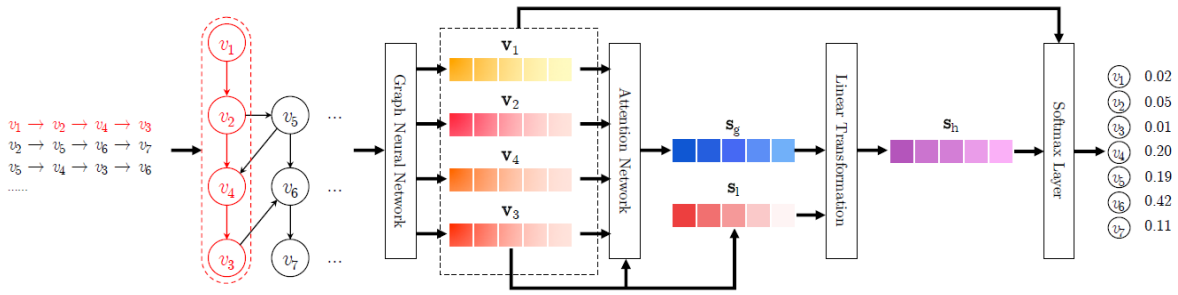
AAAI19一篇将gated GNN应用于序列推荐任务的论文

解决的问题

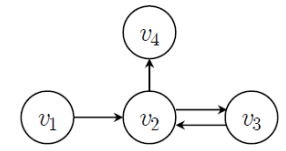
在序列推荐任务中，现有的方法很难在每条序列中取得准确的用户embedding，因为得到的序列数据往往是匿名的，且序列中记录的点击数据所透露出来的用户行为信息有限。同时，序列中物品间的关系虽然常被证实有效，但现有的方法往往只考虑一阶的前后连续关系，即对于$a \rightarrow b \rightarrow c$，只考虑$a \rightarrow b$或者$b \rightarrow c$

做法及创新



Session Graph Modeling

将每条序列s表示成一个有向图，并对图中的边进行正则化，具体做法为边的出现次数除以边起始顶点的出度。以序列$s = [v_1, v_2, v_3, v_2, v_4]$为例构建一个有向图，得到邻接矩阵：

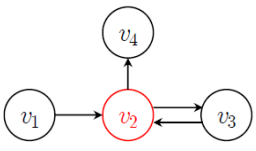


$$s = [v_1, v_2, v_3, v_2, v_4]$$

normalization

	1	2	3	4
1	0	1	0	0
2	0	0	1/2	1/2
3	0	1	0	0
4	0	0	0	0

上面的邻接矩阵以考虑顶点的出边并以出度正则化，类似地可以考虑顶点的入边并以入度正则化，将得到的两种邻接矩阵进行拼接，得到论文中提到的连接矩阵$A_s \in \mathbb{R}^{n \times 2n}$，其中的一行$A_{s,i:} \in \mathbb{R}^{1 \times 2n}$对应于所构建的有向图中的一个顶点$v_{s,i}$：



$$\mathbf{A}_{s,i:} \in \mathbb{R}^{1 \times 2n}$$

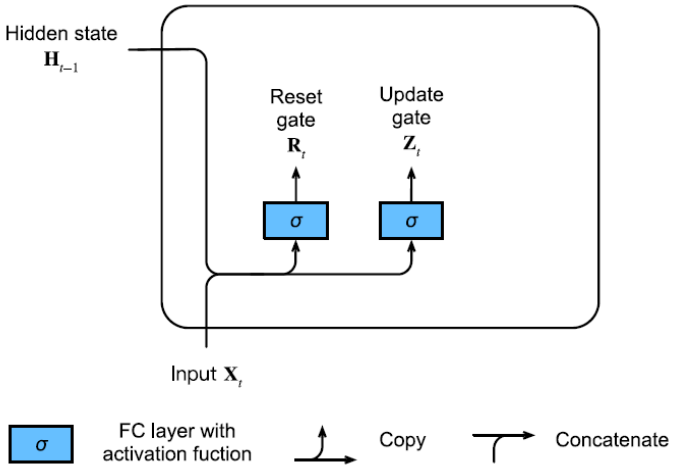
	Outgoing edges				Incoming edges			
	1	2	3	4	1	2	3	4
1	0	1	0	0	0	0	0	0
2	0	0	1/2	1/2	1/2	0	1/2	0
3	0	1	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0

Node Representation Learning

论文使用gated GNN来学习图中顶点的表示，为了类比地说明各式的具体含义，首先对Gated Recurrent Units（GRU）进行介绍，它是循环神经网络中的一个概念。

