信也科技杯 Data Science Competition | 信也 FINVOLUTION

FRAUD USER DETECTION

聚焦数创·连接未来

第七届信也科技杯算法大赛

7th Finvolution Data Science Competition

团队介绍

	Dylan	青禹小生	Xi Chu
工作经历	• 就职于风控行业数据产品公司 • 拥有多年风险标准评分产品开发经验 • 数据挖掘爱好者	• 就职于知名互联网大厂 • 在数据挖掘、NLP等领域有着丰富经验 • Kaggle Master	• 就职于风控行业头部消金公司 • 拥有多年业务侧风控模型研发经验 • 图网络模型爱好者

团队竞赛经历

2018京东JData算法大赛 - 冠军
CCF2020滴滴路况状态时空预测 - 冠军
2020厦门国际银行数创金融杯建模大赛 - 冠军
2020数字中国创新大赛 – 亚军
2020芒果TV视频推荐算法赛 – 亚军
2019天池全球城市计算AI挑战赛 - 季军

赛题描述

本届大赛的数据为经过脱敏的全连通的社交网络有向动态图，采样于主办方的实际业务数据
其中节点为注册用户，图中节点间为有向关联，图中边的不同类型代表了紧急联系人的分类

节点类型

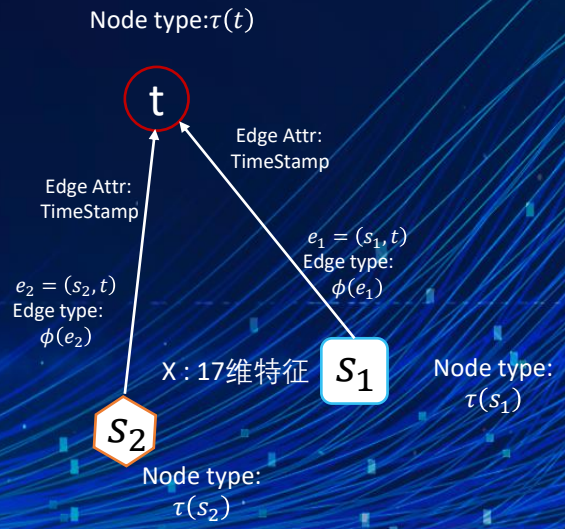
前景节点：欺诈用户 (Class 1)、正常用户 (Class 0)
背景节点：Class 2、 Class 3

数据维度

边属性：包括边连接的创建时间与11种边类型
节点属性：用户的17个维度基础特征

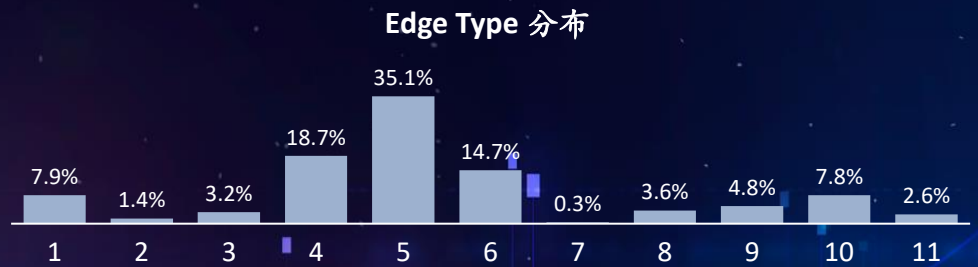
赛题目标

识别测试集中的欺诈用户，评估指标为AUC

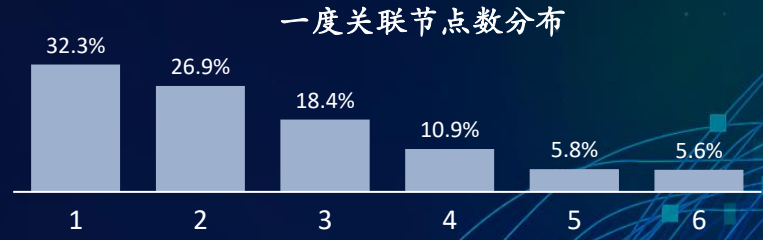


数据探索

节点主要属性



节点关联密度



关键发现

- 欺诈客群在前景节点中占比较少，而背景节点标签2/3占比很大，特征或标签传播时不能遗漏该部分客群；
- 从关联节点数分布与图节点可视化可以发现，整个图网络较稀疏 。

