

# CAU-Net: A Convolutional Attention U-Net For Radar Signal Deinterleaving

## 1) 论文主要内容

通过设计一个网络，对接收信号进行去识别和分选。

## 2) 具体实现和 CAU-Net 网络

### 1. 实验搭建

发射信号：LFM 信号。

信号设置：其脉冲重复间隔和脉冲宽度采用抖动调制。

$$T'_r = (1 - \alpha + 2\alpha \cdot rand) \cdot T_r$$

$$T'_p = (1 - \alpha + 2\alpha \cdot rand) \cdot T_p$$

接收信号：

$$S(t) = \sum_{n=1}^N \sum_{m=1}^M A_n \cdot rect\left(\frac{t - t_{n,m}}{T'_p}\right) \exp\left(j\pi k(t - t_{n,m})^2\right) \quad (5)$$

$$t_{n,m} = \begin{cases} \text{TOA}_{n,0}, m = 1 \\ \text{TOA}_{n,0} + \sum_{i=1}^{m-1} T'_{r,i}, m \geq 2 \end{cases} \quad (6)$$

为  $t_{n,m}$  为第  $n$  个发射机的第  $m$  个脉冲的到达时间。

$\text{TOA}_{n,0}$  为第  $n$  个发射机的第一个脉冲的到达时间。

$T'_{r,i}$  为第  $n$  个发射机的第  $i$  个脉冲重复间隔。

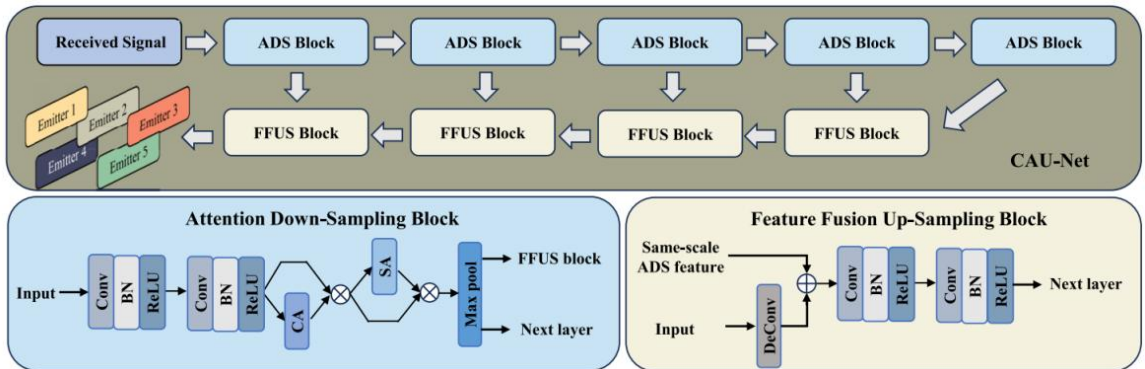
### 2. 网络结构

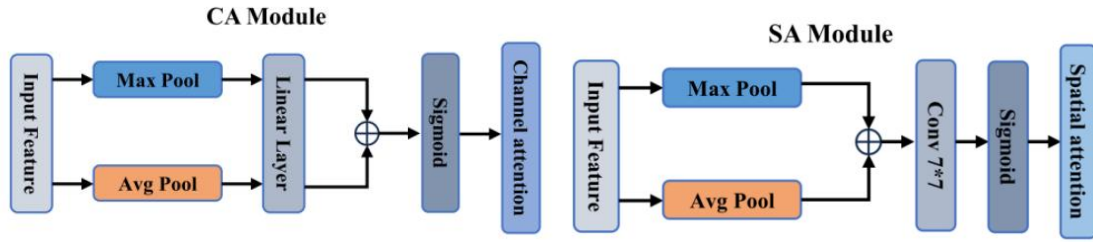
1.网络由注意力下采样（ADS）块和特征融合上采样（FFUS）块组成。

2.其下采样模块由两层卷积和 CA 与 SA 和最大池化层组成。

3.上采样模块由反卷积网络层和两层卷积网络组成。

4.CA 由一个最大池化和一个平均池化拼接之后再经过一个激活函数，SA 由一个最大池化和一个平均池化拼接之后再经过  $7 \times 7$  卷积层和激活函数。





