About me

- **Name**: 叶晨
- **Student ID**: 21821211
- **Topic**: Neural Networks
- **Email**: ye.chen@zju.edu.cn

Schedule

Task	Due	Done
1.选择论文	Mar.14	√
2.精读论文	Mar.21	√
3.复现论文	Apr.4	√
4. 完成实验	Apr.11	√
5.撰写报告	Apr.18	√

选择论文

ANRL: Attributed Network Representation Learning via Deep Neural Networks

- **Abstract**

> Network representation learning (RL) aims to transform the nodes in a network into low-dimensional vector spaces while preserving the in- herent properties of the network. Though net-work RL has been intensively studied, most ex- isting works focus on either network structure or node attribute information. In this paper, we pro- pose a novel framework, named ANRL, to incorpo- rate both the network structure and node attribute information in a principled way. Specifically, we propose a neighbor enhancement autoencoder to model the node attribute information, which recon- structs its target neighbors instead of itself. To capture the network structure, attribute-aware skip- gram model is designed based on the attribute en- coder to formulate the correlations between each node and its direct or indirect neighbors. We con- duct extensive experiments on six real-world net- works, including two social networks, two citation networks and two user behavior networks. The re- sults empirically show that ANRL can achieve rel- atively significant gains in node classification and link prediction tasks.

- **摘要**

> 网络表示学习（RL）的目的是将网络中的节点转化为低维向量空间，同时保持网络的内在属性。尽管对网络工作RL进行了深入的研究，但现有的研究大多集中在网络结构和节点属性信息两个方面。本文提出了一个新的框架，即ANRL，它将网络结构和节点属性信息有机地结合起来。具体地说，我们提出了一种邻域增强自动编码器来对节点属性信息进行建模，它可以重建目标邻域而不是自身。为了捕捉网络结构，设计了基于属性编码器的属性感知跳图模型，用以描述每个节点与其直接或间接邻居之间的关系。我们对六个真实的网络作品进行了广泛的实验，包括两个社交网络、两个引用网络和两个用户行为网络。实验结果表明，在节点分类和链路预测任务中，ANRL能取得显著的效果。

精读论文

> 提出ANRL模型，融合了网络结构相似性和节点属性密切关系。具体来说，设计了一个邻居增强自编码器对属性信息建模，并用于重构目标节点的邻居；设计了一个属性感知skip-gram模型来捕获节点和邻居之间的相关性。这两个部分共享编码器的连接层。
>
> 其中，属性增强自编码器部分目标函数为：$L_{ae} = \sum_{i=1}^{n} ||\hat{x_i} - T(v_i)||_2^2$，通过使离得近的节点重构得到相似的目标邻居来使得离得近的节点有相似的表示，其中$\hat{x_i}$是节点v_i的邻居经过解码器的重构输出，$T(.)$有两种计算方式：1). 对于给定的节点v_i，目标邻居节点由邻居节点的加权平均得到，即$T(v_i) = \frac{1}{|N(i)|} \sum_{j \in N(i)} w_{ij} x_j$. 2). 对于节点$v_i$，$T(v_i) = \breve{x_i} = [\breve{x_1}, \breve{x_2}, \dots, \breve{x_m}]$，其中$\breve{x_k} = Median(w_{i1}x_{1k}, w_{i2}x_{2k}, \dots, w_{i|N(i)|}x_{|N(i)|k})$，即对应邻居节点属性向量的第k维的中位数。
>
> 属性感知的skip-gram部分的目标函数为：$L_{sg} = -\sum i = 1^n \sum_{c \in C} \sum_{-b \le j \le b, j \ne 0} \log p(v_{i+j}|x_i)$，其中，$p(v_{i+j}|x_i) = \frac{exp(v_{i+j}'^T f(x_i))}{\sum_{v=1}^{n} exp(v_v'^T f(x_i))}$，最终的目标函数为：$L = L_{sg} + \alpha L_{ae} + \beta L_{reg}$，其中，$L_{reg} = \frac{1}{2} \sum_{k=1}^{K} (||W^{(k)}||_F^2 + ||\hat{W}^{(k)}||_F^2)$。值得注意的是，$f(.)$函数就是自编码器的编码器部分，它将节点属性信息转换为隐藏空间的表示$y_i^{(K)}$。

复现论文

主要代码

- 损失函数：

```
def make_skipgram_loss(self):
    loss = tf.reduce_sum(tf.nn.sampled_softmax_loss(
    weights=self.nce_weights,
    biases=self.nce_biases,
    labels=self.labels,
    inputs=self.Y,
    num_sampled=self.config.num_sampled,
    num_classes=self.N))
    return loss

def make_autoencoder_loss(self):
    def get_autoencoder_loss(X, newX):
        return tf.reduce_sum(tf.pow((newX - X), 2))
    def get_reg_loss(weights, biases):
        reg = tf.add_n([tf.nn.l2_loss(w) for w in weights.values()])
        reg += tf.add_n([tf.nn.l2_loss(b) for b in biases.values()])
        return reg
    loss_autoencoder = get_autoencoder_loss(self.X_new, self.X_reconstruct)
    loss_reg = get_reg_loss(self.W, self.b)
    return self.config.alpha * loss_autoencoder + self.config.reg * loss_reg
```