特征工程 – 各类特征组概览

一阶原始特征、节点属性、标签传播

1. 用户作为子节点时，其父节点feature1-feature17的统计特征
2. 用户作为父节点时，其子节点feature1-feature17的统计特征
3. 用户作为子节点时，与其关联的父节点节点数、timestamp及edge type的统计特征
4. 用户作为父节点时，与其关联的子节点节点数、timestamp及edge type的统计特征
5. 用户作为子节点时，其父节点的欺诈、通过、label2/3的个数与占比
6. 用户作为父节点时，其子节点的欺诈、通过、label2/3的个数与占比

二阶标签传播

1. 用户作为子节点时，拥有相同父节点的其他兄弟子节点的标签统计，包括欺诈、非欺诈、通过、label2/3的节点个数与占比
2. 用户作为父节点时，拥有相同子节点的其他兄弟父节点的标签统计，包括欺诈、非欺诈、通过、label2/3的节点个数与占比

关键发现与后续图网络构造启发

- 用户节点一阶关联时，可构造的有效特征维度非常丰富。而其中最为重要的是用户作为父节点所主动发生的行为统计；
- 二阶关联中的有效特征较少，只有二阶标签传播对模型有一定提升；
- 在二阶关联中，只有用户节点的兄弟节点相关统计有效，其他关联如孙子或祖父的二阶关联，没有增益。

比赛中尝试的三种图网络结构

- Heterogeneous Graph Transformer (HGT)
- **Heterogeneous** Edge-Enhanced Graph Attention Network (HEAT)
- Heterogeneous GraphSAGE

HGT : https://doi.org/10.48550/arXiv.2003.01332
HEAT : https://doi.org/10.48550/arXiv.2106.07161
GraphSage: https://doi.org/10.48550/arXiv.1706.02216

Graph Message Propagate

$$\mathbf{H^{(L)}} = \boldsymbol{\gamma^{(L)}}\left(\mathbf{H^{(L-1)}}, \mathbf{Aggregate}\left\{\mathbf{Message}\left(\mathbf{x_i^{(L-1)}}, \mathbf{x_j^{(L-1)}}, \mathbf{e_{j,i}}\right)\right\}\right)$$

$H^{(L-1)}$: L-1层的节点表示

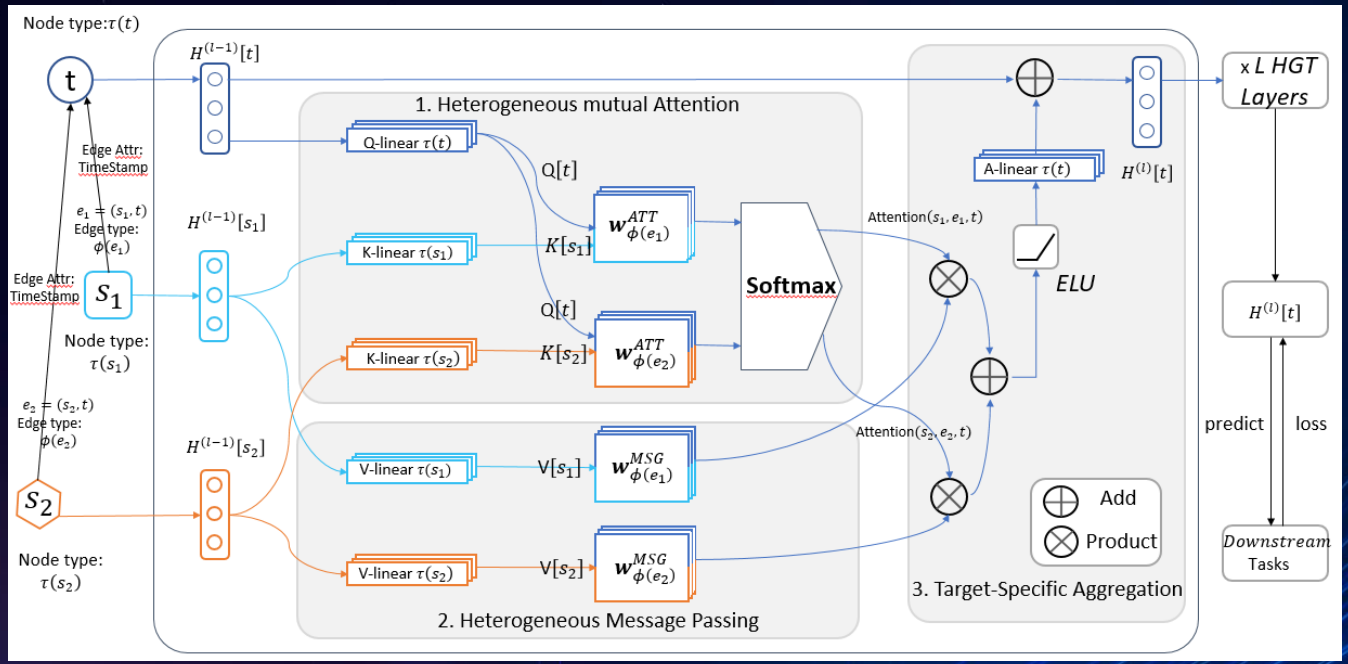
Aggregate: 聚合方式，例如mean, add, max 等或 muti-attention 方式