CA 和 SA 的权重计算方式:

$$W_c(f) = \sigma(\text{Linear}(\text{avgpool}(f)) + \text{Linear}(\text{maxpool}(f))) \quad (7)$$

$$W_s(f) = \sigma(\text{conv}_{7 \times 7}(\text{concat}(\text{avgpool}(f), \text{maxpool}(f)))) \quad (8)$$

SA 和 CA 部分的过程:

$$f' = W_c(f) \otimes f$$

$$f'' = W_s(f') \otimes f'$$

损失函数:

$$Loss = \frac{1}{L \cdot N} \sum_{l=1}^L \sum_{n=1}^N (\hat{y}_{l,n} - y_{l,n})^2$$

### 3. 实验内容

```
epoch: 83 sim: 0.8090159296989441
epoch: 84 train_loss: 0.019878664949593453
epoch: 84 sim: 0.813949704170227
epoch: 85 train_loss: 0.019762144108621267
epoch: 85 sim: 0.8173077702522278
epoch: 86 train_loss: 0.01960677446160739
epoch: 86 sim: 0.8223758935928345
epoch: 87 train_loss: 0.019495635263669413
epoch: 87 sim: 0.8257737755775452
epoch: 88 train_loss: 0.019247175140764575
epoch: 88 sim: 0.8176555633544922
epoch: 89 train_loss: 0.019145295758859607
epoch: 89 sim: 0.8233134746551514
epoch: 90 train_loss: 0.01887347029980284
epoch: 90 sim: 0.816766083240509
epoch: 91 train_loss: 0.018832483194982663
epoch: 91 sim: 0.8270451426506042
epoch: 92 train_loss: 0.01870051404526725
epoch: 92 sim: 0.8318674564361572
epoch: 93 train_loss: 0.018542775914025384
epoch: 93 sim: 0.8316900134086609
epoch: 94 train_loss: 0.018359905073103814
epoch: 94 sim: 0.8364368081092834
epoch: 95 train_loss: 0.018297128667965675
epoch: 95 sim: 0.829155445098877
epoch: 96 train_loss: 0.018107071245154635
epoch: 96 sim: 0.838212251663208
epoch: 97 train_loss: 0.01806159137477414
epoch: 97 sim: 0.8339556455612183
epoch: 98 train_loss: 0.017778068739242446
epoch: 98 sim: 0.8297494053840637
epoch: 99 train_loss: 0.017738772092012172
epoch: 99 sim: 0.8351496458053589
```

模型保存:

CAUNet_best_model.pth	2024/11/24 11:22	PTH 文件	10,849 KB
CAUNet_params_epoch_20.pth	2024/11/24 10:37	PTH 文件	10,849 KB
CAUNet_params_epoch_40.pth	2024/11/24 10:49	PTH 文件	10,849 KB
CAUNet_params_epoch_60.pth	2024/11/24 11:00	PTH 文件	10,849 KB
CAUNet_params_epoch_80.pth	2024/11/24 11:12	PTH 文件	10,849 KB
CAUNet_params_epoch_100.pth	2024/11/24 11:24	PTH 文件	10,849 KB

评估参数:

**SIM(相似系数) ACC(准确性)和 IOU(交并比) params 和 FLOPs**

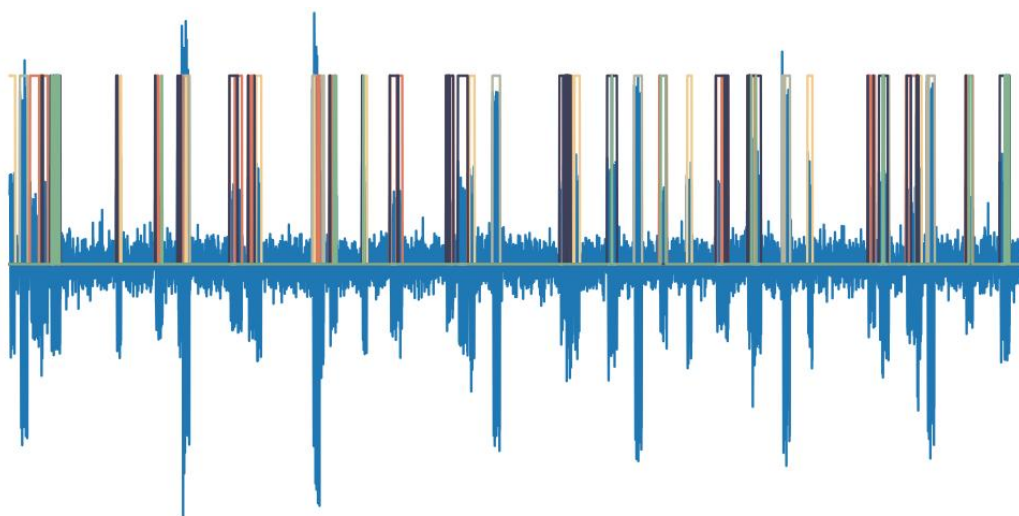
```
D:\conda\envs\my_env01\python.exe "D:\python f\pythonProject\CAUNet_code\CAUNet_code\test_flops.py"
[INF0] Register count_convNd() for <class 'torch.nn.modules.conv.Conv1d'>.
[INF0] Register count_normalization() for <class 'torch.nn.modules.batchnorm.BatchNorm1d'>.
[INF0] Register count_convNd() for <class 'torch.nn.modules.conv.ConvTranspose1d'>.
FLOPs: 2755.072
Params: 2.712165
```

```
tensor(0.6721, device='cuda:0', grad_fn=<DivBackward0>)
0.93425
0.3318089430894309
```

