathbf{u}_k^{(V)} & \hat{\mathbf{a}}_\mathbf{k}^{(V)} & \mathbf{i}_k^{(V)} \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_k \\ \mathbf{K}_k \\ \mathbf{V}_k \end{pmatrix} \tag{8}$$

For convenience we omit the subscript k in the following discussion.

Note that in Constrained Causal Module we have obtained the causality pre-skeleton $\mathbf{G}^{(pre)}$ for timestamp t and the weighted matrix $\mathbf{W}_0$. Then we can combine the self-attention mechanism with our priori constraint to inference the causality direction:

$$\mathbf{G}^{(infe)} = \sigma(\frac{\mathbf{Q}^T\mathbf{K}}{\sqrt{l}} \odot \mathbf{W}_0) \odot \mathbf{G}^{(pre)} \tag{9}$$

$$\sigma(\mathbf{X}) = \text{sigmoid}(\mathbf{X} - \mathbf{X}^T) \tag{10}$$

Here the sigmoid function is an entry-wise function, and $\odot$ is the entry-wise product.

The first formula is easy to understand, it is just the self-attention mechanism, but with two more constraints: 1) the weight matrix $\mathbf{W}_0$. It's used to measure the importance of each edge. 2) the pre-skeleton $\mathbf{G}^{(pre)}$, which is used to constrain the causality sturcture.

The second formula can be rewritten as:

$$\text{sigmoid}(\mathbf{X} - \mathbf{X}^T) = \frac{1}{1 + \exp(\mathbf{X}^T - \mathbf{X})} = \frac{\exp(\mathbf{X})}{\exp(\mathbf{X}) + \exp(\mathbf{X}^T)} \tag{11}$$

which actually is a softmax function for each symmetric pair. It's used to compare the causality direction between two nodes. Furthermore, one should notice that for each symmetric pair, the sum of these two entries is 1 or 0, which will be used later.

To mentain this property, for each batch we use the mean function for the batch-size dim to aggregate the causality prediction:

$$\mathbf{G}^{(INFE)} = \text{Mean}(\mathbf{G}^{(infe)}) \tag{12}$$

Additionally, now we can define the effcets which attributors have on users and items throuth the causality prediction $\mathbf{G}^{(infe)} \in \mathbb{R}^{(d+2)\times(d+2)}$:

$$\mathbf{User}_{\mathbf{x}_k} = \mathbf{V}_k\mathbf{G}_{\{1\}}^{(infe)}, \qquad \mathbf{Item}_{\mathbf{x}_k} = \mathbf{V}_k\mathbf{G}_{\{d+2\}}^{(infe)} \tag{13}$$

The underscript $\{1\}$ and $\{d+2\}$ means the first and last column of the matrix, respectively.

4.4 Causal Storage Module

Note that the causality prediction we got above is weighted. One can simply transfrom the weight to 0 and 1 by comparing each symmetric pair. However, this will cause the problem that the causality prediction only represent the latest interactions, which exists high uncertainty. Instead, we use the exponential moving average trick to smooth the causality prediction:

$$\mathbf{G}_0^{(initialized)} = \mathbf{M}_0 \tag{14}$$

$$\mathbf{G}_k^{(EMA)} = \alpha\mathbf{G}_k^{(INFE)} + (1-\alpha)\mathbf{G}_{k-1}^{(initialized)} \tag{15}$$

$$\mathbf{G}_k^{(initialized)} = \mathbf{G}_k^{(EMA)} \odot \mathbf{G}_k^{(pre)} + \mathbf{M}_0 - \frac{1}{2}\mathbf{G}_k^{(pre)} \tag{16}$$

$$\mathbf{G}_k = \mathcal{H}_0(\mathbf{G}_k^{(EMA)} - (\mathbf{G}_k^{(EMA)})^T), \quad \mathcal{H}_0(x) = 1_{\{x>0\}} \tag{17}$$

$\alpha \in (0,1)$ is the update rate.

The significance of introducing $\mathbf{G}_k^{(initialized)}$ lies in that after EMA those symmetric pairs which not appears in $\mathbf{G}_k^{(INFE)}$ will lose the property that the sum of these two entries is 1 or 0. And with the help of function $\mathcal{H}_0$, we finally get the directed and unweighted graph $\mathbf{G}_k$.