$x_i^{(L-1)}$: 表示L-1层的目标节点特征

$x_j^{(L-1)}$: 表示L-1层的邻居节点特征

$e_{j,i}$： 表示节点j指向节点i 的边特征

Heterogeneous Graph Transformer (HGT)



网络结构组成

1、Attention:基于异构图的muti-attention权重计算

2、Message Passing:异构图消息传递

3、Aggregation: 消息聚合

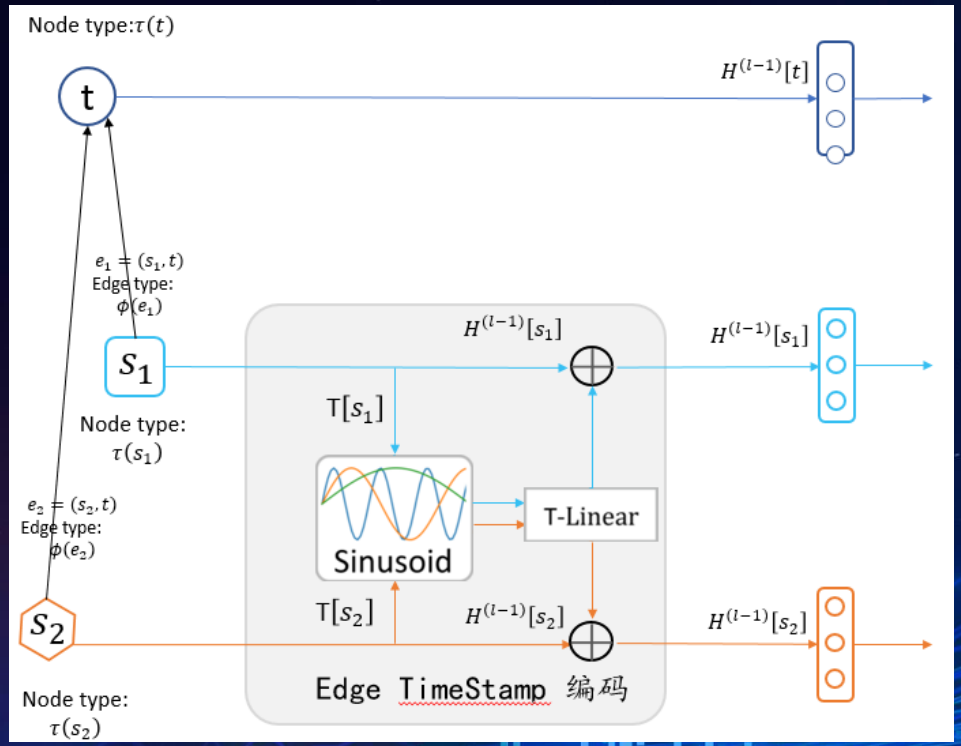
网络结构特点

1、采用基于多头注意力机制

2、元路径{s1, e1, t}和元路径{s2, e2, t}之间隐藏层参数不共享，所以该图结构参数量多

3、使用Transformer中的时间时间编码方式处理边上的TimeStamp

Heterogeneous Graph Transformer (HGT)