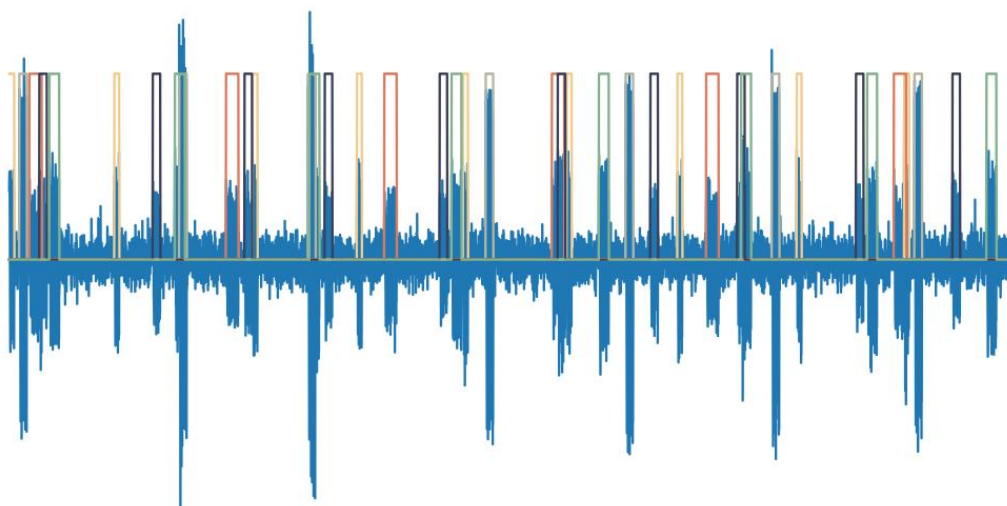
原始信号图像:



预测结果:



真实结果:



## GRAPH ATTENTION NETWORKS

### 论文主要内容:

通过引入基于自注意力机制的图神经网络层，高效处理图结构数据。

### 具体过程:

#### 1. 注意力分数的计算

输入点特征矩阵  $\mathbf{h}$

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \dots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$$

输出点特征矩阵  $\mathbf{h}'$

$$\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \dots, \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$$

$\mathbf{W}$  为一个共享线性矩阵 用于把输入维度  $F$  转化为输出维度  $F'$

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

$e_{ij}$  表示节点  $j$  的特征对节点  $i$  的重要性程度（注意力系数）。

通过一个全连接层，把  $\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j$  映射到一维

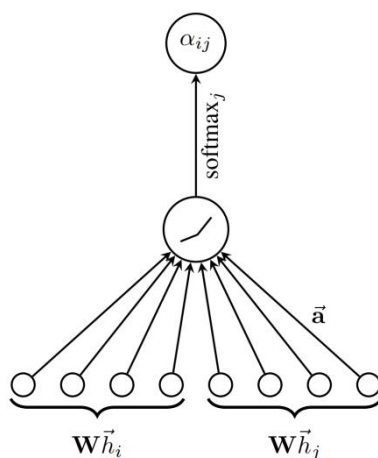
计算注意力分数:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}.$$

添加 LeakyReLU 来增加学习能力，最后的注意力分数为:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T [\mathbf{W}\vec{h}_i \parallel \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{a}^T [\mathbf{W}\vec{h}_i \parallel \mathbf{W}\vec{h}_k]\right)\right)}$$