Not it's time to explain why we need the 0-1 property. If we regard the nonzero entries of $\mathbf{G}^{(INFE)}$ as the focal edges in that timestamp, then the nonzero entries of $\mathbf{M}_0$ is the focal edges in the whole. If we omit the initialization step, the weights of those edges which not appears in $\mathbf{G}^{(INFE)}$ still mentain the same relationship of size, namely $\mathbf{G}$ will holds these non-correlated edges. However, after the initialization step, the weights of those edges will be set to 0.5, which won't pass through the $\mathcal{H}_0$ function. Additionally, this sleeping $\frac{1}{2}$-$\frac{1}{2}$ structure is also easy to "wake up", as long as the new $\mathbf{G}^{(INFE)}$ have these edges.

4.5 Pertuation Module

As we mentioned before, our intrinsic target is to find the correct causality graph, and the way to achieve this goal is to minimize the difference caused by low frequency disturbution and maximize the difference caused by high frequency pertuation.

In the Causal Recognization Module, after each batch we got a causality graph $\mathbf{G}$ (we omit the underscript). Then we use equation (2) to do the pertuation.

if $\mathbf{x} = (x_1, x_2, \cdots, x_{2n})$, we can define a cut operation:

$$\mathcal{C}_k: \quad \mathbb{R}^{2n} \to \mathbb{R}^{2n}; \quad \mathbf{x} \mapsto (x_1, \ldots, x_k, 0, \ldots, 0, x_{n+1}, \ldots, x_{n+k}, 0, \ldots, 0) \tag{18}$$

$$\mathcal{C}^k: \quad \mathbb{R}^{2n} \to \mathbb{R}^{2n}; \quad \mathbf{x} \mapsto (0, \ldots, 0, x_{n-k+1}, \ldots, x_n, 0, \ldots, 0, x_{2n-k+1}, \ldots, x_{2n}) \tag{19}$$

Then the pertuation can be written as:

$$\mathbf{x}_{low} = \mathcal{F}^{-1}(\mathcal{C}^{n-k}(\mathcal{F}\mathbf{x}) + (1-\beta)\mathcal{C}_k(\mathcal{F}\mathbf{x}) + \beta\mathcal{C}_k(\mathcal{F}\mathbf{x}')) \tag{20}$$

$$\mathbf{x}_{high} = \mathcal{F}^{-1}(\mathcal{C}_{n-k}(\mathcal{F}\mathbf{x}) + (1-\beta)\mathcal{C}^k(\mathcal{F}\mathbf{x}) + \beta\mathcal{C}^k(\mathcal{F}\mathbf{x}')) \tag{21}$$

Here $\mathbf{x}$ is the sample in the current batch, $\mathbf{x}'$ is a random sample from the adjacent timestamp (i.e. if $\mathbf{x}$ is in T_m, then $\mathbf{x}'$ is in T_{m-1} or T_{m+1}), $\beta \sim U(0, \eta)$ is the pertuation intensity, where η is the max pertuation intensity, k is the pertuation dimension And the Graph Fourier Transform is based on the causality (adjacent) graph $\mathbf{G}$.