该模型的另一个特点则是对于边上时间信息采用了Transformer中序列时间节点的处理方式；

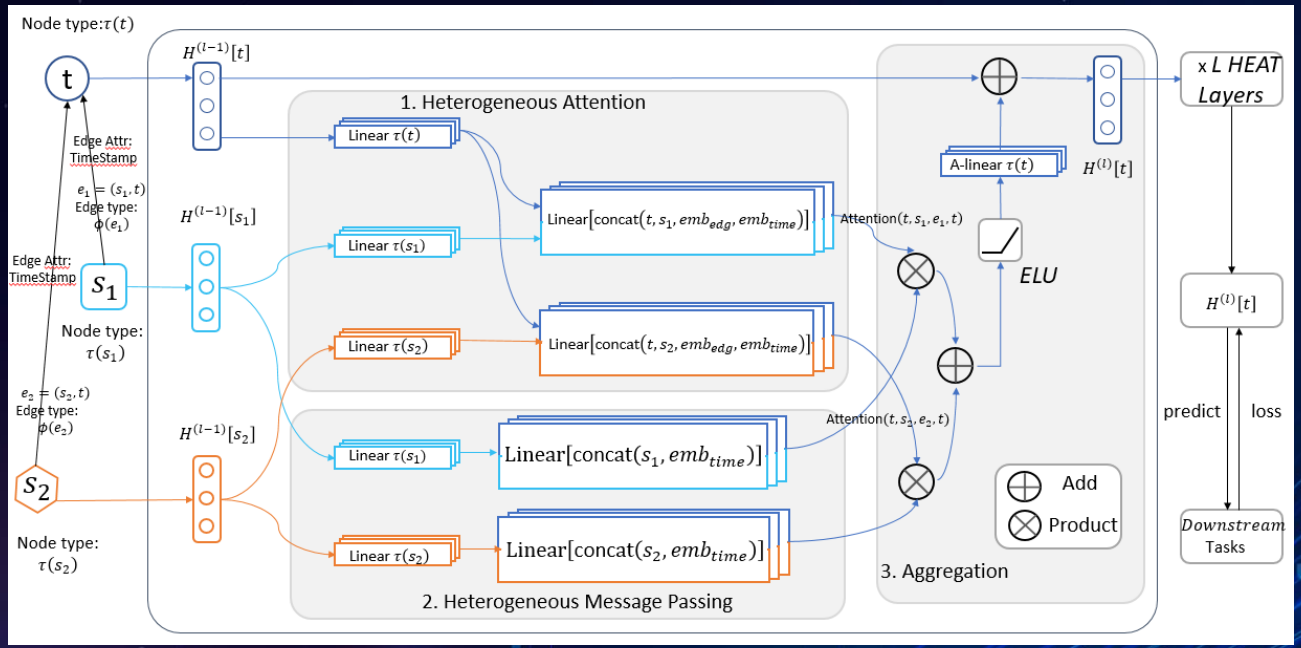
使用sin, cos周期函数编码的方式嵌入处理，学习相对时间的概念

Base(time, $2i$) = $\sin\left(\text{time}/1000^{\frac{2i}{d}}\right)$

Base(time, $2i+1$) = $\cos\left(\text{time}/1000^{\frac{2i+1}{d}}\right)$

Edge_time_encoder = T-Linear(Base(time))