整个过程为:



假设

$$\begin{matrix} h_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \xrightarrow{\mathbf{W}} \begin{matrix} h'_1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{matrix}$$

$F=3$   $F'=4$

## 2. 聚合

用注意力分数，乘以  $V(V=Wh_i)$  的值，再通过激活函数，就可以得到聚合后的新节点的特征值。

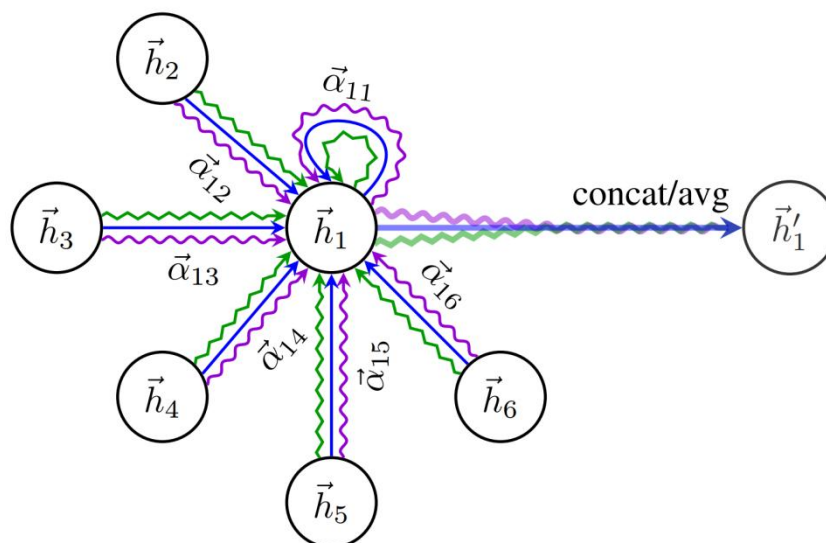
$$\vec{h}'_i = \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j \right)$$

当然，也可以设置不同的  $\mathbf{W}$  来获得多个注意力分数，再捏合在一起，计算和或者平均值。

$$\vec{h}'_i = \parallel_{k=1}^K \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

$$\vec{h}'_i = \sigma \left( \frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

多  $\mathbf{W}$  计算新节点特征值的流程图:



# SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

## 论文主要内容:

图卷积网络（GCN），通过高效的层级传播规则实现了大规模图上的半监督节点分类，显著提升了性能和计算效率。

## GCN 模型:

### GCN 层结构:

$$H^{(l+1)} = \sigma \left( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right).$$

### 损失函数:

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}, \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \|f(X_i) - f(X_j)\|^2 = f(X)^\top \Delta f(X).$$

其中  $\mathcal{L}_0$  为监督损失， $\mathcal{L}_{\text{reg}}$  为正则化损失。

$\mathcal{L}_0$  通常是交叉熵损失，用于训练有标签的节点。

$$\mathcal{L}_0 = - \sum_{l \in Y_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf}$$

$\mathcal{L}_{\text{reg}}$  为利用图的拓扑信息，增强模型对无标签节点的学习的正则化损失。

$$\mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \|f(X_i) - f(X_j)\|^2$$

## 具体过程:

### 1. 预备知识

邻接矩阵(A), 度矩阵(D), 拉普拉斯矩阵(L), 归一化拉普拉斯矩阵( $L_{\text{sym}}$ )

其中  $L = D - A$ ,  $L_{\text{sym}} = D^{-1/2} L D^{-1/2}$ 。

其中  $L$  和  $L_{\text{sym}}$  有以下性质(证明在另一个文档)

$$\hat{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

1)  $L_{\text{sym}}$  和  $L$  都是半正定实对称矩阵。

2)  $L$  可以分解为  $U \begin{bmatrix} N_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & N_n \end{bmatrix} U^T$ , 且  $U$  为正交矩阵。

3)  $L_{\text{sym}}$  的特征值范围在  $[0, 2]$ 。



## 2. 推导过程

得到其 GCN 网络层结构的主要核心思想就是，把拉普拉斯算子的特征函数  $e^{-i\omega t}$  转变为图(garph)中对应的拉普拉斯矩阵的特征向量。