4.6 Projection Module

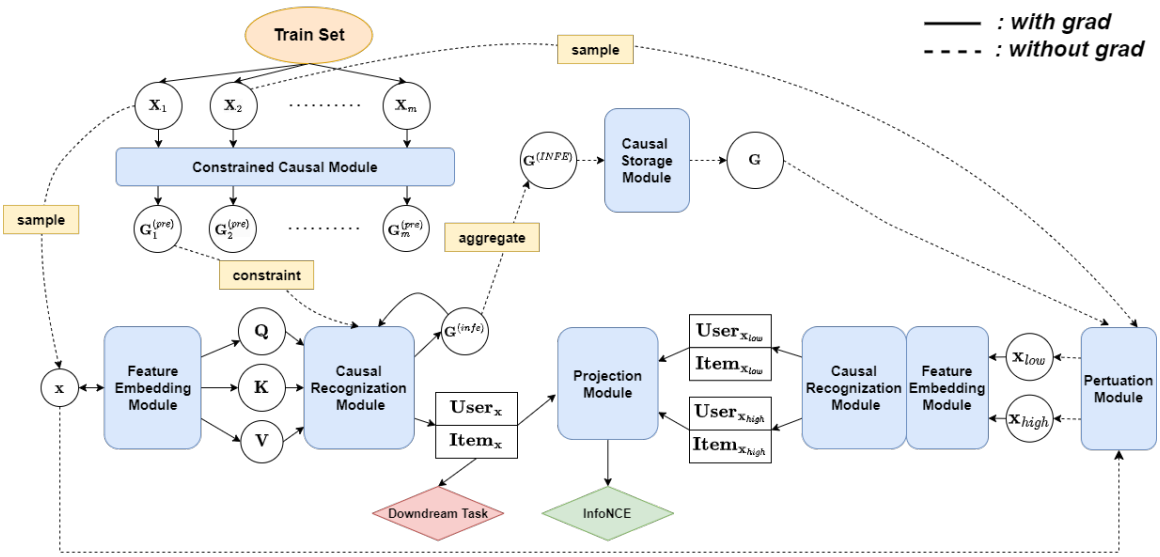
Rather than simply compare between these samples, a projector structure is more reliable in Constrastive Learning [1], namely a FFN layer, a batchnorm layer and another FFN layer. We denote it as $\mathcal{P}roj(\cdot)$.

We use the InfoNCE as the loss function:

$$\begin{aligned}
S(\mathbf{x}, \mathbf{y}) &= \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||_2||\mathbf{y}||_2} \\
L_{user}(\mathbf{x}) &= -\ln \frac{e^{S(\mathbf{User}_{\mathbf{x}}, \mathbf{User}_{\mathbf{x}_{low}})/\tau}}{e^{S(\mathbf{User}_{\mathbf{x}}, \mathbf{User}_{\mathbf{x}_{low}})/\tau} + e^{S(\mathbf{User}_{\mathbf{x}}, \mathbf{User}_{\mathbf{x}_{high}})/\tau}} \\
&= -\ln \mathrm{Sigmoid}(\frac{S(\mathbf{User}_{\mathbf{x}}, \mathbf{User}_{\mathbf{x}_{low}}) - S(\mathbf{User}_{\mathbf{x}}, \mathbf{User}_{\mathbf{x}_{high}})}{\tau}) \\
L_{item}(\mathbf{x}) &= -\ln \mathrm{Sigmoid}(\frac{S(\mathbf{Item}_{\mathbf{x}}, \mathbf{Item}_{\mathbf{x}_{low}}) - S(\mathbf{Item}_{\mathbf{x}}, \mathbf{Item}_{\mathbf{x}_{high}})}{\tau}) \\
L(\mathbf{x}) &= L_{user}(\mathbf{x}) + L_{item}(\mathbf{x})
\end{aligned}$$

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. 2020.


Train Set
sample
: with grad
: without grad
$\mathbf{X}_1$
$\mathbf{X}_2$
$\mathbf{X}_m$
Constrained Causal Module
$\mathbf{G}^{(INFE)}$
Causal Storage Module
$\mathbf{G}$
sample
$\mathbf{G}_1^{(pre)}$
$\mathbf{G}_2^{(pre)}$
$\mathbf{G}_m^{(pre)}$
aggregate
constraint
$\mathbf{G}^{(infe)}$
$\mathbf{x}$
Feature Embedding Module
Q
K
V
Causal Recognization Module
$\mathbf{User}_{\mathbf{x}}$
$\mathbf{Item}_{\mathbf{x}}$
Projection Module
$\mathbf{User}_{\mathbf{x}_{low}}$
$\mathbf{Item}_{\mathbf{x}_{low}}$
$\mathbf{User}_{\mathbf{x}_{high}}$
$\mathbf{Item}_{\mathbf{x}_{high}}$
Causal Recognization Module
Feature Embedding Module
$\mathbf{x}_{low}$
$\mathbf{x}_{high}$
Pertuation Module
Downdream Task
InfoNCE