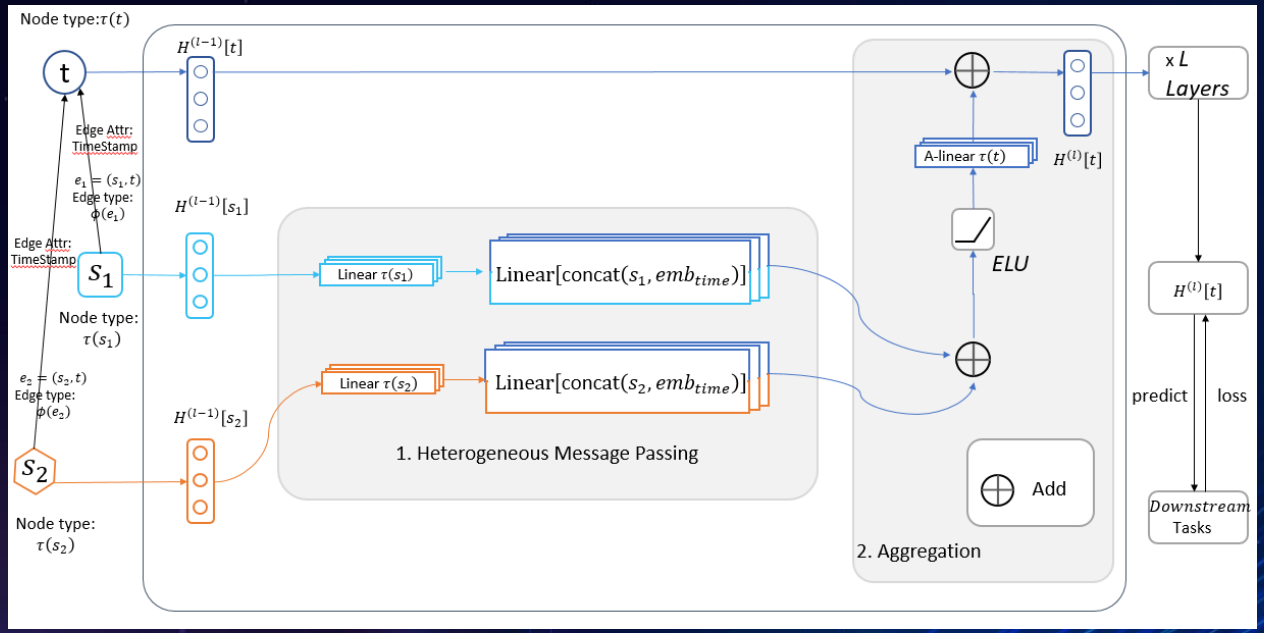
Heterogeneous Edge-Enhanced Graph Attention Network (HEAT)



网络结构特点

1. 边类型和边属性（时间）两种嵌入矩阵
2. 邻居节点权重采用目标节点特征、邻居节点特征、边类型的嵌入矩阵和边属性的前途矩阵的拼接后，接入全连接隐藏层，计算权重结果
3. 消息传播采用邻居节点特征和目标节点特征拼接的方式
4. 该网络结构参数量较少

Heterogeneous GraphSAGE



网络结构特点

1. 相比HGT和HEAT结构，去除掉了注意力权重的计算部分，网络机构轻盈。
2. 对图数据进行修改，使得同构的GraphSAGE图可以有效处理异构数据。
3. 参数量适中，效果最好。

三种图算法对比

	HGT	HEAT	Hetero - GraphSAGE
层数	2	2	2
隐藏层维度	256	256	256
残差	是	是	是
输出层normalize	否	否	是
参数量	828692	85252	141058
learning rate	0.001	0.001	0.001
epochs	100	100	100
early stop epochs	10	10	10
效果（AUC）	0.837	0.812	0.843

异构图数据生成过程

Hetero Data ← 异构图初始化声明，使用torch_gemetric结构

↓

生成两种节点类型 ← 图中y值为0，1，-100的节点设置为类型1，
图中y值为2，3的节点设置为类型2

↓

新增边和边类型 ← 邻居节点指向目标节点的方向，
新增目标节点指向邻居节点的边

↓

生成异构图数据

核心代码 - 异构图数据生成第一步

```
## 1、声明异构图
data = HeteroData()

y_df = pd.DataFrame(y)
y_df.columns = ['y']

#     y_df['node_type'] = 1
y_df.loc[y_df['y'].isin([0, 1, -100]), 'node_type'] = 1
y_df.loc[y_df['y'].isin([2, 3]), 'node_type'] = 2

# 对三类节点生成新节点排序 和 旧节点排序的映射关系
# node_type = 1
# type 1 的 新节点node id 对应着 旧节点的node id
node_type_1_new2old_id_dict = y_df[y_df['node_type'] == 1].reset_index()['index'].to_dict()

# node_type = 2
# type 2 的 新节点node id 对应着 旧节点的node id
node_type_2_new2old_id_dict = y_df[y_df['node_type'] == 2].reset_index()['index'].to_dict()

node_type_1_new2old_id_dict_back = {k:v for v , k in node_type_1_new2old_id_dict.items()}
node_type_2_new2old_id_dict_back = {k:v for v , k in node_type_2_new2old_id_dict.items()}
```