传统的傅里叶变化定义：

$$F(\omega) = \mathcal{F}[f(t)] = \int f(t)e^{-i\omega t} dt$$

为信号  $f(t)$  与基函数  $e^{-i\omega t}$  的积分，而  $e^{-i\omega t}$  则为拉普拉斯算子的特征函数。

$$\Delta e^{-i\omega t} = \frac{\partial^2}{\partial t^2} e^{-i\omega t} = -\omega^2 e^{-i\omega t}$$

同样的，处理图(garph)问题的时候需要用到拉普拉斯矩阵，对应的也就需要去找拉普拉斯矩阵对应的基矩阵。

而拉普拉斯矩阵是一个半正定对称矩阵，则有以下结果：

$$LV = \lambda V$$

离散积分是一种内积形式，而其傅里叶变化为：

$$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^N f(i)u_i^*(i)$$

$u_l(i)$  表示第  $l$  个特征向量的第  $i$  个分量， $u_l^*(i)$  为其共轭。

所以其图(garph)上的傅里叶变化为：

$$\begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix} \quad \hat{f} = U^T f$$

由此可以推出，其傅里叶逆变换为：

$$f = U \hat{f}$$

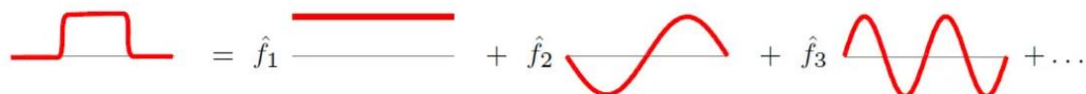
所以图卷积就是  $f$  和卷积核  $g$  的卷积，其表达式为：

$$\begin{aligned} f * g &= F^{-1}(F(f) \cdot F(g)) \\ &= F^{-1}(U^T f \cdot U^T g) \\ &= U(U^T f \cdot U^T g) \\ &= U \text{diag}(U^T g) U^T f \\ &= U \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \dots & \\ & & & \lambda_n \end{bmatrix} U^T f \\ &= U \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n) U^T f \\ &= U \Lambda_H U^T f \\ &= H f \end{aligned}$$

卷积核  $g$  的傅里叶  
变化对角矩阵为  
 $\begin{pmatrix} \hat{g}(\lambda_1) & & \\ & \ddots & \\ & & \hat{g}(\lambda_n) \end{pmatrix}$



f 和 g 卷积的就是 f 经过滤波器 H 处理后的输出结果。  
傅里叶变换把任意一个函数表示成了若干个正交函数的线性组合。



$$f = \hat{f}_1 + \hat{f}_2 + \hat{f}_3 + \dots$$

因为拉普拉斯矩阵的  $u_i$  全部为正交向量，其拉普拉斯矩阵的特征向量还是一组正交基，所以图(garph)的傅里叶变换也把图(garph)上定义的任意向量 f，表示成了拉普拉斯矩阵特征向量的线性组合，即：

$$f = \hat{f}(\lambda_1)u_1 + \hat{f}(\lambda_2)u_2 + \dots + \hat{f}(\lambda_n)u_n$$

对于传统的傅里叶变换，拉普拉斯算子的特征值  $\omega$  表示谐波频率，而拉普拉斯矩阵的特征值  $\lambda_i$  也表示图拉普拉斯变换的频率。  
这也就保证了其图(garph)卷积的公式是正确的。

综上所述，图卷积如何改进就在于其  $g_\theta(\Lambda)$  该如何选择。

下面默认

$$L = U\Lambda U^T$$

第一代 GCN

直接把  $\text{diag}(g(\lambda_i))$  变成了卷积核  $\text{diag}(\theta_i)$

$$g_\theta(\Lambda) = \begin{pmatrix} \theta_1 & & \\ & \ddots & \\ & & \theta_n \end{pmatrix} y_{\text{output}} = \sigma(U g_\theta(\Lambda) U^T x)$$

导致整个计算量过大，卷积核需要的  $n$  个参数，而且是全局连接。

第二代(基于多项式)

把  $g_\theta(\Lambda)$  设计成了求和的方式

$$g_\theta(\Lambda) = \begin{pmatrix} \sum_{j=0}^K \alpha_j \lambda_1^j & & \\ & \ddots & \\ & & \sum_{j=0}^K \alpha_j \lambda_n^j \end{pmatrix}$$

$$U \sum_{j=0}^K \alpha_j \Lambda^j U^T = \sum_{j=0}^K \alpha_j U \Lambda^j U^T = \sum_{j=0}^K \alpha_j L^j$$

$$y_{\text{output}} = \sigma \left( \sum_{j=0}^{K-1} \alpha_j L^j x \right)$$

- 1.卷积核只有  $K$  个参数，一般  $K$  远小于  $n$ ，参数的复杂度被大大降低了。
- 2.矩阵变换后，神奇地发现不需要做特征分解了，直接用拉普拉斯矩阵  $L$  进行变换，但是要计算  $L$  的高级次幂，复杂度仍然很高。
- 3.可以通过控制  $K$  来选择让聚合周围  $K$  个节点。