GRU

一个典型的GRU如下所示，输入为上一时刻的隐层表示H_{t-1}及当前时刻的表示X_t，包含一个重置门Reset Gate和一个更新门Update Gate：



直观的来说，重置门决定有多少历史信息被保留，而更新门决定利用多少当前时刻X_t的信息。给定当前时刻输入$X_t \in \mathbb{R}^{n\times d}$，上一时刻隐层表示$H_{t-1} \in \mathbb{R}^{n\times h}$，重置门与更新门的输出由下式计算得到：

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r)$$
$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z)$$

式中的W与b分别为权重与偏置参数。

Reset Gate

传统RNN网络的隐式状态更新公式为：

$$H_t = \tanh(X_t W_{xh} + H_{t-1} W_{hh} + b_h)$$

如果我们需要减少历史信息带来的影响，可以将H_{t-1}与R_t逐元素相乘。如果R_t中的元素接近于1，得到的结果就是传统的RNN，如果R_t中的结果接近于0，得到的结果就是以X_t作为输入的MLP，计算出来的$\tilde{H}_t$称为候选状态：

$$\tilde{H}_t = \tanh(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h)$$

Update Gate

更新门决定新的隐式状态H_t多大程度上与上一时刻H_{t-1}相同，以及重置门得到的候选状态$\tilde{H}_t$中有多少信息可以被利用，如果Z_t中的元素接近于1，将主要保留历史信息，当前时刻X_t的信息基本被忽略，这相当于跳过了时刻t；当Z_t中的元素接近于0时，H_t将主要由$\tilde{H}_t$决定：

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t$$

介绍完了GRU的基本概念，接下来是论文中的方法，可以类比地进行学习：

Propagation rules: connection matrix

$$\mathbf{a}_{s,i}^{t} = \mathbf{A}_{s,i:}\left[\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_n^{t-1}\right]^{\top}\mathbf{H} + \mathbf{b},$$

$$\mathbf{z}_{s,i}^{t} = \sigma\left(\mathbf{W}_z\mathbf{a}_{s,i}^{t} + \mathbf{U}_z\mathbf{v}_i^{t-1}\right), \quad \text{Reset gate}$$

$$\mathbf{r}_{s,i}^{t} = \sigma\left(\mathbf{W}_r\mathbf{a}_{s,i}^{t} + \mathbf{U}_r\mathbf{v}_i^{t-1}\right), \quad \text{Update gate}$$

$$\widetilde{\mathbf{v}}_i^{t} = \tanh\left(\mathbf{W}_o\mathbf{a}_{s,i}^{t} + \mathbf{U}_o\left(\mathbf{r}_{s,i}^{t} \odot \mathbf{v}_i^{t-1}\right)\right), \quad \text{Candidate}$$

$$\mathbf{v}_i^{t} = \left(1 - \mathbf{z}_{s,i}^{t}\right) \odot \mathbf{v}_i^{t-1} + \mathbf{z}_{s,i}^{t} \odot \widetilde{\mathbf{v}}_i^{t}. \quad \text{Final representation}$$

最主要的不同之处在公式(1)，它用于在连接矩阵A_s的约束下进行不同顶点间的信息传播，具体来说，它提取了邻域的隐向量并将它们作为GNN的输入。

Session Representation Generation

现有的做法都假设每条序列中的用户都有一个独特的隐式表示，而论文中提出的方法不对这个隐式向量做任何假设，相反，它用序列中顶点的表示来作为序列的表示，而顶点的表示正是上一步将所有序列构建的图送入gated GNN学习得到的。给定一个序列s $= [v_{s,1}, v_{s,2}, \ldots, v_{s,n}]$，这一步的目的是得到它的embedding向量$s \in \mathbb{R}^d$。为了结合用户的长期偏好与当前兴趣，生成的embedding向量也有局部和全局两部分组成。

局部embedding向量的构造非常简单，就是最后一个点击过的物品的表示，因为最后一个点击过的物品就表明了用户当前的兴趣：

$$s_l = v_n$$

全局embedding向量的构造需要将所有顶点的表示都聚合进来，论文的做法是做一个线性加权，权重使用soft-attention机制来计算得到：

$$s_g = \sum_{i=1}^{n} \alpha_i v_i$$
$$\alpha_i = q^T \sigma(W_1 v_n + W_2 v_i + c)$$

最后使用一个Linear层来将局部与全局embedding向量进行结合得到最终的序列embedding向量：

$$s_h = W_3[s_l; s_g]$$

Making Recommendation

对于一个待推荐物品$v_i \in V$，计算它在序列s中作为下一个被点击物品的概率：

$$\hat{y}_i = \text{softmax}(s_h^T v_i)$$

数据集

Yoochoose、Diginetica