核心代码 - 异构图数据生成第二步

```
## 2、交换两列，互相指向
# 增加边
edge_index = np.concatenate([edge_index[:,[1,0]], edge_index])
# 增加边类型
edge_type = np.concatenate([edge_type, edge_type + 11])
# 增加边属性
edge_timestamp = np.concatenate([edge_timestamp, edge_timestamp])
##对一些重复的双向边进行去重
edge_index_df = pd.DataFrame(edge_index)
edge_index_df['type'] = edge_type
edge_index_df['edge_weight'] = edge_timestamp
edge_index_df = edge_index_df.drop_duplicates(subset= [0, 1])
edge_index_df = edge_index_df.reset_index(drop=True)

##### 3、生成去重后的边，节点类型，节点权重
edge_index = edge_index_df[[0, 1, 'edge_weight']].values
edge_type = edge_index_df['type'].values
#生成节点值
data['type_1'].x = torch.tensor(x[list(node_type_1_new2old_id_dict.values())], dtype=torch.float).contiguous()
data['type_2'].x = torch.tensor(x[list(node_type_2_new2old_id_dict.values())], dtype=torch.float).contiguous()
```

核心代码-异构图数据生成第三步

```
## 4、生成异构图边数据

for edge_type_num in tqdm(range(1, len(set(edge_type)) + 1)):

    sub_edge_index = edge_index[edge_type == edge_type_num]

    # 先找出type 1
    tmp_edge_index_df = pd.DataFrame(sub_edge_index)

    # sub_edge_weight = edge_weight[edge_type == edge_type_num]
    ##### 不同的边类型的处理
    data = create_node_func(data, main_id_dict= node_type_1_new2old_id_dict, main_id_dict_back =
                            node_type_1_new2old_id_dict_back, src_node_name = 'type_1', edg_name =
                            f'{edge_type_num}', tmp_edge_index_df = tmp_edge_index_df )
    data = create_node_func(data, main_id_dict= node_type_2_new2old_id_dict, main_id_dict_back =
                            node_type_2_new2old_id_dict_back, src_node_name = 'type_2', edg_name =
                            f'{edge_type_num}', tmp_edge_index_df = tmp_edge_index_df )

## 5、生成y
data['type_1']['y'] = torch.tensor(y_df[y_df['node_type'] == 1]['y'].values, dtype=torch.int64)
```

核心代码-异构图生成后的数据格式

```
HeteroData(
  type_1={
    x=[1181925, 17],
    y=[1181925],
    train_mask=[1181925],
    valid_mask=[1181925],
    test_mask=[1181925],
    test_mask_map=[354578]
  },
  type_2={ x=[2877110, 17] },
  (type_1, 1, type_1)={
    edge_index=[2, 79047],
    edge_weight=[79047]
  },
  (type_1, 1, type_2)={
    edge_index=[2, 59541],
    edge_weight=[59541]
  },
)
```

```
(type_2, 1, type_1)={
  edge_index=[2, 121986],
  edge_weight=[121986]
},
(type_2, 1, type_2)={
  edge_index=[2, 130251],
  edge_weight=[130251]
},
(type_1, 2, type_1)={
  edge_index=[2, 7367],
  edge_weight=[7367]
},
  .....

(type_2, 22, type_1)={
  edge_index=[2, 16144],
  edge_weight=[16144]
},
(type_2, 22, type_2)={
  edge_index=[2, 59926],
  edge_weight=[59926]
}
```

核心代码-异构图模型构建

```
class GraphSageModel(torch.nn.Module):
    def __init__(self, hidden_channels, out_channels):
        super().__init__()

        self.conv1 = SelfConv_src( 18, hidden_channels , normalize = True, self_flag = True)
        self.conv2 = SelfConv_src( hidden_channels, hidden_channels, normalize = True)

        self.fc = Linear(hidden_channels, out_channels)


    def forward(self, x, edge_index, edge_weight):

        x = self.conv1(x, edge_index, edge_weight).relu()
        x = self.conv2(x, edge_index, edge_weight).relu()
        x = self.fc(x)

        return x
```

最终方案构成

- 一阶 特征、节点属性及标签传播
- 二阶 标签传播
- 节点基础特征

LightGBM
Xgboost
Catboost

HeteroData异构图数据生成

Hetero Graphsage

AUC: 0.8141

信也科技杯 Data Science Competition | 信也 FINVOLUTION

TNANK YOU