### 第三代(基于切比雪夫多项式)

$$g_{\theta}(\Lambda) = \sum_{k=0}^{K-1} \beta_k T_k(\tilde{\Lambda})$$

$T_k$  是  $k$  阶切比雪夫多项式， $\tilde{\Lambda}$  是重新缩放的特征值对角矩阵，进行这个变换的原因是第一类 Chebyshev 多项式的输入要在  $[-1, 1]$  之间。

第一类 Chebyshev 多项式为:

$$T_k(x) = \cos(k \cdot \arccos(x))$$

$\arccos$  的输入范围必须是  $[-1, 1]$ ，所以设置  $\tilde{\Lambda}$  为:

$$\frac{2\Lambda_{\hat{L}}}{\lambda_{max}} - I$$

进而得到:

$$\tilde{L} = 2\hat{L}/\lambda_{max} - I$$

这样就可以使用第一类切比雪夫多项式。

$$\begin{aligned} X^{l+1} &= HX^l \\ &= U\Lambda_H U^T X^l \\ &= U \sum \beta_k T_k(\tilde{\Lambda}_{-1,1}) U^T X^l \quad (1) \\ &= [\sum \beta_k T_k(U\tilde{\Lambda}_{-1,1}U^T)] X^l \quad (2) \\ &= [\sum \beta_k T_k(U(\frac{2\Lambda_{\hat{L}}}{\lambda_{max}} - I)U^T)] X^l \\ &= [\sum \beta_k T_k(\frac{2U\Lambda_{\hat{L}}U^T}{\lambda_{max}} - I)] X^l \\ &= [\sum \beta_k T_k(\frac{2\hat{L}}{\lambda_{max}} - I)] X^l \end{aligned}$$

(1)和(2)相等可以用数学归纳法证明:

最后得到其输出为:

$$y = \sigma \left( \sum_{k=0}^{K-1} \beta_k T_k(\tilde{L}) x \right)$$

将上面过程在一阶进行展开

$$T_k(\tilde{L}) = 2\tilde{L}T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L})$$

$$T_0(\tilde{L}) = I, T_1(\tilde{L}) = \tilde{L}$$

可以得到:

$$\begin{aligned} X^{l+1} &= HX^l \\ &= U\Lambda_H U^T X^l \\ &= \sum \beta_k T_k\left(\frac{2\hat{L}}{\lambda_{max}} - I\right) X^l \quad \text{一阶展开} \\ &= [\beta_0 T_0\left(\frac{2\hat{L}}{\lambda_{max}} - I\right) + \beta_1 T_1\left(\frac{2\hat{L}}{\lambda_{max}} - I\right)] X^l \\ &= [\beta_0 I + \beta_1 \left(\frac{2\hat{L}}{\lambda_{max}} - I\right)] X^l \quad \text{令 } \lambda_{max} = 2 \\ &= [\beta_0 I + \beta_1 (\hat{L} - I)] X^l \\ &= [\beta_0 I + \beta_1 (I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} - I)] X^l \\ &= [\beta_0 I - \beta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}}] X^l \quad \text{令 } \beta_1 = -\beta_0 \\ &= [\beta_0 I + \beta_0 D^{-\frac{1}{2}} A D^{-\frac{1}{2}}] X^l \\ &= \beta_0 (D^{-\frac{1}{2}} A D^{-\frac{1}{2}} + I) X^l \\ &= \beta_0 (D^{-\frac{1}{2}} A D^{-\frac{1}{2}} + D^{-\frac{1}{2}} D D^{-\frac{1}{2}}) X^l \\ &= \beta_0 D^{-\frac{1}{2}} (A + D) D^{-\frac{1}{2}} X^l \end{aligned}$$

最后再对其归一化得到:

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$$

也就是最后 GCN 模型中的结构。

### 3. 半监督学习中进行点分类

两层的 GCN 模型为:

$$Z = f(X, A) = \text{softmax}\left(\hat{A} \text{ReLU}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right).$$

