HAN

<https://github.com/Jhy1993/HAN>

来源：www 2019

作者：王晓，纪厚业 北京邮电大学 史川，王白 北京邮电大学

崔鹏， P. Yu 清华大学 叶廷芳 美国西弗吉尼亚大学

图网络面临的新问题：

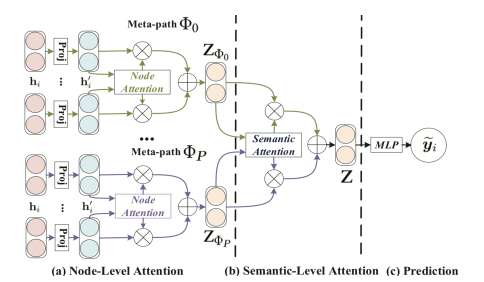
1. 图的异构性：如何同时处理复杂的结构信息并同时保存多样的特征信息
2. 语义级别注意力：如何选择最有意义的元路径并为特定任务融合语义信息

3、节点级别注意力：如何区分邻居之间的细微差别并选择一些信息丰富的邻居？ 将期望如何设计才能够发现邻居的细微差异并适当地学习其权重的模型？

基于以上三个问题本文提出HAN。

HAN：

1. 同时考虑了节点级别和语义级别的注意力；
2. 模型效率高，相对基于元路径的节点对的数量具有线性复杂度，可以应用于大规模异构图。



图一 HAN模型框架

1. 将所有类型的节点投影到统一的特征空间中，并且可以通过节点级注意力来学习基于元路径的节点对的权重。

节点级注意力学习过程:

以特定类型的转换矩阵进行投影:

基于元路径的节点对（i，j）的重要性可以表述为：attnode表示执行节点级注意力的深度神经网络.

我们对其进行归一化，以通过softmax函数获得权重系数，σ表示激活函数,aΦ是元路径Φ的节点级注意力向量：

节点i的基于元路径的嵌入：由邻居的投影特征以相应的系数进行加和：

节点级别的注意力重复K次，并将学习到的嵌入连接为特定于语义的嵌入：

1. 联合学习每个元路径的权重，并通过语义级别的注意力来融合特定于语义的节点嵌入。

语义级注意力学习过程：

融合可以通过元路径显示的多个语义特定的节点嵌入生成的学习权重，attsem表示执行语义级注意力的深度神经网络：

每个元路径的重要性表示为，其中W是权重矩阵，b是偏差向量，q是语义级别注意向量。

可以通过使用softmax函数归一化所有元路径的上述重要性来获得元路径的权重，称为：

越高，元路径越重要。

以学习到的权重为系数，我们可以融合这些特定于语义的嵌入以获得最终的嵌入Z，

c）计算HAN的损耗和端到端的优化。

对于半监督节点分类，我们可以使地面真实性与预测之间的所有标记节点的交叉熵最小：

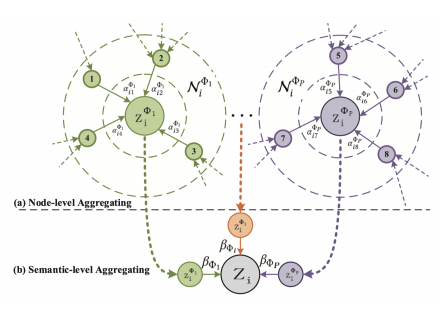


图2节点级和语义级的聚合过程的说明

数据集：

Acm <http://dl.acm.org/>

Dblp <https://dblp.uni-trier.de>

imdb [https://www.imdb.com](   https://www.imdb.com)

论文ppt:

<https://github.com/Jhy1993/HAN>

代码环境配置：

Python == 3.6+ conda create -n name python==3.6

Tensorflow pip install tensorflow

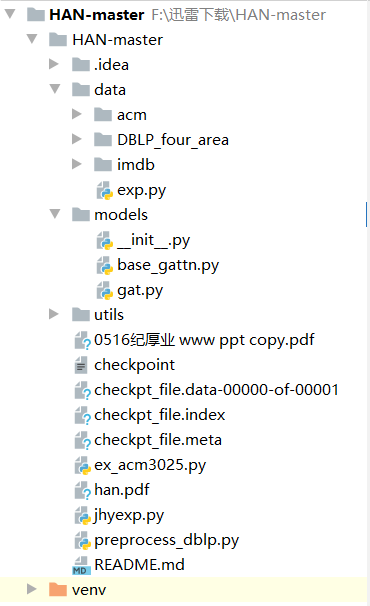
下载数据集ACM:

https://pan.baidu.com/s/1V2iOikRqHPtVvaANdkzROw 提取码：50k2

或者：<https://bupteducn-my.sharepoint.com/:u:/g/personal/jhy1993_bupt_edu_cn/EfLZcHE2e4xBplCVnzcJbQYBurNVOCk7ZIn>

更改路径：

def load\_data\_dblp(path='/home/jhy/allGAT/acm\_hetesim/ACM3025.mat'):  
python ex\_acm3025.py



Data:分别是acm,dblp,imdb三个数据集

Models：本论文所用模型的构建和相关公式的实现

Utils:

Ex\_acm3025.py:对acm数据集训练分析学习到的注意力权重。

Jhyexp:对imdb数据集进行分类和聚类证明HAN的有效性

Preprocess\_dblp：对数据集dblp进行预测

Gat.py:建立模型：

图注意力网络模型，多节点图注意力网络模型，异构图注意力网络模型

Base\_gattn.py:公式的代码实现：

对于半监督节点分类，使地面真实性与预测之间的所有标记节点的交叉熵最小：

获得交叉熵

Def loss(logits, labels, nb\_classes, class\_weights):

class\_weights表示语义级权重，nb\_classes表示节点级权重，labels表示标签

return tf.reduce\_mean(xentropy, name=**'xentropy\_mean'**)

通过minimize函数获得交叉熵最小值

train\_op = opt.minimize(loss + lossL2)

以特定类型的转换矩阵M进行投影

#以特定类型的转换矩阵进行投影  
def preshape(logits, labels, nb\_classes):  
 new\_sh\_lab = [-1]  
 new\_sh\_log = [-1, nb\_classes]  
 log\_resh = tf.reshape(logits, new\_sh\_log)  
 lab\_resh = tf.reshape(labels, new\_sh\_lab)  
 return log\_resh, lab\_resh  
#转换矩阵M函数  
def confmat(logits, labels):  
 preds = tf.argmax(logits, axis=1)  
 return tf.confusion\_matrix(labels, preds)

基于元路径的节点对（i，j）的重要性

def masked\_accuracy(logits, labels, mask):  
 """Accuracy with masking."""  
 correct\_prediction = tf.equal(  
 tf.argmax(logits, 1), tf.argmax(labels, 1))  
 accuracy\_all = tf.cast(correct\_prediction, tf.float32)  
 mask = tf.cast(mask, dtype=tf.float32)  
 mask /= tf.reduce\_mean(mask)  
 accuracy\_all \*= mask  
 return tf.reduce\_mean(accuracy\_all)

我们对其进行归一化，以通过softmax函数获得权重系数，σ表示激活函数,aΦ是元路径Φ的节点级注意力向量：

def masked\_softmax\_cross\_entropy(logits, labels, mask):  
 """Softmax cross-entropy loss with masking."""  
 loss = tf.nn.softmax\_cross\_entropy\_with\_logits(  
 logits=logits, labels=labels)  
 mask = tf.cast(mask, dtype=tf.float32)  
 mask /= tf.reduce\_mean(mask)  
 loss \*= mask  
 return tf.reduce\_mean(loss)