- 迭代训练

```
for iter_cnt in range(max_iters):
    idx += 1
    batch_index, batch_labels = next(graph_context_batch_iter(all_pairs, batch_size
    # train for autoencoder model
    start_idx = np.random.randint(0, N - batch_size)
    batch_idx = np.array(range(start_idx, start_idx + batch_size))
    batch_idx = np.random.permutation(batch_idx)
    batch_X = X[batch_idx]
    feed_dict = {model.X: batch_X, model.inputs: batch_idx}
    _, loss_ae_value = sess.run([model.train_opt_ae, model.loss_ae], feed_dict=feed
    loss_ae += loss_ae_value
    # train for skip-gram model
    batch_X = X[batch_index]
    feed_dict = {model.X: batch_X, model.labels: batch_labels}
    _, loss_sg_value = sess.run([model.train_opt_sg, model.loss_sg], feed_dict=feed
    loss_sg += loss_sg_value
    if idx % print_every_k_iterations == 0:
        end = time.time()
        print('iterations: %d' % idx + ', time elapsed: %.2f, ' % (end - start), en
        total_loss = loss_sg / idx + loss_ae / idx
        print('loss: %.2f, ' % total_loss, end='')
        y = read_label(inputLabelFile)
        embedding_result = sess.run(model.Y, feed_dict={model.X: X})
        macro_f1, micro_f1 = multiclass_node_classification_eval(embedding_result,
        print('[macro_f1 = %.4f, micro_f1 = %.4f]' % (macro_f1, micro_f1))
```

- 其他代码详见 `code` 文件夹

完成实验

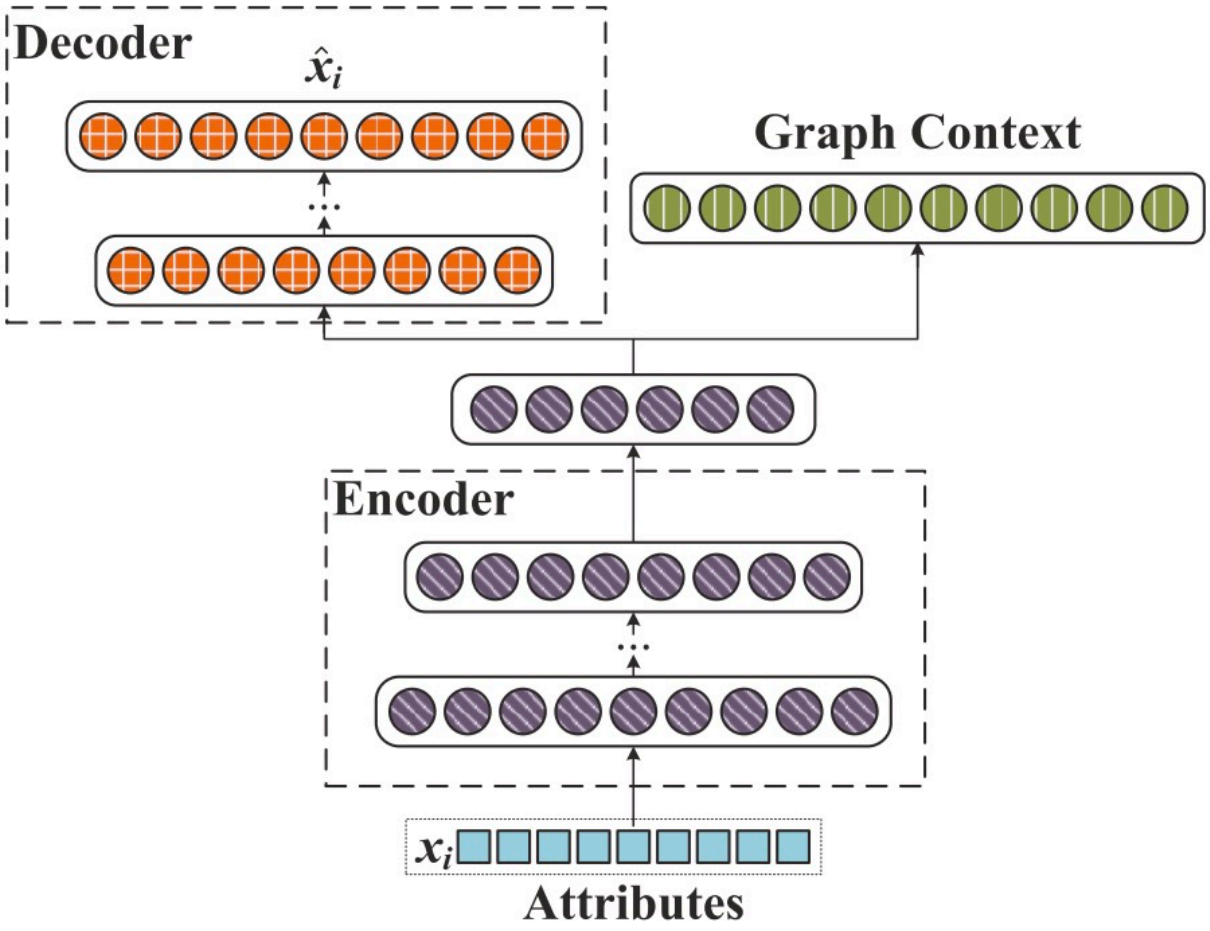
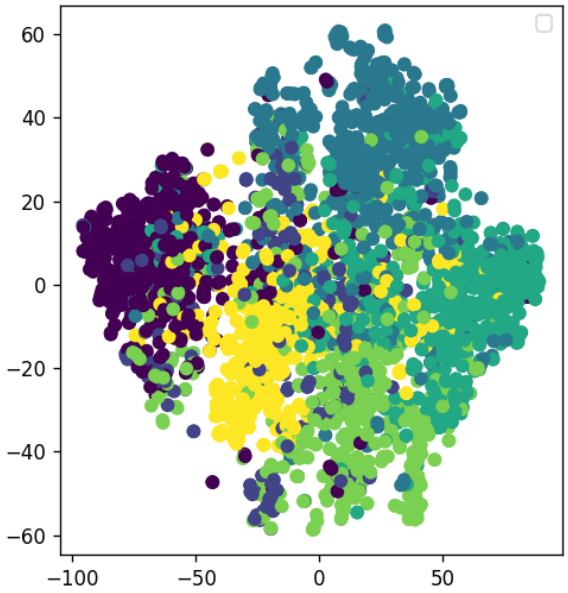