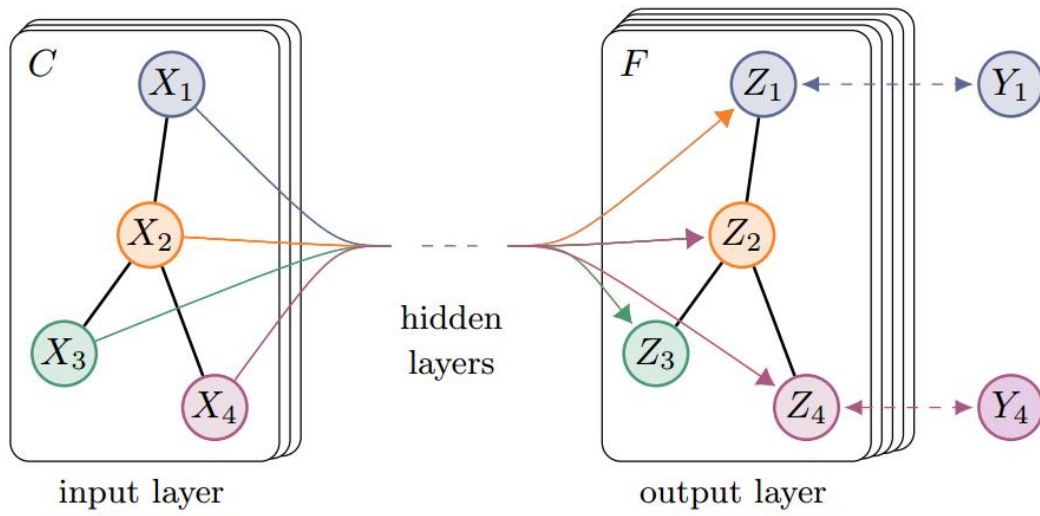
其中

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

损失函数为:

$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf},$$

两层 GCN 结构:



## 监督损失 ( $\mathcal{L}_{\text{sup}}$ ) 的参数

$$\mathcal{L}_{\text{sup}} = - \sum_{i \in \mathcal{Y}_L} \sum_{c=1}^C Y_{ic} \log P_{ic}$$

### 1. $\mathcal{Y}_L$ :

- 含义: 标注节点的集合, 表示图中哪些节点的标签已知。
- 作用: 在监督损失中, 优化仅发生在这些标注节点上。

### 2. $C$ :

- 含义: 类别总数, 表示每个节点可能属于的分类类别数量。
- 作用: 监督损失需要针对每个类别计算模型预测的概率。

### 3. $Y_{ic}$ :

- 含义: 节点  $i$  在类别  $c$  下的真实标签值 (one-hot 编码)。
  - $Y_{ic} = 1$ : 如果节点  $i$  属于类别  $c$ 。
  - $Y_{ic} = 0$ : 否则。
- 作用: 引导模型在训练中优化, 使得预测值  $P_{ic}$  尽可能接近  $Y_{ic}$ 。

### 4. $P_{ic}$ :

- 含义: 模型预测的节点  $i$  属于类别  $c$  的概率, 通常是通过 softmax 函数得到:

$$P_{ic} = \frac{\exp(Z_{ic})}{\sum_{c'} \exp(Z_{ic'})}$$

- $Z_{ic}$ : 隐藏层输出的特征, 经过权重变换后得到的分类得分。
- 作用: 衡量模型的预测准确性。

## 正则化损失 ( $\mathcal{L}_{\text{reg}}$ ) 的参数

$$\mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \|f(X_i) - f(X_j)\|^2$$

### 1. $A_{ij}$ :

- 含义: 图的邻接矩阵元素, 表示节点  $i$  和  $j$  是否相连。
  - $A_{ij} = 1$ : 如果节点  $i$  和  $j$  相连。
  - $A_{ij} = 0$ : 否则。
- 作用: 控制正则化损失只作用于相邻节点。

### 2. $f(X_i)$ 和 $f(X_j)$ :

- 含义: 节点  $i$  和  $j$  的特征嵌入, 表示经过图卷积网络后的节点特征。
- 作用: 通过最小化嵌入差异, 强制相连节点的嵌入尽可能接近。

### 3. $\|\cdot\|^2$ :

- 含义: 节点嵌入之间的欧氏距离 (平方)。
- 作用: 测量相邻节点特征嵌入之间的相似性。