节点i的基于元路径的嵌入：由邻居的投影特征以相应的系数进行加和：

def masked\_sigmoid\_cross\_entropy(logits, labels, mask):  
 """Softmax cross-entropy loss with masking."""  
 labels = tf.cast(labels, dtype=tf.float32)  
 loss = tf.nn.sigmoid\_cross\_entropy\_with\_logits(  
 logits=logits, labels=labels)  
 loss = tf.reduce\_mean(loss, axis=1)  
 mask = tf.cast(mask, dtype=tf.float32)  
 mask /= tf.reduce\_mean(mask)  
 loss \*= mask  
 return tf.reduce\_mean(loss)

#F测度的获取

def micro\_f1(logits, labels, mask):  
 """Accuracy with masking."""  
 predicted = tf.round(tf.nn.sigmoid(logits))  
 # Use integers to avoid any nasty FP behaviour  
 predicted = tf.cast(predicted, dtype=tf.int32)  
 labels = tf.cast(labels, dtype=tf.int32)  
 mask = tf.cast(mask, dtype=tf.int32)

mask = tf.expand\_dims(mask, -1)  
 # Count true positives, true negatives, false positives and false negatives.  
 tp = tf.count\_nonzero(predicted \* labels \* mask)  
 tn = tf.count\_nonzero((predicted - 1) \* (labels - 1) \* mask)  
 fp = tf.count\_nonzero(predicted \* (labels - 1) \* mask)  
 fn = tf.count\_nonzero((predicted - 1) \* labels \* mask)  
 precision = tp / (tp + fp)  
 recall = tp / (tp + fn)  
 fmeasure = (2 \* precision \* recall) / (precision + recall)  
 fmeasure = tf.cast(fmeasure, tf.float32)  
 return fmeasure

Preprocess\_dblp.py

对数据集dblp进行预测，获得z，α，β

Jhyexp.py

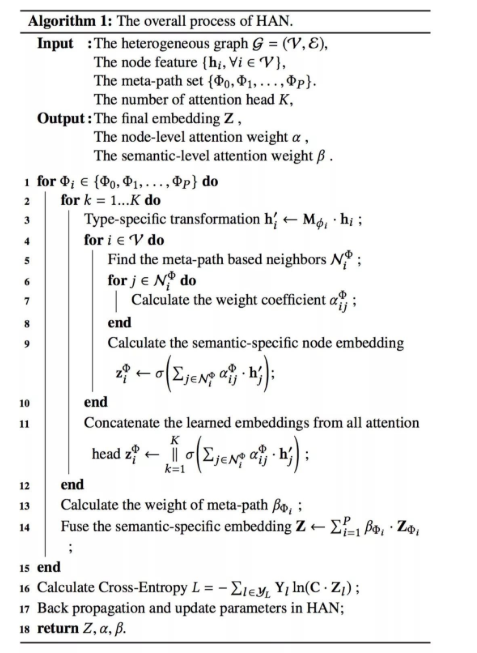
对数据集imdb进行分类和聚类处理，证明HAN的有效性

分类用KNN算法；聚类用K-means算法

Ex\_acm3025.py

引入acm数据集生成字典，通过HAN训练模型，

算法具体实现:



输入：

bias\_mat\_list：原路径列表；n\_heads：注意力头

class HeteGAT\_multi(BaseGAttN):  
 def inference(inputs\_list, nb\_classes, nb\_nodes, training, attn\_drop, ffd\_drop,  
 bias\_mat\_list, hid\_units, n\_heads, activation=tf.nn.elu,

residual=False,  
 mp\_att\_size=128):

前馈运算：获得节点嵌入，注意力值

logits, final\_embedding, att\_val = model.inference(ftr\_in\_list, nb\_classes, nb\_nodes, is\_train,attn\_drop, ffd\_drop, bias\_mat\_list=bias\_in\_list, hid\_units=hid\_units, n\_heads=n\_heads, residual=residual, activation=nonlinearity)