Figure 1: The architecture of the proposed ANRL model.



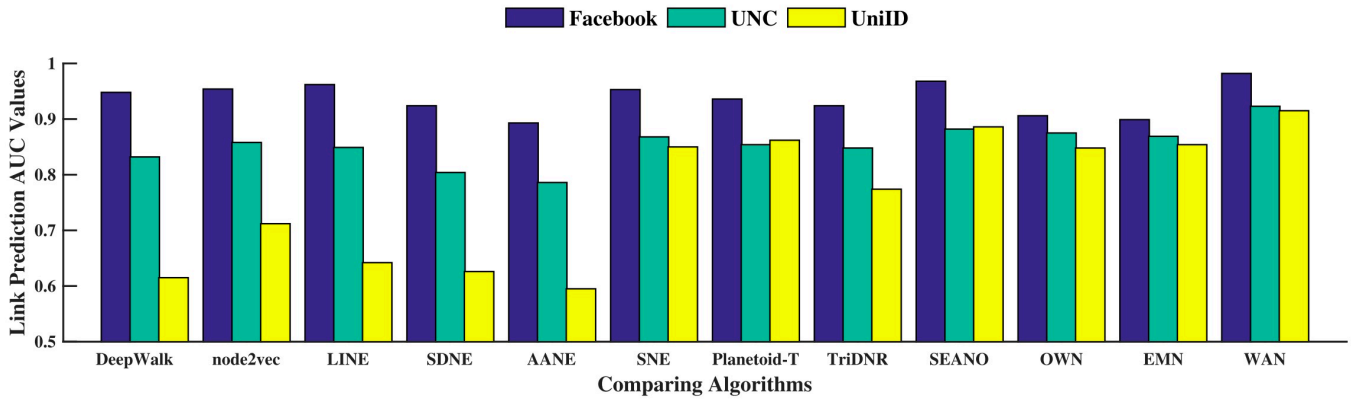


Figure 2: Link prediction performance comparisons of different algorithms on the Facebook, UNC and UniID. The y-axis represents the AUC value of each method while the x-axis shows the name of different methods. Note that we omit the prefix of the proposed ANRL variants.

Datasets	Citeseer		Pubmed		Fraud Detection	
Evaluation	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
SVM	0.667	0.626	0.856	0.855	0.725	0.719
autoencoder	0.630	0.565	0.792	0.800	0.732	0.726
DeepWalk	0.583	0.534	0.809	0.795	0.509	0.464
node2vec	0.607	0.561	0.815	0.802	0.571	0.519
LINE	0.542	0.512	0.766	0.749	0.659	0.654
SDNE	0.569	0.528	0.699	0.677	0.662	0.656
AANE	0.579	0.541	0.784	0.765	0.654	0.643
SNE	0.632	0.615	0.803	0.797	0.662	0.654
Planetoid-T	0.656	0.594	0.851	0.847	0.692	0.693
TriDNR	0.633	0.587	0.843	0.824	0.686	0.685
SEANO	0.713	0.662	0.859	0.848	0.703	0.704
ANRL-OWN	0.652	0.606	0.842	0.845	0.724	0.720
ANRL-EMN	0.716	0.668	0.865	0.867	0.733	0.731
ANRL-WAN	**0.729**	**0.673**	**0.876**	**0.871**	**0.759**	**0.755**

Table 3: Node classification results on Citeseer, Pubmed and Fraud Detection datasets. We use **blue** to highlight wins.

撰写报告